ROBOMAT 07

Helder Araújo
Maria Isabel Ribeiro

Coimbra, Portugal,
17-19 September, 2007

29

ISBN: 978-989-95011-3-3
Título: ROBOMAT07
Autor: Araújo, Helder e Ribeiro, Maria Isabel
Data: 11-03-2008
Editor: Centro Internacional de Matemática
Morada: Almas de Freire, Coimbra
Código Postal: 3040 COIMBRA
Correio Electrónico: cim@mat.uc.pt
Telefone: 239 802 370
Fax: 239 445 38

Contents

Invited Presentations

Graph-theoretic methods for multi-agent coordination

Magnus Egerestedt*

Abstract

By ignoring the geometric constraints that inevitably govern inter-robot interactions in decentralized robot networks, a purely combinatorial description of the network is obtained. In fact, it can be described as a graph, with vertices corresponding to the individual robots, and edges corresponding to the existence of an inter-robot communication (or sensing) link. In this note, we report on some of the recent results that have emerged in the general area of graph-based multi-agent control. Most notably of these might be the consensus equation that allows us to drive a scalar state value to the same value for the different robots, in a completely decentralized fashion.

1 Introduction: combinatorics vs. geometry

The emergence of decentralized, mobile multi-agent networks, such as distributed robots, mobile sensor networks, or mobile ad-hoc communications networks, has imposed new challenges when designing control algorithms. These challenges are due to the fact that the individual agents have limited computational, communications, sensing, and mobility resources. In particular, the information flow between nodes of the network must be taken into account explicitly already at the design phase and a number of approaches have been proposed for addressing this problem, e.g. [6, 7, 8, 9, 10, 13, 18].

Regardless of whether the information flow is generated over communication channels or through sensory inputs, the underlying geometry is playing an important role. For example, if an agent is equipped with omnidirectional range sensors, it can only detect neighboring agents if they are located in a disk around the agent. Similarly, if the sensor is a camera, the area becomes a wedge rather than a disk. But, to make the interaction geometry explicit when designing control laws is not an easy task, and an alternative view is to treat interactions as purely combinatorial. In other words, all that matters is whether or not an interaction exists

*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. E-mail:magnus@ece.gatech.edu. This work was supported by the U.S. Army Research Office through the grant # 99838.

between agents, and under certain assumptions on the global interaction topology, one can derive remarkably strong and elegant results. (For a representative sample, see [6, 13, 18].) What then remains to be shown is that the actual geometry in fact satisfies the combinatorial assumptions.

This view, i.e. that the geometry is abstracted down to a purely combinatorial relationship allows the control laws to be designed based without explicit dependency on the geometry of the system. Rather, the control laws will end up depending solely on certain topological properties of the network, such as connectivity or balancing. As such, one has effectively traded away a hard problem (the geometric problems are for example addressed in [1, 9]) for a simple problem under some assumptions that may or may not always be valid (e.g. [7]). And, in this note, we discuss some of the main results in the area of graph-based (or combinatorial) multi-agent control in the domain of multi-agent robotics.

2 Algebraic graph theory and proximity graphs

2.1 Basic notation

Assume that the multi-robot system consists of N agents, evolving in a d-dimensional state space, i.e. that $x_i \in \Re^d,\ i = 1, \ldots, N$. Let $V = \{1, \ldots, N\}$ be a set of vertices in a graph G, corresponding to the identity of the robots. Moreover, let the graph $G = (V, E)$, where the edge set $E \subset V \times V$ is a set of unordered pairs of vertices. The interpretation is that an edge (v_i, v_j) is in E if agents i and j can interact with each other.[1]

Graphs are combinatorial objects (sets and pairwise relations between elements in he sets). In order to endow these objects with algebraically manipulable items, such as matrices, one has to look at the area of *algebraic graph theory*. (See for example [5].) For example, some standard matrices associated with G are the *degree matrix* D and the *adjacency matrix* A. The degree matrix is a diagonal matrix $D = \text{diag}(\deg(v_1), \ldots, \deg(v_N))$, where $\deg(v_j)$ is the degree of vertex v_j, i.e. the number of vertices adjacent to v_j. We will let N_j denote the set of adjacent, or neighboring nodes, i.e. $N_j = \{v_i \mid (v_i, v_j) \in E\}$ and hence $|N_j| = \deg(v_j)$. The adjacency matrix $A = [a_{ij}] \in \{0, 1\}^{N \times N}$, where

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Another matrix of fundamental importance in algebraic graph theory is the *graph Laplacian* $L = D - A$, which has the following key properties:

- $L = L^T \succeq 0$, i.e. it is positive semi-definite.
- If the graph is connected, i.e. there exists a path between any two vertices, then

[1] In this note, we will assume that the edges are undirected, i.e. that $(v_i, v_j) \in E \ \Leftrightarrow \ (v_j, v_i) \in E$.

– The (ordered), non-negative, real eigenvalues of L satisfies $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_N$.

– $\text{null}(L) = \text{span}\{\mathbf{1}\}$, i.e. $L\mathbf{1} = 0$, where $\mathbf{1} = (1, \ldots, 1)^T$.

These facts about the graph Laplacian will play an important role in subsequent sections.

2.2 Proximity graphs

The way geometry enters into the picture is through so-called *proximity graphs* [9]. The idea here is that edges in the graph exist when the underlying geometry satisfies certain properties. For example, if the robots, whose positions are $x_1, \ldots, x_N \in \Re^d$, are all equipped with omni-directional range sensor, with an effective range δ, the induced δ-disk proximity graph is $G(t) = (V, E(t))$. Here the vertex set is $V = \{v_1, \ldots, v_N\}$, and $(v_i, v_j) \in E(t) \Leftrightarrow \|x_i(t) - x_j(t)\| \leq \delta$. An example of such a graph is given in Figure 1.

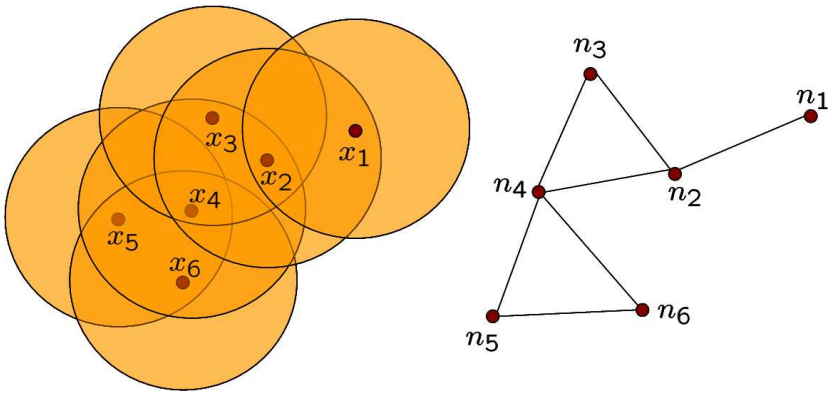


Figure 1: δ-disk proximity graph.

One should note that as the agents move around, neighbors (i.e. adjacent vertices) may be introduced or lost. As such, the graph is no-longer a static structure and this type of graph is referred to as a *dynamic* graph. Other types of recently studied proximity graphs are the Gabriel graphs, Vornoi graphs, and DeLaunay graphs, just to name a few [9, 16].

3 Consensus problems

3.1 Static networks

The consensus problem is in essence a problem involving having multiple agents reach an agreement about a scalar state value over a network. This problem is a canonical problem in decentralized coordination and it can be solved quite elegantly using algebraic graph theory.

One instantiation of the consensus problem is the so-called *rendezvous* problem where a collection of agents are supposed to meet at an unspecified common location. In particular, we will assume that each agent is located at x_i and that it can only measure the relative position of its neighboring agents, i.e. it can measure $x_i - x_j,\ \forall j \in N_i$. Moreover, assuming that each agent has single-integrator dynamics, i.e. $\dot{x}_i = u_i$ one reasonable control strategy is to drive each agent towards the centroid of its neighboring set as

$$\dot{x}_i = -\sum_{j \in N_i} (x_i - x_j),$$

where N_i is the set of robots adjacent to robot i. In fact, in the proceeding paragraphs, we will assume that the underlying graph is static. Based on this assumption, the above equation can be rewritten as

$$\dot{x}_i = -\deg(v_i)x_i + \sum_{i=1}^{N} a_{ij}x_j,$$

where, as before, $\deg(v_i)$ is the degree of node i, and a_{ij} is the (i,j):th entry in the adjacency matrix.

Now, if we for the sake of argument, assume that $x_i \in \Re$, i.e. each robot evolves in a one-dimensional space, then by recalling that $L = D - A$, the above equation can be rewritten as

$$\dot{x} = -Lx,$$

where $x = (x_i, \ldots, x_N)^T$. Note that this is a standard, linear, time-invariant system whose stability properties are entirely given by the eigenvalues to $-L$. But, we already know that as long as G is connected, then L is positive semidefinite, and, as such, $-L$ is negative semidefinite. In fact, we know that $-L$ has a single 0 eigenvalue and all other eigenvalues are negative and real. As such, we have that the system is stable and that x will tend to the null-space of L asymptotically. In other words, $x_i \to \alpha$ as $t \to \infty,\ \forall i$, where $\alpha \in \Re$. This, by now widely utilized consensus-equation has appeared with a number of variations, e.g. [6, 10, 13, 15].

The interpretation here is that all components of x (i.e. the scalar positions of all the robots) will tend to the same value. And hence the rendezvous problem is solved. In fact, it is easy to establish that the centroid

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

is static. And as a consequence $\alpha = \bar{x}$, i.e. *all agents will tend asymptotically to the static centroid of the robot team.* In fact, in [11], it was shown that the rate of convergence to the centroid is given by

$$\|x(t) - \bar{x}\mathbf{1}\| \leq \|x(0) - \bar{x}\mathbf{1}\|e^{-\lambda_2 t},$$

where λ_2 is the second smallest eigenvalue of the graph Laplacian.

If $x_i \in \Re^d$ one can directly note that it is possible to decouple the dynamics along each dimension, i.e. if we let $\text{comp}(x, j) = (x_{1,j}, \ldots, x_{N,j})^T$, where $x_i = (x_{i,1}, \ldots, x_{i,N})^T$, then we have

$$\frac{d\text{comp}(x, j)}{dt} = -L\text{comp}(x, j), \ j = 1, \ldots, d.$$

As such, the previous argument can be applied in this case as well. An example of running the consensus algorithm is shown in Figure 2, where 10 agents have to reach an agreement (or consensus). Note that even though they all start out with different values, they quickly converge to a common value based solely on local information.

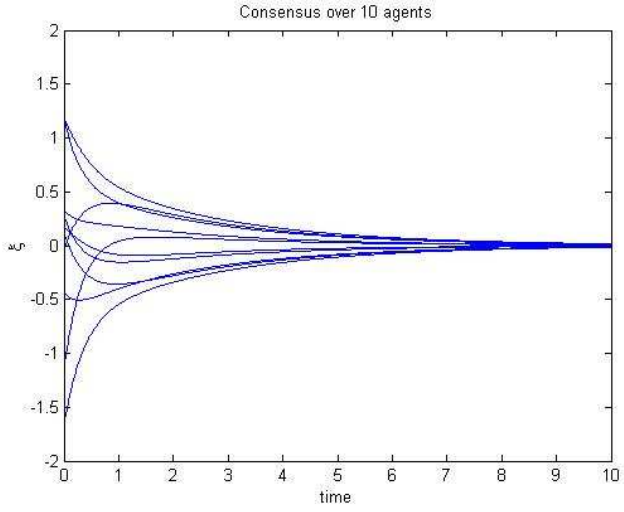


Figure 2: Application of the consensus algorithm.

3.2 Dynamic networks

Note that so far we have assumed that G is connected and *static*. However, as shown in [6, 7, 8, 18], the above argument still holds as long as the graph stays connected. The connection between combinatorics and geometry is thus made through the assumption that the underlying geometry satisfies this key assumption. However, if we assume that the underlying graph is a disk graph, this assumption may not always hold, as shown below, in Figure 3.

3.3 Linear formation control

One reason why the consensus idea is so powerful is that even though the rendezvous problem may be of limited use per se, we can still apply the same thinking to a number of problem

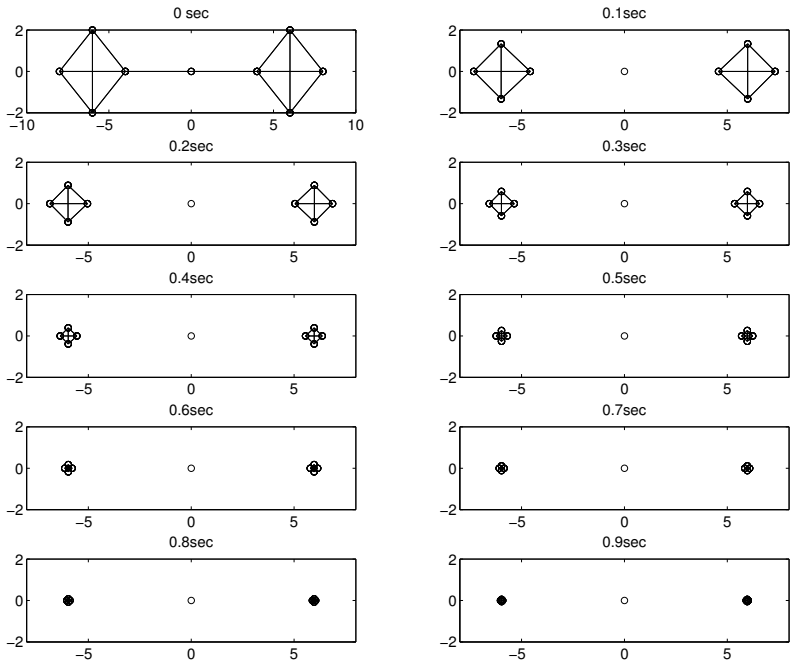


Figure 3: A progression is shown where connectedness is lost even though the initial graph is connected.

formulations, including coverage control (e.g. [1, 2]), containment control [4], distributed filtering [12], and *formation control*, e.g. [3, 7, 15].

In this context, formations are specified in terms of a collection of desired inter-agent distances $d_{ij},\ (i,j) \in D \subset \{1,\dots,N\}^2$. If we assume that these distances are feasible, i.e. there exist points $y_1,\dots,y_N$ such that $\|y_i - y_j\| = d_{ij},\ \forall (i,j) \in D$, we can let the relative errors be given by $\gamma_i = x_i - y_i$.

Now, by observing that if we can drive all relative errors to the same value, we have achieved a pure translation of the points $y_1,\dots,y_N$. As such, one can attempt to solve the formation control problem by simply running a consensus equation over the relative errors [3, 7], as

$$\dot{\gamma}_i = -\sum_{j\in N_i} (\gamma_i - \gamma_j).$$

But, noting that $\dot{\gamma}_i = \dot{x}_i$ and $\gamma_i - \gamma_j = x_i - x_j - (y_i - y_j)$, we get that each agent should move as

$$\dot{x}_i = -\sum_{j\in N_i} (x_i - x_j - \zeta_{ij}),$$

where $\zeta_{i,j} = y_i - y_j$.

One should keep in mind that even though these are elegant results, they all hinge on the fact that connectivity is preserved. And, as shown above, this is not always the case. Recently, a number of papers dealing with the issue of preserving connectedness through nonlinear weights have appeared [3, 7].

4 Controllability and anchor networks

Another area where graph-based methods for multi-agent coordination have proved useful are for networks where some agents take on special, so-called leader (or anchor) roles [17].

4.1 Leader-Follower Structures

Assume that a single agent (let's say robot N) is stationary while the others are executing the consensus equation discussed in the previous section, as

$$\begin{aligned} \dot{x}_i &= -\textstyle\sum_{j\in N_i}(x_i - x_j),\ i = 1, \ldots, N-1 \\ \dot{x}_N &= 0, \end{aligned}$$

one can wonder if rendezvous (or consensus) is still achieved. The answer to this question is yes, and, as long as the network stays connected, all agents will converge to the static leader (or anchor) agent, as shown in [17].

This fact is interesting since it essentially allows us to control the network by moving the leader agent around. In fact, as long as it moves slow enough (as compared to the convergence rate of the consensus equation) we can expect the other agents to follow the leader agent rather closely.

Moreover, if we have a number of stationary leader agents, it was shown in [4] that the remaining agents will in fact converge to the convex hull spanned by the leader agents. This observation moreover allows us to exert boundary value control of the network by changing the shape of this convex hull, as was the case in [4], and as is illustrated in Figure 4.

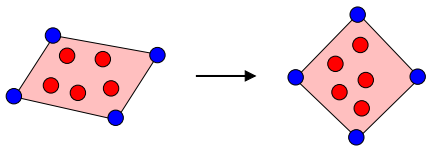


Figure 4: The containment problem: The leaders move in such a way that the followers remain in the convex leader-polytope for all times.

4.2 Graph-based controllability

If we assume that the network is static and that the first $N - N_l$ agents are followers, and the last N_l agents are leaders, we can decompose the graph Laplacian as

$$L = \left[\begin{array}{c|c} L_l & \ell \\ \hline \ell^T & L_f \end{array}\right].$$

Here $L_l \in \Re^{(N-N_l)\times(N-N_l)}$, $\ell \in \Re^{(N-N_l)\times N_l}$, and $L_f \in \Re^{N_l \times N_l}$. If we as before, without loss of generality, assume that $x_i \in \Re,\ i = 1, \ldots, N$, we can now let $x = (x_1, \ldots, x_{N-N_l})^T$. Moreover, if we assume that the followers are executing the consensus equation while we can control the position of the leaders directly (or it's velocities - it does not matter from a controllability point-of-view), we can let $u = (x_{N-N_l+1}, \ldots, X_N)^T$. The corresponding control system thus becomes

$$\dot{x} = -L_l x - \ell u.$$

One can thus ask the following question: For what topologies does the above equation correspond to a completely controllably system? In other words, if (L_l, ℓ) was a controllable pair, we could drive the followers to whatever position we would like. And, as it turns out, tools from algebraic graph theory once again help us understanding this issue.

Below, in Figure 5, are given three different graphs. The first two are not controllable and the reason for this is that the followers are somehow symmetric with respect to the leader, i.e. if $x_1(0) = x_2(0)$ then $x_1(t) = x_2(t),\ \forall t \geq 0$. This is not the case in the third case.

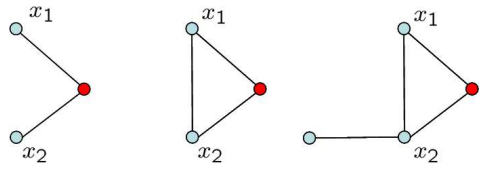


Figure 5: Two uncontrollable graph structures (left and middle) and one controllable (right).

Technically, what happened here is that in the uncontrollable cases, it was possible to relabel the non-leader agents while still maintaining the same edge relations. Technically speaking, such an adjacency preserving vertex permutation is called a *graph automorphism*. In other words, $\psi : V \to V$ is a rgaph automorphism if $(v_i, v_j) \in E \ \Leftrightarrow\ (\psi(v_i), \psi(v_j)) \in E$. And, in [14], it was shown that a sufficient (and in some cases necessary and sufficient) condition for a single leader network to be uncontrollable is that there exists a non-trivial (not the identity) graph automorphism, with $\psi(v_N) = v_N$.

For multiple leaders, things become more complicated, but analogous sufficient conditions for uncontrollability have been found in [14] based on so-called *equitable partitions* of the graph. (Interested readers are referred to [14].)

5 Conclusions

In this note, we report on some of the recent results that have emerged in the general area of graph-based multi-agent control. In fact, by focusing purely on the combinatorial nature of the network (and thus ignoring the geometric constraints on the inter-robot interactions) a number of powerful results can be obtained. Most notably of these might be the consensus equation that allow us to drive a scalar state value to the same value for the different robots, in a completely decentralized fashion. This is possible as long as the network stays connected, which is an assumption that one may or may not always be justified in making.

References

[1] J. Cortes, S. Martinez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, 2006.

[2] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[3] D. V. Dimarogonas and K. J. Kyriakopoulos. On the rendezvous problem for multiple nonholonomic agents. *IEEE Transactions on Automatic Control*, 52(5):916–922, 2007.

[4] G. Ferrari-Trecate, M. Egerstedt, A. Buffa, and M. Ji. Laplacian sheep: A hybrid, stop-go policy for leader-based containment control. In *Hybrid Systems: Computation and Control*, pages 212–226, Santa Barbara, CA, March 2006. Springer-Verlag.

[5] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer, 2001.

[6] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[7] M. Ji and M. Egerstedt. Distributed coordination control of multi-agent systems while preserving connectedness. *IEEE Transactions on Robotics*, 23(4):693–703, 2007.

[8] Z. Lin, B. Francis, and M. Maggiore. Necessary and sufficient graphical conditions for formation control of unicycles. *IEEE TRansactions on Automatic Control*, pages 121–127, 2005.

[9] S. Martinez, J. Cortes, and F. Bullo. Motion coordination with distributed information. *IEEE Control Systems Magazine*, 27(4):75–88, 2007.

[10] M. Mesbahi. On state-dependent dynamic graphs and their controllability properties. *IEEE Transactions on Automatic Control*, 50(3):387–392, 2005.

[11] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. on Automatic Control*, 51(3):401–420, 2006.

[12] R. Olfati-Saber. Distributed Kalman filtering for sensor network. *IEEE Conference on Decision and Control*, December 2007.

[13] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[14] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt. Controllability of multi-agent systems from a graph-theoretic perspective. SIAM Journal on Control and Optimization, to appear.

[15] W. Ren and R. W. Beard. *Distributed Consensus in Multi-Vehicle Cooperative Control*. Communication and Control Engineering Series. Springer Verlag, New York, 2007.

[16] B. Shucker, T. D. Murphey, and J. Bennett. Switching rules for decentralized control with simple control laws. *American Control Conference*, 2007.

[17] H. G. Tanner. On the controllability of nearest neighbor interconnections. *IEEE Conference on Decision and Control*, December 2004.

[18] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. *Transactions on Automatic Control*, 52(5):863–868, 2007.

Grammars and their many incarnations

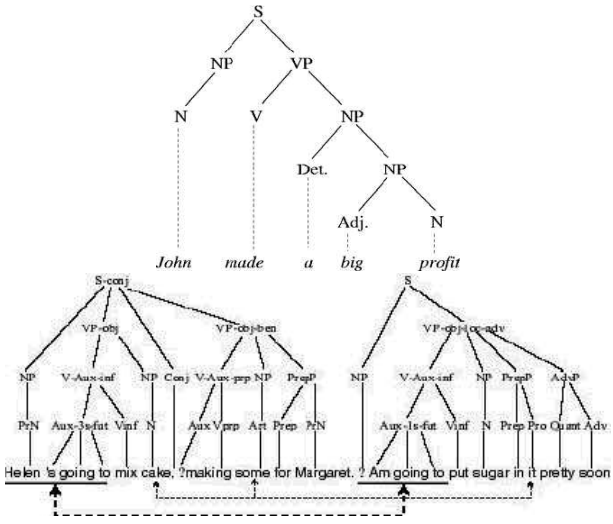
David Mumford*

Outline

- Grammars in language (Panini's Sanskrit grammar, 500 BCE? to Chomsky), the basic ideas
- Generalizations to grammars of perception, plans and thoughts
- Grammars in Vision, Segmentation, Gestalt rules, perspective rules. (S.Geman, S.-C. Zhu, J.-M. Morel).

1 Two parse trees

From the speech of a $2\frac{1}{2}$ year old:

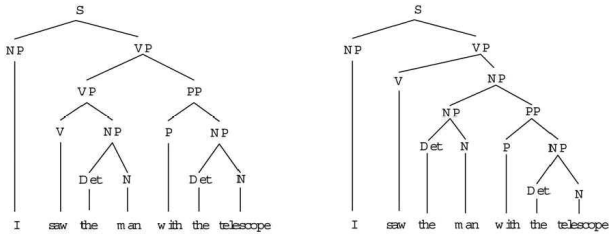


*Division of Applied Mathematics, Brown University, USA. E-mail: David_Mumford@brown.edu.

2 The basic ideas

- Re-usable parts = subsets of the utterance that are meaningful in themselves and so can be moved intact into other utterances.
- Give these labels to denote which parts are interchangeable
- Production rules, agreement
- Probabilities (ambiguities, ungrammatical speech, semantics vs. syntax)

3 Parsing is not unique: ambiguity



"Time flies like an arrow"

a. Time = subject, flies = verb OR

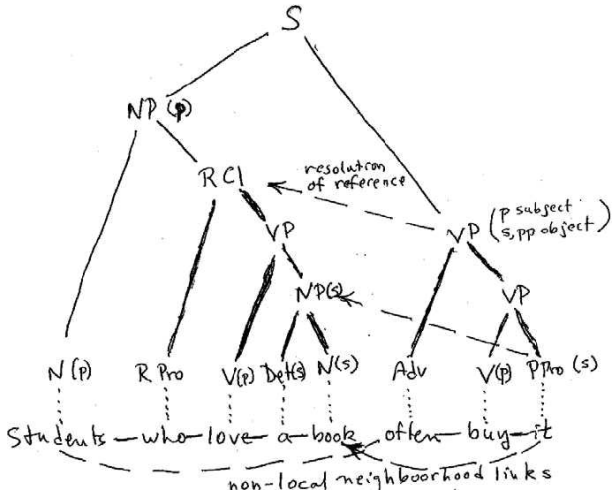
b. Time = verb, flies = object OR

c. Time = adjective, like = verb

4 Random branching trees incorporate probabilities

- Start at the root. Each node decides to have k children with probability p_k, where $\sum p_k = 1$. Continue infinitely or until no more children.
- $\lambda = \sum k p_k$ is the expected number of children; if $\lambda \leq 1$, then the tree is a.s. finite; if $\lambda > 1$, it is infinite with pos. prob.
- Can put *labels*, from some finite set L, on the nodes and make a labelled tree from a prob. distr. which assigns probabilities to a label $\{\lambda\}$ having k children with labels $(\widehat{\lambda}_1, \ldots, \widehat{\lambda}_k)$.
- This is identical to what linguists call PCFG's (=probabilistic context-free grammars). For them, L is the set of attributed phrases (e.g. 'singular feminine noun phrases') plus the lexicon (which can have no children) and the tree is assumed a.s. to be finite.

5 Non-local dependencies go beyond random branching trees



6 Context sensitive grammars

$$P(\omega) = Q(L(\text{root})) \cdot \prod_{v \in V} P_{L(v)}(R(v)) \cdot \prod_{e=\langle v,w \rangle \in E} B_e\left(\alpha(v), \alpha(w), \alpha(\text{parents})\right)$$

where

$$\begin{aligned} \omega &= \text{parse graph} \\ V &= \text{vertices} \\ E &= \text{edges} \\ L &= \text{labels} \\ R &= \text{production rule} \\ \alpha &= \text{attributes} \end{aligned}$$

Simplest case (Grenander, Mark and Miller): add horizontal edges between adjacent terminals of a context free parse tree and add bigram probabilities B_e.

7 Parse graphs of other types

- In vision, nodes = parts of image which are naturally grouped and can occur (with some changes) in other images.
- In robotics, nodes = parts of an action or plan which are naturally grouped and can occur (with some changes) in other actions or plans.

Parts may *include each other* giving vertical edges. Parts may *abut in space and time* or share attributes, hence horizontal edges.

8 Some comments on grammars in robotics

- The parse tree of plans and actions gives plans and actions a natural recursive structure.
- The attributes of each node include some sort of location in space-time and production rules must enforce spatio-temporal compatibility
- Multiple production rules gives plans with alternate futures, overlays on space-time.
- Broca's area in the brain is adjacent to fine hand motor control: is this an accident? Both seem to have grammatical skills (P.Lieberman).

9 Are there parse graphs for all thoughts?

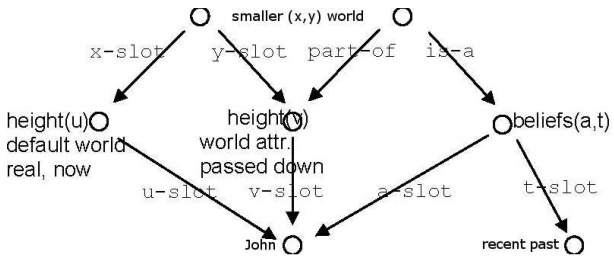
First, what is a *thought*? My version:

A constellation of objects, events and actions in one or more worlds

(here 'worlds' are locations in space and time, past or future, either the presumed 'real' world or the internal world of an agent or a made up world)

My conjecture: a thought has a parse graph. A constellation has natural subsets, groupings which re-occur in many other situations and these are the nodes. Like sentences and images, which are made up of phonemes/pixels, an abstract thought has its smallest constituent objects and actions. Some groupings are part of bigger ones, giving rise to edges. As in any grammar, groupings have *labels* and *attributes*.

10 Example: 'John turned out not to be as tall as he thought he was'



11 Related but distinct graphs

- *Actions* have another sort of edge:

  The ur-action: the thought of a baby in a cradle: "(I) hit rattle"

  We get graphs with directed edges for *causality*. (Bayesian belief networks).

- Traditional language grammars use *'slot-filler'* edges, and their specific productions.

- *Semantic nets* are graphs with '*is-a*' edges, a partial ordering on the set of parse graph labels.

- *Roget's thesaurus* leads to the mother of all graphs among words, analogous to the adjacency edges in the vision example, but mixing labels/attributes.

- *Neurally*, a thought could be instantiated by multiple cell-assemblies coordinated through inter-area white matter fibres, which form a huge directed graph.

12 Grammars in Vision

- An image is 2 dimensional, so the right use of parse trees was not clear (cf. K.-S. Fu).

- The gestalt school of visual psychophysics (c.1900-1950) had studied how parts of images are grouped.

- Computer vision is beginning to recognize how central this structure is.

13 The gestalt rules of grouping

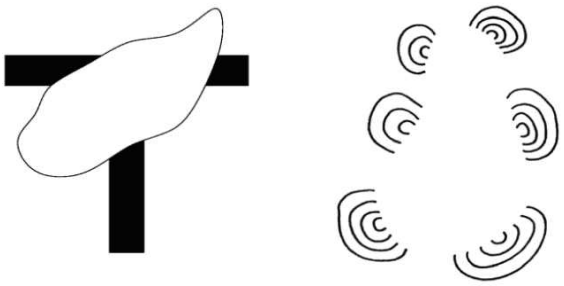
(Metzger, Wertheimer, Kanisza,...)

Elements of images are linked on the basis of:

- Proximity and Similar color/texture These are the factors used in segmentation

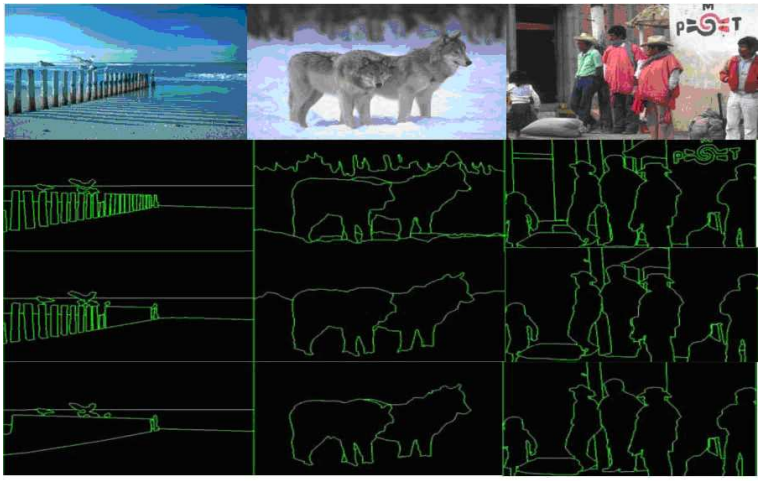
- Good continuation This is studied as contour completion

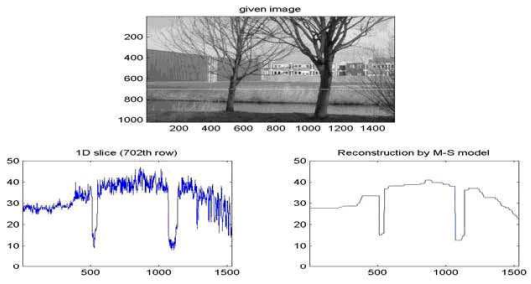
- Parallelism, Symmetry, Convexity Higher order properties Reconstructed edges and objects can be amodal as well as modal:



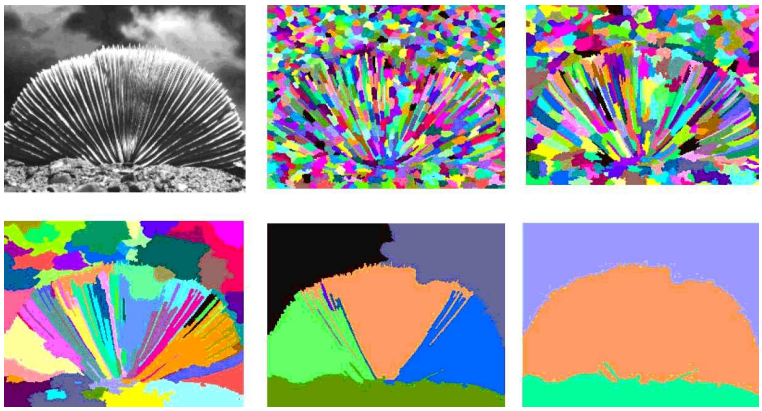
14 The simplest level of grouping: image segmentation

(3 images from the Malik database, each with 3 human segmentations)

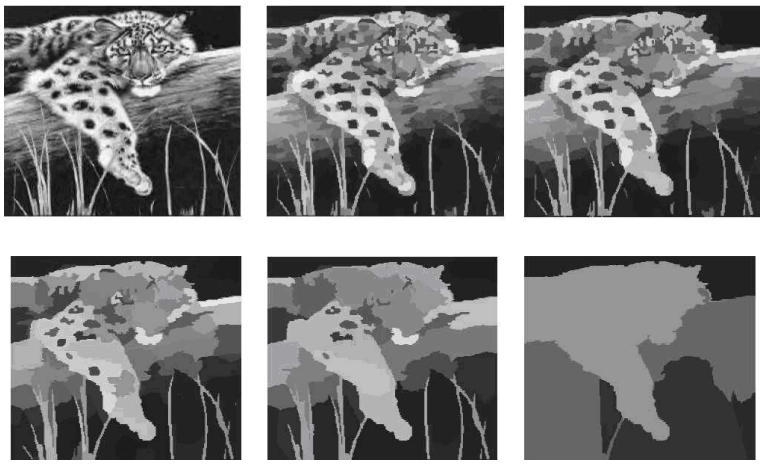




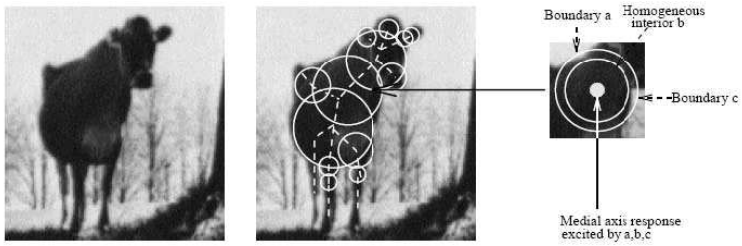
15 Algebraic Multigrid algorithm of Galun, Sharon, Basri and Brandt



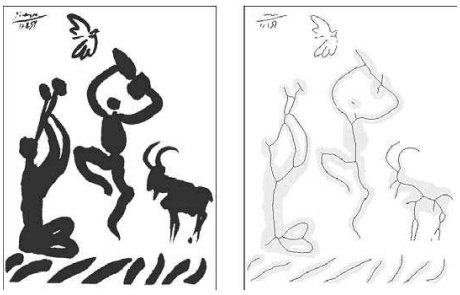
The segmentation of a shell by progressive grouping, the set of colored regions at each level forming the nodes of the grid at that level and the grouping following weights obtained by aggregating statistics from below. We have a parse tree here, but needs pruning.



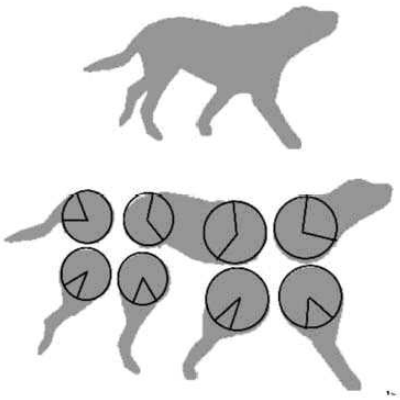
16 The next level of grouping: structuring shapes via their axes



The medial axis encodes symmetry between opposite sides of objects. The rapid way to decode shape:

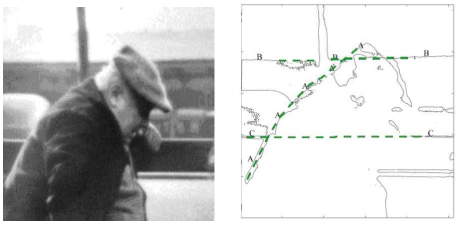


17 The medial axis defines a decomposition of the shape into parts, 'ribbons' and 'blobs' (Kimia)

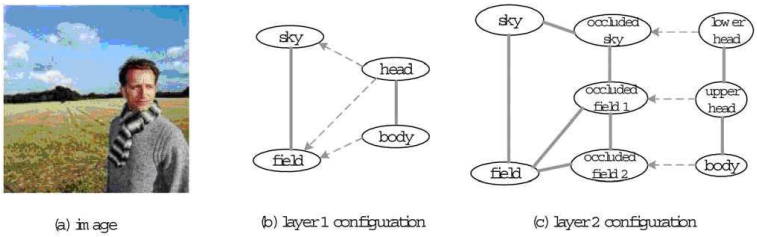


18 The most characteristic grouping: good continuation of lines

'Edge detectors' often fail to trace subjectively obvious contours (the man's back); other contours continue each other even though a middle part is occluded (amodal contours).



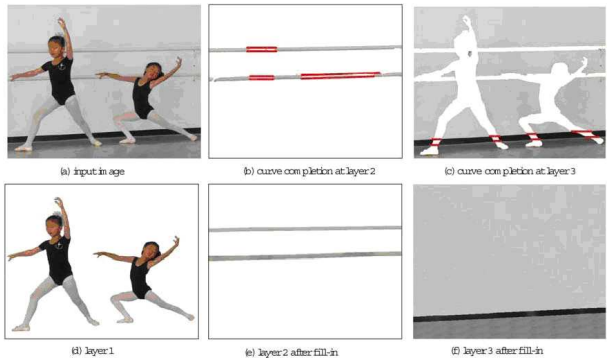
19 Images are 2D projections of 3D scenes. Occlusion is a major clue and can be represented in a parse tree with partial 3D attributes



(a) image (b) layer 1 configuration (c) layer 2 configuration

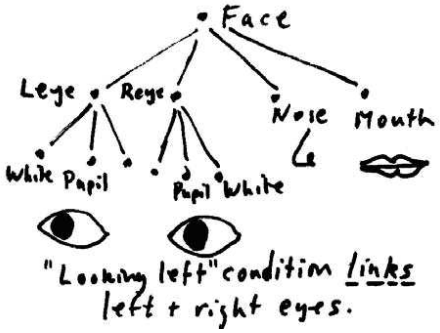
An amodal contour is constructed across the man's head continuing the sky/field boundary. The dotted horizontal edges represent depth inequalities, 'in-front-of' links.

Another example



20 Some complex groupings

A face has bilateral symmetry. In addition this symmetry can be broken and this may be a significant thing to note.



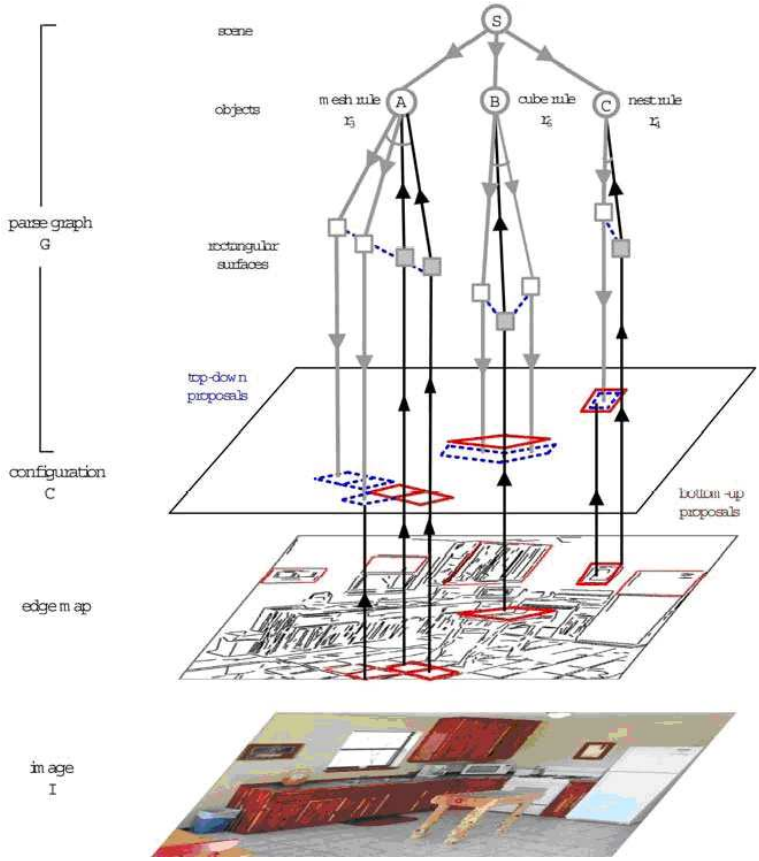
Incorporating all the gestalt grouping rules to formalize Renaissance Perspective (Zhu). Here is a simple kitchen scene:

a) The tiled floor,

b) The cubical butcher block tabletop,

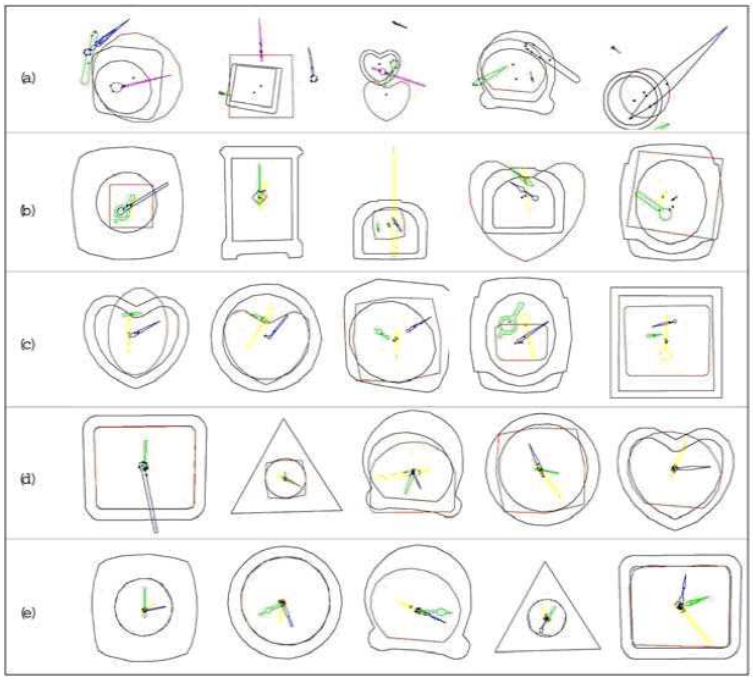
c) The frame around the picture,

are all discovered (Han, Zhu).



21 Stochastic grammars can be learned

Learning the potentials in the grammar model for clock faces. Successive stages correspond to minimax learning of a) part positions, b) relative scales, c) the 'hinge' relation and d) the containment relation (Porway, Yao, Zhu).



22 Conclusions

- Grammar is ubiquitous.
- Working out the parse is central to understanding the percept, thought or action.
- General grammars are, however, more complex: they need horizontal as well as vertical edges, their stochastic models have many new terms which can be learned.
- I hope this can and will be developed in the robotics context.

Other Presentations

A POMDP approach to cooperative localization in sparse environments

Francisco S. Melo* Isabel Ribeiro*

Abstract

In this paper we discuss how communication can be used advantageously for cooperative navigation in sparse environments. Specifically, we analyze the tradeoff between the cost of communication cost and the efficient completion of the navigation task. We make use of a partially observable Markov decision process (POMDP) to model the navigation task, since this model allows to explicitly consider the tradeoff between information-gathering actions and actions that move the robot towards the goal. By explicitly including communication in the POMDP model as an information-gathering action with an associated cost, we are able to optimally settle this tradeoff between the gain in information arising from the use of communication and the corresponding cost. We illustrate our results in a small test application.

1 Introduction

Consider a group of robots moving in a sparse environment described by a topological map with M nodes. We refer to the nodes in the map as *states*. Each robot must navigate from an initial state to a goal state, known only to that single robot. We admit that several states in the environment have *distinctive landmarks* that the robots can generally perceive through its sensors. However, until a robot is able to reach one such state and observe the corresponding landmark, it must generally navigate through several other states receiving no sensorial feedback from the environment (*e.g.*, using only dead-reckoning).

In this paper we focus only on the problems of global localization and navigation/planning, disregarding other problems such as motion control or obstacle avoidance. We model the navigation task as a sequence of decisions: at each decision instant, each robot must choose from

*Institute for Systems and Robotics, Instituto Superior Tcnico, Lisboa, Portugal. E-mail: [fmelo,mir]@isr.ist.utl.pt. This work was partially supported by POS_C that includes FEDER funds. The first author acknowledges the PhD grant SFRH/BD/3074/2000.

a set of action primitives that control its movements. The robots are allowed to communicate with each other and *share* sensorial information. This received sensorial data can then be used by the robot to improve its localization in the environment.

However, if communication can be used to improve the sensorial capabilities of each robot, it is also true that the communication process generally takes time and consumes resources. Furthermore, it may happen occasionally that no useful information is received. Therefore, it is important to realize in which situations is communication advantageous and in which situations it should be avoided. For example, it may happen that the cost of communication is much higher than the benefit obtained from it. To settle this problem, we make use of a *partially observable Markov decision process* (POMDP) to model each robot in the environment. POMDPs have successfully been used for topological navigation [5–7] and are particularly amenable to the use of Markov localization methods [4]. Furthermore, POMDPs explicitly consider the tradeoff between choosing actions to *disambiguate the state of the robot* (information-gathering) and actions to *move the robot towards the goal*. By explicitly including *communication* in the POMDP model as an information-gathering action with an associated cost, we are able to optimally settle this tradeoff between the gain in information arising from the use of communication and the corresponding cost. This appealing feature of the POMDP framework has led some researchers to address *active sensing* using POMDP models [8].

The paper is organized as follows. In Section 2 we introduce the general POMDP framework. In Section 3 we apply this framework to model the particular problems addressed in the paper. We introduce a simple illustrative example that is used throughout the paper to illustrate the main ideas. Finally, Section 4 concludes the paper with a summary of the main conclusions and points out several possible directions for future work.

2 Partially observable Markov decision processes

A POMDP is a tuple $(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$, where $\mathcal{X}$ is the finite set of possible states of the system, $\mathcal{A}$ is a finite set of control primitives and $\mathcal{Z}$ corresponds to a finite set of possible observations. At each time instant t, a decision-maker chooses an action A_t depending on the past history of events, causing the system to move from its current state X_t to state X_{t+1}. We denote by $\mathsf{P}_a(i,j)$ the probability of moving from state i to state j under action a. As soon as the transition occurs, the decision-maker receives an observation Z_{t+1} that depends on the new state of the system. We denote by $\mathsf{O}_a(j,z)$ the probability of $Z_{t+1} = z$ when $X_{t+1} = j$ and $A_t = a$. Also, as soon as the transition occurs, the decision-maker is granted a numerical reward $r(i,a,j)$, verifying $|r(i,a,j)| \leq R_{\max}$. The purpose of the decision-maker is to choose the control sequence $\{A_t\}$ so as to maximize the functional

$$V(b_0, \{A_t\}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(X_t, A_t, X_{t+1}) \mid X_0 \sim b_0\right], \tag{1}$$

where $\gamma < 1$ is a positive discount factor, b_0 is the *initial belief* for the process and $X_0 \sim b_0$ denotes the fact that X_0 is distributed according to b_0. The initial belief b_0 is a probability vector describing the initial distribution of the state of the system.

In this paper, we are interested in using a POMDP model to describe a robot moving in a sparse environment described topologically. Using a POMDP model, the state-space $\mathcal{X}$ corresponds to the set of sites in the environment. The state of the system at time t, X_t, corresponds to the position of the robot in the environment at time t. The control primitives, or *actions*, correspond to the high-level navigation commands that allow the robot to move between sites in the environment and the observations correspond to the sensorial information. With a POMDP model, Markov localization [4] can be implemented in a straightforward way: at each time instant t the robot maintains a *belief-vector* b_t, each component $b_t(i)$ describing the probability of being in a particular state $i \in \mathcal{X}$. This belief-vector is updated componentwise as

$$b_{t+1}(j) = \mathsf{B}_a(b,z)_j = \frac{\sum_{i\in\mathcal{X}} b_t(i)\mathsf{P}_a(i,j)\mathsf{O}_a(j,z)}{\sum_{i,k\in\mathcal{X}} b_t(i)\mathsf{P}_a(i,k)\mathsf{O}_a(k,z)},$$

where $A_t = a$ and $Z_{t+1} = z$ and $\mathsf{B}_a(b,z)_j$ represents the jth component of the vector $\mathsf{B}_a(b,z)$.

The *optimal value function* for a POMDP is defined as $V^*(b) = \max_{\{A_t\}} V(\{A_t\}, b)$ and verifies the following recursive relation

$$V^*(b) = \max_{a\in\mathcal{A}} \sum_{i,j\in\mathcal{X}} b(i)\mathsf{P}_a(i,j)\left[r(i,a,j) + \gamma \sum_{z\in\mathcal{Z}} \mathsf{O}_a(j,z)V^*\big(\mathsf{B}_a(b,z)\big)\right].$$

The value $V^*(b)$ represents the total expected discounted reward received along an optimal trajectory starting from the initial state distribution b. The optimal decision rule can be defined by means of the mapping

$$\pi^*(b) = \arg\max_{a\in\mathcal{A}} \sum_{i,j\in\mathcal{X}} b(i)\mathsf{P}_a(i,j)\left[r(i,a,j) + \gamma \sum_{z\in\mathcal{Z}} \mathsf{O}_a(j,z)V^*\big(\mathsf{B}_a(b,z)\big)\right].$$

is called the *optimal policy* for the POMDP $(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$.

There are numerous methods in the literature to compute the optimal policy for a POMDP (see the survey works [1,2]). In this paper we adopt the incremental prunning (IP) algorithm [3]. Further details can be found in an extended version of this paper.

3 The POMDP model

We are interested in addressing the situation in which a group of robots moves in a sparse environment, each robot trying to reach its goal location. We use a POMDP model to analyze how the robots can benefit from efficiently using communication, by explicitly including *communication* in the POMDP model as an information-gathering action with an associated cost.

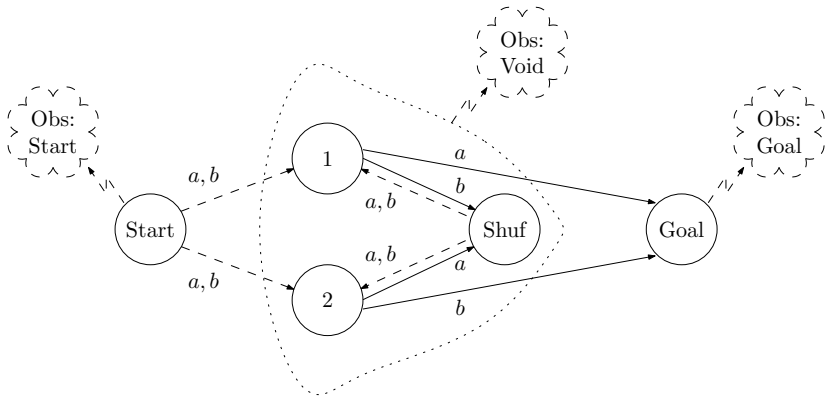


Figure 1: A general sparse environment.

The optimal POMDP policy optimally settle this tradeoff between the gain in information arising from the use of communication and the corresponding cost.

Two important observations are in order. First of all, communication between two robots is modeled as a *directed exchange of sensorial information* and we assume communication to be *peer-to-peer* and not broadcasted.

Secondly, even though we consider the existence of multiple communicating robots in the environment, *we do not address the interaction between these robots*. In particular, the actions of one robot do not affect the behavior of any other robot nor its ability to reach the goal. Therefore, each robot can be modeled independently of the other robots and there is no need to consider multi-agent decision models, such as Dec-POMDPs or stochastic games.

From the discussion above, it should be clear that we can focus our analysis on the behavior of a *single robot*, considering the other robots as part of the environment. We resort to a simplified model that encompasses all the fundamental features of the class of navigation problems considered in the paper. This model is represented in Figure 1.

In this simplified model, a robot departs from the "Start" state by choosing any of the two available actions a or b. It then moves to either state 1 or state 2 with equal probability. The robot will then move to the "Goal" state by choosing a in state 1 or b in state 2 and to the state marked as "Shuf", otherwise. Upon reaching the Goal state, the robot receives a reward of $+20$ and upon reaching the Shuf state, it receives a reward of -5. At the Shuf state, independently of the robot's action, its position is randomly reset to either state 1 or 2, with equal probability. In this model, the robot has 3 available observations: "Start", in the Start state, "Void", in the states inside the dotted line, and "Goal", in the goal state.

In terms of navigation in sparse environments, the set of three undistinguishable states describes those situations in which the robot gets lost due to long periods of dead-reckoning. Upon reaching the Goal state, the robot is back to a location with distinguishing features and can use this information to localize once again. In this simplified model, the robot merely ignores the existence of other robots and just chooses its actions so as to maximize its total

Table 1: Comparison between the performance of the cooperative and the non-cooperative robots.

Test	**Total disc. reward**
Non-cooperative	58.118 ± 22.185
Cooperative	81.498 ± 9.867

reward (reaching the Goal state as quiclly as possible).

We ran IP to compute the optimal policy for the problem above and tested the performance of the obtained policy by running 2000 independent Monte Carlo trials, each consisting of a 10-time step trajectory. We then computed the average total discounted reward. The results are reported in Table 1.

Notice that, after the first action, the robot will get lost between states 1 and 2 and can only "bet" in one of the two possible actions a and b, hoping that it will lead to the desired outcome. However, whatever action the robot chooses, it lead to a successful outcome only 50% of the time. A particularly "lucky" run can bring quite a large reward, while a particularly "unlucky" run can bring an alarmingly large penalty. This justifies the large standard deviation observed in the results portrayed in Table 1. An important aspect of the behavior just described is that it matches the one expected from a robot navigating in a sparse environment: when it gets lost, it keeps moving in a direction where a recognizable state is expectable.

We now describe how the model in Figure 1 is modified to include the existence of another robot in the environment. In particular, and unlike the situation analyzed before, the robot has now two further actions available, dubbed as "Comm 1" and "Comm 2". None of the latter actions affects the position of the robot in the environment. Instead, each action "Comm i", $i = 1, 2$, sends a message "from" state i. This message is sent to a second robot (Robot B) who, upon receiving it, will send the first robot (Robot A) its sensorial information. Notice that such communication actions only suceed in the corresponding states. This means that, if the robot sends a message "Comm i" when at state $j \neq i$, there is a high probability of receiving no reply. Finally, the robot receives a reward of -1 for every communication action.

Two important observations are in order. First of all, notice that there is a cost involved in communication, even if less significant than getting into the "Shuff" state. Secondly, the communication may not succeed. This can happen either because Robot A chose the "wrong" communication action, or simply because Robot B can give Robot A no information on its position (*e.g.*, Robot B cannot "see" Robot A).

We ran IP and computed the optimal policy for this new scenario to compare the performance of the robot with the one observed from the non-cooperative robot. As before, we tested the performance of the optimal policy by running 2000 independent Monte Carlo trials, each consisting of a 10-time step trajectory, and computed the average total discounted

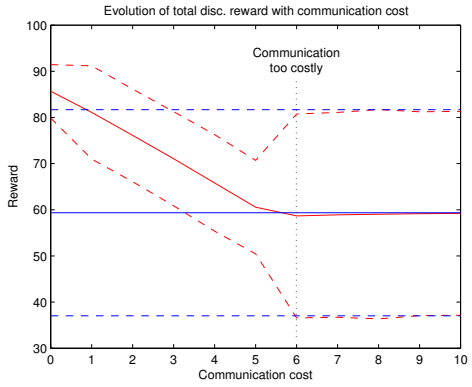


Figure 2: Tradeoff between the cost of communication, and the value of information. Blue line – Non-cooperative robot; Red line – Cooperative robot.

reward. The results are reported in Table 1.

Notice, first of all, the tremendous difference in performance between the two robots. The robot relying on communication exhibited an average increase in performance of about 30%, even receiving the negative rewards arising from communication. Furthermore, the optimal policy in the presence of communication leads to a much more reliable performance, since the observed standard deviation is much smaller.

Finally, to assess the explicit tradeoff between the cost of communication and the "value of information", we have conducted similar tests, varying the communication cost, r_{comm}, between 0 and -10. The corresponding results are summarized in Figure 2, where the solid lines correspond to the mean total discounted reward over 2000 independent Monte-Carlo runs and the dotted line the corresponding standard deviation.

Notice that, as the cost of communication increases, the performance of the cooperative (communicating) robot approaches that of the non-cooperative robot. The two performances reach a similar level when the cost of communication is similar to that of getting lost (*i.e.*, $r_{\text{comm}} = -5$). An important aspect to emphasize is that, when $r_{\text{comm}} = -5$, the performance of the optimal policy in the cooperative case is much more *reliable* than that of the non-cooperative case. This is easily seen from the standard deviation observed. Therefore, even at a high cost, the robot does rely on communication to navigate and this actually translates in an actual improvement in terms of performance. Finally, for $r_{\text{comm}} \geq 6$, communication becomes too costly, as indicated in Figure 2. This means that the optimal policy in both the cooperative and non-cooperative case are similar, as seen from the performance observed in Figure 2.

4 Conclusions and future work

In this paper we addressed the problem of cooperative localization and navigation in sparse environments. We showed that, even if it is possible for a robot moving in such an environment to reach its goal without ever considering the existence of other robots in the environment, communication can greatly improve its overall performance. We made use of a POMDP model to explicitly consider the tradeoff between choosing actions to disambiguate the state of the robot (information-gathering) and actions to move the robot towards the goal. By explicitly including communication in the POMDP model as an information-gathering action with an associated cost, we were able to optimally settle this tradeoff between the gain in information arising from the use of communication and the corresponding cost.

Our results suggest several interesting avenues for future research. First of all, decision-theoretic models as the one described in this paper can be further explored to analyze situations in which communication is used to combine the sensing information from different sources, so as to optimize the total information obtained from the sensorial data. Secondly, multi-agent variations of the model used here can also be used to address *active sensor networks*, by casting such networks as communicating, cooperative multi-agent systems.

References

[1] D. Aberdeen. A (revised) survey of approximate methods for solving partially observable Markov decision processes. Technical report, National ICT Australia, 2003.

[2] A. Cassandra. Optimal policies for partially observable Markov decision processes. Technical Report CS-94-14, Dep. Computer Sciences, Brown University, 1994.

[3] A. Cassandra, M. Littman, and N. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. *UAI'97*, pages 54–61, 1997.

[4] D. Fox. *Markov localization: A probabilistic framework for mobile robot localization and navigation*. PhD thesis, University of Bonn, 1998.

[5] F. Melo and I. Ribeiro. Transition entropy in partially observable Markov decision processes. *IAS-9*, pages 282–289, 2005.

[6] N. Roy and S. Thrun. Coastal navigation with mobile robot. *NIPS 13*, pages 1043–1049, 1999.

[7] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. *IJCAI'95*, pages 1080–1087, 1995.

[8] S. Whitehead and D. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83, 1991.

Task-space and null-space control design for robotic-assisted minimally invasive surgery

Rui Cortesão* Walid Zarrad† Philippe Poignet† Olivier Company†
Etienne Dombre†

Abstract

This paper discusses the control design of robotic-assisted minimally invasive surgery (MIS) with haptic feedback. The operational space control has a position-position telemanipulation architecture with the phantom in the loop, enabling telepresence in free-space and contact. The null space control guarantees that surgical kinematic constraints are fulfilled. Both task and posture control run active observers (AOBs) in Cartesian domain, taking into account force, velocity and position signals. Experiments with a D2M2 (direct drive medical manipulator) robot are presented[1].

Keywords: Medical robotics, haptics, active observers, task space, null space.

1 Introduction

Robotic assisted surgery can greatly improve surgeons' skills. Dedicated instruments that enter in the human body through tiny holes can be tele-controlled through robotic systems, enhancing perception and execution of surgery tasks. Haptic feedback provides realism to contact interactions, distinguishing not only different objects, but also free-space to contact (and vice-versa) transitions. Scaling on-line tactile feedback can magnify or reduce haptic telepresence whenever needed, which is a key functionality for advanced surgery. Guaranteed stability in contact with stiff objects is an important milestone (e.g. touching another instrument or a rib for cardiothoracic surgery) which demands advanced control techniques, being a major obstacle for practical applications. Nowadays, robotized MIS does not include force feedback in the main control loop to avoid instability problems. The Da Vinci^{TM} system

*University of Coimbra, Institute of Systems and Robotics, 3030 Coimbra, Portugal. E-mail: cortesao@isr.uc.pt

†LIRMM-UMR CNRS, University of Montpellier II, F-34392 Montpellier cedex 5, France. E-mails: {zarrad, poignet, company, dombre}@lirmm.fr.

[1]This paper has been published in part in [6].

from Intuitive Surgical is the state-of-the-art robotic system for MIS, in which surgeon hand movements are scaled, filtered and sent to the robotic end-effector. This setup has been used in many MIS, such as cardiac, urology and other abdomen surgeries, without haptic feedback. One main feature is a mechanically created fixed point that coincides with the penetration point. General purpose robotic manipulators can add this feature through proper control design, resorting from geometric models or null space functions. Although the problems of backlash and friction are highly reduced for direct-drive robots, nonlinear coupling among links are significant and the motor dynamics is complex. Therefore, good dynamic models and robust control techniques ought to be applied to achieve high performances.

The advantages over traditional (i.e., non-robotized) surgical practices include:

1. Augmented/scaled reality (e.g. motion and force augmentation or scaling for micro-surgery);
2. Better comfort for the surgeon;
3. Real-time integration of intra-operative data (e.g. image-guided motion and force-controlled motion);
4. Accurate path following;
5. Enhanced mobility. Extra degrees of freedom inside the body can be controlled by the surgeon;
6. Compensation of physiological motions and surgical constraints;
7. Compensation for surgeon's hand tremor;
8. Less pain and trauma and shorter recovery time;
9. Tele-surgery;
10. Training, learning and mentoring using virtual models;

This paper proposes a systematic control design for robotized MIS, based on operational and null space techniques [9] linked to AOBs [4], [5]. It extends the work of [10] through haptic feedback, eliminating gradient-based functions from the control design.

Section 2 introduces the experimental setup and the D2M2 kinematic and dynamic models. Task space control is addressed in Section 3, focusing on operational space techniques, feedback linearization and AOB design. Null space control is discussed in Section 4, considering specific constraints of MIS. Experimental results are presented in Section 5. The conclusions are summarized in Section 6.

Figure 1: Experimental setup. D2M2 robot tele-controlled by the Phantom for robotic surgery. The medical instrument has 40 [cm] height.

2 Experimental setup

The master station has a Phantom 1.5 haptic device with six degrees of freedom (DOF) for position and force. The slave robot is the D2M2 robot designed for beating heart surgery experiments. It has five DOF with direct drive technology providing fast dynamics and low friction. An ATI force sensor is attached to the end-effector. The D2M2 is connected to a Pentium III at 500 [MHz] running under RTX/Windows 2000. The closed loop sampling time $h = 0.7$ [ms], and the system time delay

$$T_d = 5h \tag{1}$$

was obtained experimentally. Master and slave stations are connected via UDP communication under Windows XP. A picture of the experimental setup is represented in Fig. 1.

2.1 D2M2 kinematic and dynamic models

The kinematic structure of the D2M2 is depicted in Fig. 2. The first joint is prismatic and the others are revolute with scara-like disposition. To optimize the overall weight and to boost the dynamic behavior, the motor for the third joint was placed on the prismatic axis, affecting the D2M2 models. Introducing the virtual parallelogram represented in Fig. 2 and also the active, passive and cut joint positions (q, q_p and q_c, respectively), the dynamic model

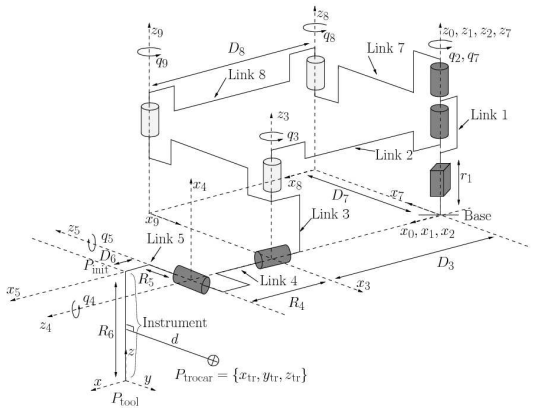


Figure 2: D2M2 Kinematic model. Active, passive and cut joints.

which has contributions from active and passive joints is of form [8]:

$$M\ddot{q} + v(q, \dot{q}) + g(q) = \tau. \tag{2}$$

M is the mass matrix, $v(q, \dot{q})$ is the vector of Coriolis and centripetal forces, $g(q)$ is the gravity term and τ is the generalized torque acting on q. For the D2M2 robot, $g(q)$ is mechanically compensated, being not used in computational design. Further details, including numerical data, can be seen in [10]. In the experiments, the Symoro$^{\text{TM}}$ software has been used to compute the kinematic and dynamic models.

3 Task space control

A picture of the teleoperation scheme for the task space is represented in Fig. 3. The task is described in the base frame by 3D (three dimensional) Cartesian position/force vectors, associated to the D2M2 end-effector. The system plant $G(s)$ embeds operational space and feedback linearization techniques. The phantom position x_p scaled by β_p is compared with the end-effector position[2] X_t, generating a desired force $f_{i,t}$ through the virtual coupling K_v. To enhance the haptic feeling quality of stiff objects, K_v may increase while in contact. A constant K_v establishes the trade-off between telepresence in contact, and robustness and comfortable performance in free space [5]. y_k is computed from force sensor measures projected into the base frame. It tracks the reference $f_{i,t}$ with desired dynamics through the AOB, even if the system stiffness K_s changes. $K_{s,n}$ is the value of K_s used in the control design, being estimated on-line from force data [5]. The human arm perceives $f_{i,t}$ scaled by β_f, which anticipates the real force. Telepresence is achieved if y_k follows well $f_{i,t}$. The main advantage of such position-position control scheme is that the human can feel position errors in free-space, due to motion,

[2]The subscript "t" is used for task space variables, whenever appropriate.

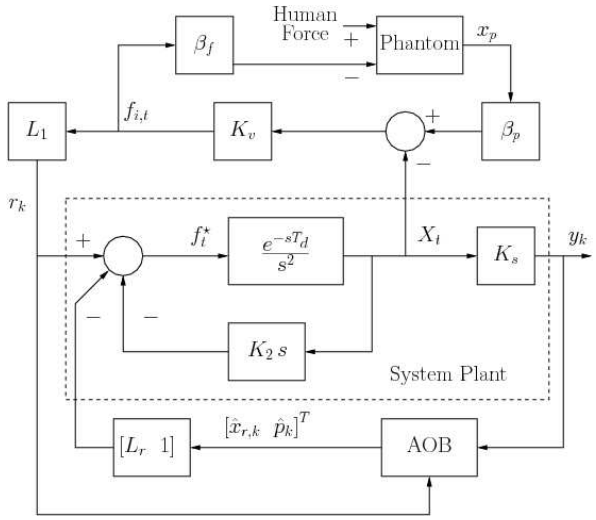


Figure 3: Task space control scheme with AOBs for each Cartesian dimension. The master station, which includes the human and phantom, generates the 3D Cartesian force $f_{i,t}$ through the virtual coupling K_v. β_p scales the phantom position x_p, and β_f scales back $f_{i,t}$ to the master station. $G(s)$ has a damping term K_2, and is controlled by AOB estimates $[\hat{x}_{r,k}\ \ \hat{p}_k]^T$ through the state feedback gain $[L_r\ 1]$. K_s is the system stiffness and L_1 is the first element of L_r.

robot collision (not necessarily at the end-effector), or human-robot interaction. The main disadvantage is that impact events (high-frequency) at the end-effector are not accurately felt by the human (in this case, feeding back y_k would give better results). The AOB performs control actions only based on force signals (i.e., r_k and y_k).

3.1 Task space dynamics

Equation (2) for the operational space is

$$\Lambda_t \ddot{X}_t + V_t(q, \dot{q}) + g_t(q) = F_t, \tag{3}$$

with

$$\dot{X}_t = J_t \dot{q}, \tag{4}$$

$$\Lambda_t = (J_t^+)^T M J_t^+, \tag{5}$$

$$V_t(q, \dot{q}) = (J_t^+)^T v(q, \dot{q}) - \Lambda_t \dot{J}_t \dot{q}, \tag{6}$$

$$\tau_t = J_t^T F_t \tag{7}$$

and

$$g_t(q) = (J_t^+)^T g(q). \tag{8}$$

J_t, F_t and τ_t are respectively the Jacobian matrix, Cartesian force and task torque. J_t is a truncated non-squared matrix (3×5), therefore, J_t^+ represents its pseudo-inverse. In the experiments, the dynamically consistent pseudo-inverse has been used, i.e.

$$J_t^+ = M^{-1} J_t^T (J_t M^{-1} J_t^T)^{-1}, \tag{9}$$

which minimizes the robot kinetic energy [2], [8]. One advantage of truncated Jacobian solutions for robots that are not intrinsically redundant is the possibility to utilize the null space to optimize another objective function, very useful in robotic-assisted MIS. If external manipulation of the orientation is desired, a full Jacobian can be used, with zero values for the orientation vector (keeping the same task space controller)[3].

3.2 Feedback Linearization

Whenever the robot is in contact, an external force F_e appears at the end-effector. Hence, (3) can be written as

$$\Lambda_t \ddot{X}_t + V_t(q, \dot{q}) + g_t(q) = F_{c,t} + F_e, \tag{10}$$

where $F_{c,t}$ is the commanded force. For the desired Cartesian-decoupled system plant

$$\ddot{X}_t = f_t^\star, \tag{11}$$

$F_{c,t}$ should be[4]

$$F_{c,t} = -\hat{F}_e + \hat{V}_t(q, \dot{q}) + \hat{g}_t(q) + \hat{\Lambda}_t f_t^\star. \tag{12}$$

The estimation of F_e, $\hat{F}_e$, affects the control strategy [5]. The terms $\hat{V}_t(q, \dot{q})$, $\hat{g}_t(q)$ and $\hat{\Lambda}_t$ can be computed for a given robot. Introducing K_2, T_d and $K_{s,n}$, the desired system plant is[5]

$$G(s) = \frac{K_{s,n}\, e^{-sT_d}}{s(s + K_2\, e^{-sT_d})}. \tag{13}$$

Since T_d is small (see (1)),

$$G(s) \approx \frac{K_{s,n}\, e^{-sT_d}}{s(s + K_2)} \tag{14}$$

for a wide range of frequencies. Its equivalent temporal representation is

$$\ddot{y}(t) + K_2 \dot{y}(t) = K_{s,n} u(t - T_d), \tag{15}$$

where $y(t)$ is the plant output (Cartesian force at the robot's end-effector), and u is the plant input (force). Defining the state variables $x_1(t) = y(t)$ and $x_2(t) = \dot{y}(t)$, (15) can be written as

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -K_2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ K_{s,n} \end{bmatrix} u(t - T_\mathrm{d}). \tag{16}$$

[3]This solution is not appropriate for null space methods, though.

[4]The symbol $\hat{}$ means estimate.

[5]The analysis is done for each Cartesian dimension.

In compact form,

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t - T_d) \\ y(t) = x_1(t) \end{cases} . \tag{17}$$

Discretizing (14) with sampling time h [1], the equivalent discrete time system is

$$\begin{cases} x_{r,k} = \Phi_r x_{r,k-1} + \Gamma_r u_{k-1} \\ y_k = C_r x_{r,k} \end{cases} , \tag{18}$$

with

$$T_d = (d-1)h + \tau', \tag{19}$$

$$0 < \tau' \leq h, \tag{20}$$

$$x_{r,k} = \begin{bmatrix} x_k & u_{k-d} & \cdots & u_{k-2} & u_{k-1} \end{bmatrix}^T, \tag{21}$$

$$\Phi_r = \begin{bmatrix} \Phi_1 & \Gamma_1 & \Gamma_0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{22}$$

$$\Gamma_r = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T \tag{23}$$

and

$$C_r = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{24}$$

Φ_1, Γ_0 and Γ_1 are given by

$$\Phi_1 = e^{Ah} = \phi(h), \tag{25}$$

$$\Gamma_0 = \int_0^{h-\tau'} \phi(\lambda)\, d\lambda\, B \tag{26}$$

and

$$\Gamma_1 = \phi(h - \tau') \int_0^{\tau'} \phi(\lambda)\, d\lambda\, B. \tag{27}$$

In our case, x_k has dimension seven. The first two states represent the force and force derivative, respectively. The other five states appear due to T_d. The continuous state transition and command matrices are

$$\phi(t) = \begin{bmatrix} 1 & \frac{1-e^{-K_2 t}}{K_2} \\ 0 & e^{-K_2 t} \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 \\ K_{s,n} \end{bmatrix}. \tag{28}$$

From (28), the computation of Φ_1, Γ_0 and Γ_1 is straightforward.

3.3 AOB design

To accomplish model-reference adaptive control, the AOB reformulates the Kalman filter, based on [4], [5]:

1. A desired closed loop system (reference model) that enters in the state estimation.
2. An extra equation to estimate an equivalent disturbance referred to the system input, due to unmodeled terms including higher order dynamics, parameter mismatches and unknown disturbances. An active state p_k (extra state) describes the equivalent disturbance.
3. The stochastic design of the Kalman matrices for the AOB application. The first-order AOB algorithm is summarized in the sequel.

Controlling (18) through state feedback from an observer, and inserting p_k and $\hat{p}_k$ in the loop, the overall system can be represented by [5]

$$\begin{aligned}\begin{bmatrix} x_{r,k} \\ p_k \end{bmatrix} &= \begin{bmatrix} \Phi_r & \Gamma_r \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{r,k-1} \\ p_{k-1} \end{bmatrix} \\ &\quad + \begin{bmatrix} \Gamma_r \\ 0 \end{bmatrix} u_{k-1} + \xi_k \end{aligned} \tag{29}$$

and

$$y_k = C_a \begin{bmatrix} x_{r,k-1} & p_{k-1} \end{bmatrix}^T + \eta_k, \tag{30}$$

where

$$u_{k-1} = r_{k-1} - \begin{bmatrix} L_r & 1 \end{bmatrix} \begin{bmatrix} x_{r,k-1} \\ \hat{p}_{k-1} \end{bmatrix}. \tag{31}$$

The stochastic inputs $\xi_k = \begin{bmatrix} \xi_{x_{r,k}} & {}^0 w_k \end{bmatrix}^T$ and η_k represent respectively model and measure uncertainties. The state estimate of (29) is

$$\begin{aligned}\begin{bmatrix} \hat{x}_{r,k} \\ \hat{p}_k \end{bmatrix} &= \begin{bmatrix} \Phi_r - \Gamma_r L_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{r,k-1} \\ \hat{p}_{k-1} \end{bmatrix} \\ &\quad + \begin{bmatrix} \Gamma_r \\ 0 \end{bmatrix} r_{k-1} + K_k(y_k - \hat{y}_k), \end{aligned} \tag{32}$$

with

$$\hat{y}_k = C_a \left(\begin{bmatrix} \Phi_r - \Gamma_r L_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{r,k-1} \\ \hat{p}_{k-1} \end{bmatrix} + \begin{bmatrix} \Gamma_r \\ 0 \end{bmatrix} r_{k-1} \right) \tag{33}$$

and

$$C_a = \begin{bmatrix} C_r & 0 \end{bmatrix}. \tag{34}$$

The Kalman gain K_k reflects the uncertainty associated to each state, which is a function of ξ_k and η_k [3], [7]. It is given by

$$K_k = P_{1k}\, C_a^T \left[C_a\, P_{1k}\, C_a^T + R_k\right]^{-1}, \tag{35}$$

with

$$P_{1k} = \Phi_n\, P_{k-1}\, \Phi_n^T + Q_k \tag{36}$$

and

$$P_k = P_{1k} - K_k\, C_a\, P_{1k}. \tag{37}$$

Φ_n is the augmented open loop matrix,

$$\Phi_n = \begin{bmatrix} \Phi_r & \Gamma_r \\ 0 & 1 \end{bmatrix}. \tag{38}$$

The system noise matrix Q_k is

$$Q_k = \begin{bmatrix} Q_{x_{r,k}} & 0 \\ 0 & Q_{p_k} \end{bmatrix}. \tag{39}$$

P_k and R_k are respectively the mean square error and measurement noise matrices.

3.4 Computed torque for the task

From (3), (7), (10), (12) and Fig. 3, the commanded torque for the task $\tau_{\mathrm{c,t}}$ is

$$\begin{aligned} \tau_{c,t} \;=\; & J_t^T \Big\{ -\hat{F}_e + \hat{V}_t(q,\dot{q}) + \hat{g}_t(q) + \\ & \hat{\Lambda}_t \left(r_k - [L_r \;\; 1]\, [\hat{x}_{r,k} \;\; \hat{p}_k]^T - K_2 \dot{X}_t \right) \Big\}. \end{aligned} \tag{40}$$

In the experiments, $\dot{X}_t$ given by (4) was filtered by a discrete first-order low-pass filter. The estimate $\hat{F}_e$ is the first state of $\hat{x}_{r,k}$. The gravity $\hat{g}_t(q) = 0$ (see Section 2.1).

4 Null space control

For MIS, the task robot is teleoperated by a haptic device with force feedback, as explained in Section 3. A secondary task is performed by a virtual null space robot, which attempts to have always the trocar position on the medical instrument. Virtual null space robots[6] can be defined by operational space techniques, short-cutting inverse kinematics problems, enabling the same control architecture of the task space. Although potential fields and associated gradient functions are very popular cost functions, this paper proposes another method to deal with MIS kinematic constraints, enabling posture control from 2D Cartesian positions only.

[6]There is just one real robot that can be described by several virtual robots.

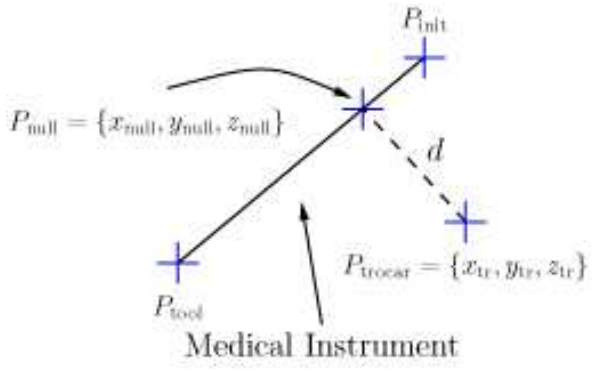


Figure 4: End-effector of task and null space robots (P_{tool} and P_{null}, respectively). Trocar projection onto the medical instrument (P_{null}). P_{init} is the starting point of the medical instrument.

4.1 Null space robot

The difference between the null and task space robots represented in Fig. 4 is only at the tool length. The end-effector of the task space robot is in P_{tool}, with constant tool length. The virtual end-effector of the null space robot is in P_{null}, with time varying tool length, corresponding to the trocar projection onto the medical instrument. Both robots have the same dynamic model (the weight of the medical instrument is small). Only the kinematic models are slightly different. P_{null} is computed from

$$P_{\text{null}} = P_{\text{tool}} + (P_{\text{init}} - P_{\text{tool}})\rho \tag{41}$$

with

$$\rho = -\frac{(P_{\text{tool}} - P_{\text{trocar}}) \cdot (P_{\text{init}} - P_{\text{tool}})}{|P_{\text{init}} - P_{\text{tool}}|^2} \tag{42}$$

where $\cdot$ denotes the dot product and P_{init} is the tool starting point. To assess the null space robot behavior, the shortest distance between the medical instrument and trocar d_{tr} can be computed at each time step.

$$d_{\text{tr}} = \frac{|(P_{\text{init}} - P_{\text{tool}}) \wedge (P_{\text{tool}} - P_{\text{trocar}})|}{|P_{\text{init}} - P_{\text{tool}}|}, \tag{43}$$

where $\wedge$ denotes the cross product.

4.2 Control design

The AOB control architecture for the null space is represented in Fig. 5. The robot is only position controlled in x_{null} and y_{null} (2D Cartesian coordinates), with a constant reference corresponding to the trocar position $\{x_{\text{tr}}, y_{\text{tr}}\}$. The virtual robot is synthesized by feedback

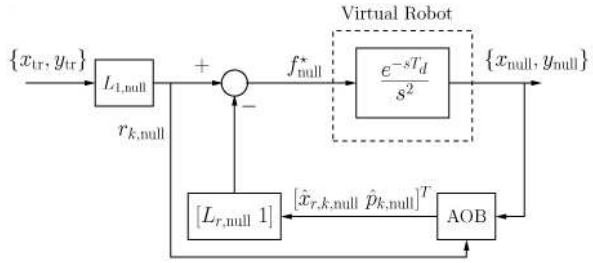


Figure 5: Null space control scheme with AOBs for each Cartesian dimension. Position control architecture. $L_{1,\rm null}$ is the first element of the state feedback gain $L_{\rm null} = [L_{r,\rm null}\ 1]$, The trocar position is the input and the trocar projection onto the medical instrument is the output.

linearization techniques in the null Cartesian space[7]. The same procedure of (2)-(12) has been applied with minimal changes. The null Jacobian matrix $J_{\rm null}$ is non-squared (2×5), and takes into account the value of $P_{\rm null}$. The system plant for each null Cartesian position is then

$$G_{\rm null}(s) = \frac{e^{-sT_d}}{s^2}. \tag{44}$$

In the time domain[8]

$$\ddot{x}_{\rm null} = f^{\star}_{\rm null}(t - T_d). \tag{45}$$

Defining $x_{1,\rm null}(t) = x_{\rm null}$ and $x_{2,\rm null}(t) = \dot{x}_{\rm null}(t)$, (45) can be written as

$$\begin{bmatrix} \dot{x}_{1,\rm null}(t) \\ \dot{x}_{2,\rm null}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_{1,\rm null}(t) \\ x_{2,\rm null}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f^{\star}_{\rm null}(t - T_{\rm d}), \tag{46}$$

The state space description of (46) has the same form of (16). Therefore, the subsequent analysis (including the AOB design) done for the task space control applies to null space as well. The computed torque for the null space robot $\tau_{c,\rm null}$ is

$$\begin{aligned} \tau_{c,\rm null} \ = \ & J^T_{\rm null} \Big\{ -\hat{F}_{e,\rm null} + \hat{V}_{t,\rm null}(q, \dot{q}) + \hat{g}_{t,\rm null}(q) - F_{i,\rm null} \\ & + \hat{\Lambda}_{\rm null} \left(r_{k,\rm null} - [L_{r,\rm null}\ 1]\, [\hat{x}_{r,k,\rm null}\ \hat{p}_{k,\rm null}]^T \right) \Big\}, \end{aligned} \tag{47}$$

where $\hat{F}_{e,\rm null}$, $\hat{V}_{t,\rm null}(q, \dot{q})$ and $\hat{g}_{t,\rm null}(q)$ are the 2D truncated versions of $\hat{F}_e$, $\hat{V}_t(q, \dot{q})$ and $\hat{g}_t(q)$, respectively. The induced Cartesian force on the null space robot due to the task $F_{i,\rm null}$ is

$$F_{i,\rm null} = \hat{\Lambda}_t f^{\star}_{t,\rm null}, \tag{48}$$

[7]The expressions of form "null *" mean "* for the null space robot".

[8]The analysis is done for $x_{\rm null}$, which is the same for $y_{\rm null}$.

Table 1: AOB design parameters for each Cartesian dimension of both task and null space robots. The uncertainty associated to each state is $Q_{i,i}$ with $i = 1, \cdots, 8$. The units of $Q_{i,i}$ and R_k are state dependent.

	K_2	$K_{s,n}$	τ_d	$Q_{1,1}$	$Q_{i,i}$	$Q_{8,8}$	R_k
units	[Ns/m]	[N/m]	[s]				
x_t	5	300	0.05	10^{-9}	10^{-12}	10^{-3}	100
y_t	5	300	0.05	10^{-9}	10^{-12}	10^{-3}	100
z_t	5	300	0.05	10^{-9}	10^{-12}	10^{-3}	100
x_{null}			0.0035	0.7	10^{-12}	3500	100
y_{null}			0.0035	0.7	10^{-12}	3500	100

corresponding to the 2D truncated value of $\hat{\Lambda}_t f_t^\star$. This force is compensated in (47). Projecting (47) in the task null space and adding (40), the total commanded torque τ_c is

$$\tau_c = \tau_{c,t} + (I - J_t^T (J_t^+)^T)\tau_{c,\text{null}}, \tag{49}$$

where I is the identity matrix [9].

5 Experiments

This section reports tracking capabilities of the overall system subject to trocar constraints. It should be pointed out that at this design stage no real trocar has been inserted into the system. A trocar point was defined in space, along which the tele-controlled task was carried out. For the engineering design, system and AOB parameters have to be specified. In the experiments, $K_v = 1000$ [N/m], $\beta_p = 0.7$ and $\beta_f = 0.5$. L_r and $L_{r,\text{null}}$ have been computed by Ackermann's formula to achieve a critically damped response (for force and position, respectively) with desired time constant τ_d. The five additional poles (see Section 3.2) were mapped at the origin (z-domain). Table 1 summarizes the AOB design. $K_{s,n}$ is kept low for free space, allowing fast motions. For contact tasks, the closed loop bandwidth is small, which is adequate for human manipulation. A higher null space bandwidth is required to avoid embarrassing values of d_{tr}. The stochastic estimation structure is more sensor based for the null space ($Q_{1,1}$ and $Q_{8,8}$ are higher), reacting faster to position errors. Both schemes follow model-reference adaptive control strategies, since the uncertainty associated with p_k ($Q_{8,8}$ value) is much higher than for the other states. The absolute values of Q_k and R_k are irrelevant for the AOB Kalman gain. Only relative relations are important [4].

5.1 Experimental results

Figures 6 and 7 present tracking capabilites in free space with haptic feedback. Position errors are felt by the user, increasing motion perception. 3D tele-manipulation is well followed by the robot (Fig. 6), satisfying MIS kinematic constraints (Fig. 7). The root mean square errors (γ) in [mm] for the task are

$$\gamma(x_t) = 0.87, \;\; \gamma(y_t) = 0.79 \;\; \text{and} \;\; \gamma(z_t) = 0.49, \tag{50}$$

and the mean value of d_{tr} is 2.62 [mm].

6 Conclusions

This paper has presented a haptic control architecture for robotic-assisted MIS. The task space is tele-controlled by a human operator through a haptic device. A position-position teleoperation architecture has been used, where 3D Cartesian errors generate a desired force through virtual coupling. Tracking errors below 1 [mm] have been achieved, while satisfying MIS constraints. A virtual null space robot has been introduced, enabling posture control to satisfy MIS kinematic constraints. 2D Cartesian position commands have been used for the posture, without resorting to gradient-based functions, inverse kinematics or explicit orientation commands. Induced Cartesian forces on the null space due to the task have been compensated. Null space position errors are below 3 [mm]. Both controllers apply AOBs, which run on top of operational space and feedback linearization techniques. Discrete state space methods, augmented states and stochastic state estimation belong to AOB design. For the engineering point of view, the control design is straightforward and the stochastic parameters provide enough flexibility to add dynamic functionalities without affecting control gains. Experimental results have shown good performance in free space motion subject to trocar constraints.

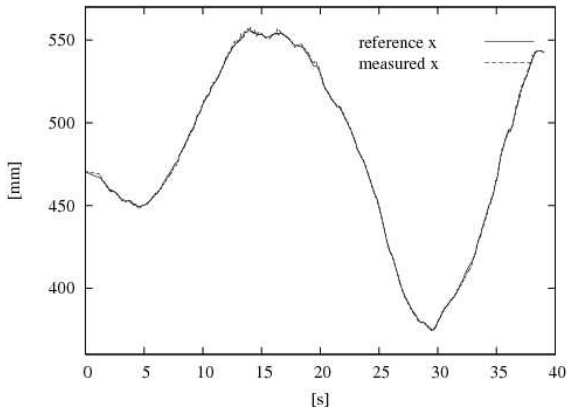
References

[1] K. J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design.* Prentice Hall, 1997.

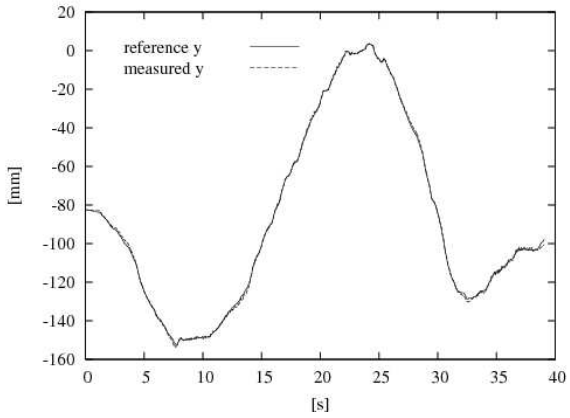
[2] M. Benoit, M. Briot, H. Donnarel and. A. Liégeois, M. Meyer, and M. Renaud. Synthèse de la comande dynamique d'un téléopérateur redondant. *RAIRO*, pages 89–103, 1975.

[3] S. M. Bozic. *Digital and Kalman Filtering.* Edward Arnold, London, 1979.

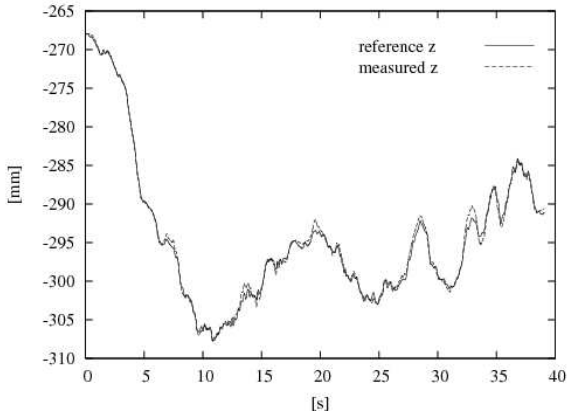
[4] R. Cortesão. *Kalman Techniques for Intelligent Control Systems: Theory and Robotic Experiments.* PhD thesis, University of Coimbra, 2003.



(a)



(b)



(c)

Figure 6: Position tracking performance of the task space robot. 3D Cartesian motions tele-controlled by the phantom, satisfying trocar constraints.

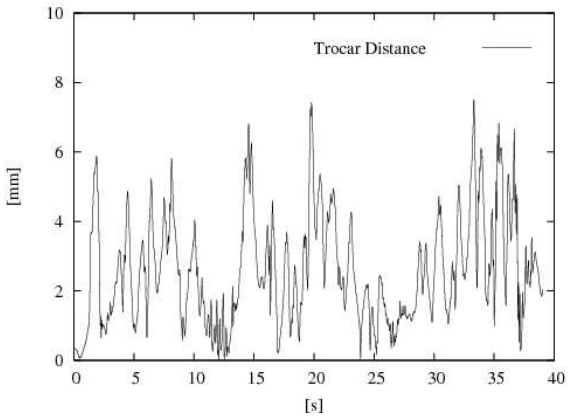


Figure 7: Position tracking performance of the null space robot. Shortest distance between the medical instrument and trocar.

[5] R. Cortesão, J. Park, and O. Khatib. Real-time adaptive control for haptic telemanipulation with kalman active observers. *IEEE Trans. on Robotics*, 22(5):987–999, October 2006.

[6] R. Cortesão, W. Zarrad, P. Poignet, O. Company, and E. Dombre. Haptic control design for robotic-assisted minimally invasive surgery. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 454–459, China, 2006.

[7] A. Jazwinsky. *Stochastic Processes and Filtering Theory*, volume 64 of *Mathematics In Science and Engineering*. Academic Press, 1970. (Edited by R. Bellman).

[8] W. Khalil and E. Dombre. *Modeling, Identification and Control of Robots*. Hermes Penton Ltd, 2002.

[9] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Int. J. on Robotics and Automation*, 3(1):43–53, February 1987.

[10] M. Michelin, P. Poignet, and E. Dombre. Dynamic task / posture decoupling for minimally invasive surgery motions. In *International Symposium on Experimental Robotics (ISER)*, Singapore, 2004.

A curious robot: an explorative-exploitive inference algorithm

Kim Steenstrup Pedersen* Peter Johansen*

Abstract

We propose a sequential learning algorithm with a focus on robot control. It is initialised by a teacher who directs the robot through a series of example solutions of a problem. Left alone, the control chooses its next action by prediction based on a variable order Markov chain model selected to minimise a MDL criterion based on generalised code length L_α of the past robot-environment interaction. The user specifies the parameter α and as a result, the robot can be directed towards exploratory behaviour if confidence in the teacher is low ($\alpha < 0$), and towards goal-seeking exploitive behaviour if confidence in the teacher is high ($\alpha > 0$). The novelty of the proposed method lies in the use of generalised code length in the MDL model selection criterion.

1 Introduction

The control for a robot can be explicitly programmed for each task. However it would be preferable that the robot possesses a general-purpose program that allows it to learn from its experience. This way an explicit specification of the problem to be solved would not be needed or rather, the specification can be made more and more explicit by showing the robot additional examples of its intended behaviour. Behaviour by generalisation from example must be guided by some principle. By choosing the principle appropriately one can achieve either exploitive or explorative behaviour. In this contribution we characterise behaviour in terms of model selection based on minimisation of the Rényi entropy [6]. The selection uses the minimum description length (MDL) principle which is used in data compression and model selection problems [7]. We model the interaction with the environment by considering the pair of action, e.g. motor control, taken by the robot and the reaction from the environment, e.g. input received on the robot's sensors. Each robot action is a question to the environment, and the answer is the next input to the robot. Viewed this way, we group the sequence of alternating outputs and inputs into compound symbols - into action-input pairs. Based on the robot's previous experience - its sequence of compound action-input pairs - it

*Dept. of Computer Science, University of Copenhagen, Denmark. E-mail:kimstp@diku.dk.

makes a prediction of the next compound symbol using the control algorithm. The output component of the predicted compound symbol becomes its action, and the input component is the expected next input. The next compound symbol in the case that prediction of the next input fails, indicating insufficient teaching, is to combine the predicted action (which has already been executed) with the actual (but unexpected) input. In an initial learning phase the teacher guides the robot through repeated examples of problem solving, for instance by remote control. When teaching is over, the robot determines its next moves by the inferred model. We make the assumption about the robot-environment interaction, that it is a stationary and ergodic random process. And we model the interaction as a variable order Markov chain with finite discrete state space. We use the 2-pass Context algorithm [5] for minimisation of the proposed MDL criterion.

Using variable order Markov chain models for prediction of future behaviour has previously been proposed by among others Ron et al. [9]. Csiszar et al. [3] and Bühlmann [1] have previously studied the Context algorithm for model selection and have investigated different optimality criteria, also studying the effect and selection of inherent construction parameters. The novelty in our work is that we apply generalised code length with a meta parameter in order to obtain a continuum of behaviour.

2 Rényi's entropy and generalised code length

We represent the robot-environment interaction over time as a discrete time sequence $\mathbf{x}^n = x_1 x_2 \cdots x_n$ of length n, where each symbol $x_t \in \Sigma$ represents a particular action-input pair. We assume that both actions and input are discrete, and the alphabet Σ representing all possible action-input pairs is finite $|\Sigma| = M$. The set of all suffixes of the sequence $\mathbf{x}^n \in \Sigma^n$ is defined as $\text{Suffix}(\mathbf{x}^n) = \{x_i \cdots x_n | 1 \leq i \leq n\} \cup \{\lambda\}$, where Σ^n denotes the set of sequences of length n and λ denotes the empty sequence. The elements of this set are called *suffixes* of $\mathbf{x}^n$. A suffix is called *proper* if it is not equal to $\mathbf{x}^n$. We will use the terms suffix and context interchangeably. The set of all prefixes of $\mathbf{x}^n \in \Sigma^n$ is defined as $\text{Prefix}(\mathbf{x}^n) = \{x_1 \cdots x_i | 1 \leq i \leq n\} \cup \{\lambda\}$ and the elements of this set are called *prefixes* to $\mathbf{x}^n$.

The interaction may be subject to noise, e.g. noisy motor control (action) and noisy sensory measurements (input). We will therefore consider the robot-environment interaction as a discrete time discrete state random process $\{X_t\}_{t \geq 1}$, where the random variables X_t takes values from Σ. A specific sequence $\mathbf{x}^n$ may be thought of as a particular realisation of the random process from time $t = 1$ to $t = n$. Since Σ is discrete, we may write the probability of the i'th symbol, $i \in \Sigma$, as $p_i \equiv P(X = i)$. We will furthermore assume that the past robot-environment interaction is a homogeneous ergodic (irreducible positive recurrent) variable order Markov chain. Hence learning a task reduces to estimating a Markov chain model. Without the last assumption we cannot expect to learn a model from a single realisation of

$\mathbf{x}^n$.

A Markov chain is defined by a set of states $\mathcal{S}$, the probability distribution of the initial state and probabilities on the allowed transitions. The probability of the transition from state $\mathbf{s}_i$ to $\mathbf{s}_j$ is denoted by p_{ij} and has the property that $\sum_j p_{ij} = 1$. In case the conditional probability $P(\mathbf{s}_j|\mathbf{s}_i)$ is defined, whenever $P(\mathbf{s}_i) \neq 0$, we have $p_{ij} = P(\mathbf{s}_j|\mathbf{s}_i)$. We identify the states of a variable order Markov chain with suffixes, such that $\mathbf{s}_j \in \Sigma^{k_j}$ denotes the j'th state of order k_j. A transition between two states $\mathbf{s}, \mathbf{w} \in \mathcal{S}$, $|\mathbf{s}| = k_1$ and $|\mathbf{w}| = k_2$, is only possible, if a proper suffix $\mathbf{v} \in \text{Suffix}(\mathbf{s})$ exist such that $\mathbf{v}$ is a prefix of $\mathbf{w}$, i.e. $\mathbf{v} \in \text{Prefix}(\mathbf{w})$. Let $\tau_{\mathbf{s}} = \{\mathbf{w}|\mathbf{v} \in \text{Suffix}(\mathbf{s}) \ \mathbf{s} \wedge \mathbf{v} \in \text{Prefix}(\mathbf{w})\}$ denote the set of possible states to which we can transition from the state $\mathbf{s}$.

Let $X \in \Sigma$, $|\Sigma| = M$, be a discrete random variable with probability distribution $p_1, \ldots, p_M$ which we want to encode using some finite code alphabet Λ, $|\Lambda| = D$. We will in this paper assume without loss of generality that the coding alphabet $\Lambda = \Sigma$ and Σ to be binary and let log denote the base 2 logarithm. Rényi [6] introduces a generalisation of Shannon entropy [10] known as the Rényi or generalised entropy of order α defined as,

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^{M} p_i^\alpha \right) \ , \ \alpha > 0 \ , \ \alpha \neq 1 \ . \tag{1}$$

The Rényi's entropy $H_\alpha(X)$ is a generalisation of the Shannon entropy $H(X) = -\sum_i p_i \log p_i$, because $H_\alpha(X) \to H(X)$ as $\alpha \to 1$.

Campbell [2] introduces a generalised coding theorem concerning Rényi's entropy. The theorem is stated in terms of the generalised average $L(t) = \phi^{-1}\left(\sum_i p_i \phi(l_i)\right)$ of the code lengths l_i of symbol $i \in \Sigma$ with $\phi(x) = 2^{tl_i}$ and $\phi^{-1}(x) = \frac{1}{t}\log(l_i)$, where $0 < t < \infty$ and $\alpha = 1/(t+1)$. The theorem states that for $\alpha = 1/(t+1)$ it is possible to make the expected code length $L(t)$ of order t as close as desired to H_α by encoding sufficiently long sequences and $H_\alpha \leq L(t)$ for all choices of code books. Campbell [2] shows that in order for $L(t) = H_\alpha$, the code length of the symbol $i \in \Sigma$ must be

$$l_\alpha(i) = -\log \left(\frac{p_i^\alpha}{\sum_{j=1}^{M} p_j^\alpha} \right) = -\alpha \log p_i + \log(\sum_{j=1}^{M} p_j^\alpha) \ . \tag{2}$$

We will call $l_\alpha(i)$ the generalised code length of order α. For $\alpha = 1$, $l_\alpha(i)$ reduces to the Shannon code length $l(i) = -\log p_i$. We can interpret (2) as stating that generalised coding arises from using the Shannon code length based on the modified distribution $p_i^\alpha / \sum_{j=1}^{M} p_j^\alpha$. By using the modified distribution we may put different weight to the different symbols dependent on our choice of α, e.g. we may put a limit to the maximum code length we want to consider.

3 Robot control by generalised MDL

We will construct a variable order Markov chain model of the past robot-environment interaction and base the prediction of future interaction on this model. To select this model we use the 2-pass Context algorithm proposed by Nohre [5], a 2-pass variant of the algorithm Context by Rissanen [7]. The model selection is based on the MDL principle [8] in which a model is selected from a class of models such that it minimises the combined length of the description of both the data as well as the model itself. Instead of using the standard definition of MDL as in [5, 7], we propose to use a definition based on generalised code length.

The Context algorithm represent the past interaction sequence $\mathbf{x}^n$ by a suffix tree representation of the suffix set $\text{Suffix}(\mathbf{x}^n)$. Every node in the suffix tree correspond to a suffix $\mathbf{s}$ of the past $\mathbf{x}^n$. The root node represent the empty sequence λ. The suffix $\mathbf{s} = s_1 \cdots s_k \in \text{Suffix}(\mathbf{x}^n)$ is represented as the node reached by traversing the tree starting at the root and moving to the child node representing s_1 and so forth down until s_k is reached. Hence, the sequence $\mathbf{s}$ points to a node at depth k in the suffix tree of $\mathbf{x}^n$. The Context algorithm selects a model of the past by pruning the tree, thereby selecting a subset of the suffix set $\mathcal{S} \subseteq \text{Suffix}(\mathbf{x}^n)$ which forms the state set $\mathcal{S}$ of the Markov model. In the following, we distinguish between the suffix tree and the model tree. The *suffix tree* is the representation of the robot-environment interaction up to now and the *model tree* is the result of the model selection. The leaves of the model tree form the model state set $\mathcal{S}$.

As part of building and selecting the model we need to estimate state transition probabilities. This is done on a per node basis and we assume that the suffix tree is complete and use the Laplace probability estimator to handle the case of unseen events. If the sequence was generated by a variable order Markov chain with state set $\mathcal{S}$, the probability estimates in nodes corresponding to states $\mathbf{s}_i \in \mathcal{S}$ will correspond to estimates of the transition probabilities $p_{ij} = P(\mathbf{s}_j|\mathbf{s}_i)$ of the Markov source, where $\mathbf{s}_j \in \mathcal{S} \cap \tau_{\mathbf{s}_i}$.

In the Context algorithm [5, 7], the code length of a context, i.e. a node in the suffix tree, is computed based on predictive coding of the memory-less continuation after the context. Form a sequence $\mathbf{x}_\mathbf{s}$ of the symbols following immediately after every occurrence of the node sequence $\mathbf{s}$ in $\mathbf{x}^n$. E.g. let $\mathbf{x}^n = 10100101$ and $\mathbf{s} = 10$, then $\mathbf{x}_\mathbf{s} = 101$. It is clear that $\mathbf{x}_\mathbf{s}$ represents all possible continuations after the sequence $\mathbf{s}$. This allows us to compute the probability estimates of p_{ij} using the Laplace estimator and by collecting the frequency statistics of the symbols in $\mathbf{x}_{s_i}$. Assuming $\mathbf{x}^n$ was generated by a Markov source, the symbols in $\mathbf{x}_\mathbf{s}$ are samples from conditional independent random variables given the context $\mathbf{s}$.

The generalised code length of the continuation $\mathbf{x}_\mathbf{s}$ of $\mathbf{s}$ can be derived from (2) to be

$$l_\alpha(\mathbf{x}_\mathbf{s}) = -\alpha \log(\nu_0!\nu_1!) + \sum_{i=1}^{\nu_1} \log(i^\alpha + 1) + \sum_{j=1}^{\nu_0} \log((\nu_1 + 1)^\alpha + j^\alpha) \;, \tag{3}$$

where ν_i denotes the frequency of occurrence of the ith symbol. When $\alpha = 1$, this expression

reduces to the predictive code length of $\mathbf{x_s}$ based on Shannon code length $l(\mathbf{x_s}) = -\log P(\mathbf{x_s})$ as used in [5, 7].

We can now compute the total generalised code length of the sequence $\mathbf{x}^n$ given a model tree T with state set $\mathcal{S}_T$, i.e. the set of leaf nodes in T, by summation of $l_\alpha(\mathbf{x_s})$ over all states $\mathbf{s} \in \mathcal{S}$ of the model $L_\alpha(\mathbf{x}^n|T) = \sum_{\mathbf{s}\in\mathcal{S}_T} l_\alpha(\mathbf{x_s})$. Generalising the 2-pass Context [5] MDL formalisation, we get the following optimisation problem with respect to the total generalised code length,

$$L_\alpha(\mathbf{x}^n|T) \equiv \sum_{\mathbf{s}\in\mathcal{S}_T} l_\alpha(\mathbf{x_s}) \tag{4}$$

$$L_\alpha(\mathbf{x}^n, T) \equiv L_\alpha(\mathbf{x}^n|T) + L(T) \tag{5}$$

$$\hat{T}_{\mathbf{x}^n,\alpha} \equiv \arg\min_T L_\alpha(\mathbf{x}^n, T) \tag{6}$$

where $L(T)$ is the code length of the model and $L_\alpha(\mathbf{x}^n, T)$ is the MDL criterion. The 2-pass Context algorithm will solve the minimisation problem of (6).

The effect of α on the model selection only manifests itself through the total generalised code length $L_\alpha(\mathbf{x}^n|T)$. From (3) we see that $L_\alpha(\mathbf{x}^n|T) \to \infty$ for both $\alpha \to \infty$ and $\alpha \to -\infty$. For $\alpha \ll 0$, the selected model will have a tendency to include states which has a low probability of occurrence in the training sequence, where as $\alpha \gg 0$ will lead to a model that include the most probable states from the training sequence.

The cost of encoding the model tree T should include both the cost of encoding the tree structure as well as the probability estimates in every node in the model tree. We define the code length of the model T as $L(T) \equiv |T| + L(p)$, where $|T|$ denotes the cost of encoding the tree structure. We can choose a code for the tree structure such that we only need 1 bit per node in the tree. The term $L(p)$ is the cost of encoding the probability estimates at every node in the model tree T. Since we in practise are operating with finite length sequences it is important to encode the probability estimates taking into account the uncertainty of the estimates. The probability estimates $\hat{p}_i$ can be encoded with precision $\delta = 2^{-q}$ by $\tilde{p}_i = \left[\frac{\hat{p}_i}{\delta}\right]\delta$, where $[\cdot]$ denotes rounding towards the nearest integer and q denotes the number of significant bits. To include the estimator error we choose the precision in terms of the variance σ_i of the estimator $\hat{p}_i$. The variance σ_i of the Laplace estimator $\hat{p}_i$ is straightforwardly derived by using the binomial distribution of the frequency counts ν_i. For every symbol we therefore encode the two integers $\left[\frac{\hat{p}_i}{\sigma_i}\right]$ and $[-\log\sigma_i]$. The case where the context continuation $\mathbf{x_s}$ is empty will be encoded by a constant. We can encode the integers using an Elias code [4] which allows us to formulate the code length of encoding the model T at a single node as $l(T) = 1 + l(p)$ where $l(p) = 4\log^*(2)$ if $n = 0$, otherwise $l(p) = \sum_{i=0}^{1}(\log^* \frac{\hat{p}_i}{\sigma_i} + \log^*(-\log\sigma_i))$. Here $\log^*(x) = \log(x) + \log\log(x) + \ldots$ including all positive terms, and we ignore the integer truncation and a constant cost found in the Elias code [4]. In order to ensure a symmetric code length, this has to be done for the probability estimates $\hat{p}_i$ for all symbols $i \in \Sigma$ in each node of the model tree.

The prediction part of our inference algorithm is based on the selected model of the past interaction sequence. In order to make a prediction the algorithm has to select a model node context. This prediction node search goes through the suffix set of $\mathbf{x}^n$, Suffix($\mathbf{x}^n$), starting with the longest proper suffix of $\mathbf{x}^n$. If the longest proper suffix node is not in the model tree, consider the second longest proper suffix of $\mathbf{x}^n$, and so forth until a model tree node is reached. The algorithm will make a prediction of the future robot-environment interaction by sampling the next symbol from the distribution $p_i^\alpha / \sum_j p_j^\alpha$ at the selected model tree node. If $\mathbf{s}$ is the selected node, then p_i is the distribution over the symbols following after $\mathbf{s}$, i.e. the distribution over the child nodes of $\mathbf{s}$, $p_i = P(i|\mathbf{s})$ for all $i \in \Sigma$.

The optimal probability distribution on symbol code length l_i is $2^{l_i} = p_i^\alpha / \sum_j p_j^\alpha$, which leads to the per symbol generalised average code length approaching the generalised entropy $L(t) = H_\alpha$. With this in mind, it is consistent with our model selection criterion to use the distribution $p_i^\alpha / \sum_j p_j^\alpha$ for doing predictions. Furthermore, the α parameter gives us a continuum of robot behaviour. Letting $\alpha \to \infty$ leads to deterministic prediction of the most probable symbol i, i.e. $\hat{i}_{\max} = \arg\max_{i \in \Sigma} p_i$, because $\frac{p_i^\alpha}{\sum_j p_j^\alpha} \to \delta_{\hat{i}_{\max}}(i)$, where $\delta_{\hat{i}_{\max}}(i)$ denotes the Dirac distribution with all its probability mass at $\hat{i}_{\max}$. This will lead to an exploitive behaviour of the robot, where it will prefer the most likely part of state space. The robot will prefer to do as the teacher has programmed it to do. In the case $\alpha \to 0$, the prediction distribution reduces to the uniform distribution, $\frac{p_i^\alpha}{\sum_j p_j^\alpha} \to \frac{1}{M}$, $\alpha \to 0$. When $\alpha \to -\infty$ the prediction again becomes deterministic leading to prediction of the most unlikely symbol, $\hat{i}_{\min} = \arg\min_{i \in \Sigma} p_i$, because $\frac{p_i^\alpha}{\sum_j p_j^\alpha} \to \delta_{\hat{i}_{\min}}(i)$. This will lead to an explorative behaviour of the robot where it will prefer the unlikely parts of state space. The robot will predict sequences of action-input symbols which it has not tried before and will do the opposite of what the teacher has programmed.

4 Summary

We presented a novel sequential learning algorithm based on an extension of the 2-pass Context algorithm [5]. The extension consists in using a MDL criterion based on generalised code length, whereby we introduce the α parameter for controlling the behaviour of the algorithm. For $\alpha \ll 0$ the algorithm is explorative and for $\alpha \gg 0$ the algorithm is exploitive. We assume that the source is a stationary ergodic random process, and our method selects a model of the source from the class of homogeneous ergodic variable order Markov chain models. Prediction of future robot-environment interaction is based on the learnt model.

References

[1] P. Bühlmann. Model selection for variable length markov chains and tunning the context algorithm. *Annals of the Institute of Statistical Mathematics*, 52:287–315, 2000.

[2] L. L. Campbell. A coding theorem and Rényi's entropy. *Information and Control*, 8(4):423–429, 1965.

[3] I. Csiszar and Z. Talata. Context tree estimation for not necessarily finite memory processes, via bic and mdl. *IEEE transaction on Information Theory*, 52(3):1007–1016, March 2006.

[4] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transaction on Information Theory*, 21(2):194–203, 1975.

[5] R. Nohre. *Some Topics in Descriptive Complexity.* PhD thesis, Linköping University, 1994. Chapter 3.

[6] A. Rényi. Some fundamental questions of information theory. *MTA III. Oszt. Közl.*, 10:251–282, 1960.

[7] J. Rissanen. A universal data compression system. *IEEE Transaction on Information Theory*, 29(5):656–664, 1983.

[8] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific, 1989.

[9] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25:117–149, 1996.

[10] C. E. Shannon. The mathematical theory of communication. *Bell Sys. Tech. Journal*, 27:379–423,623–656, 1948.

A neuro-inspired architecture for goal inference and decision making in joint action tasks*

Nzoji Hipólito† Luís Louro† Estela Bicho† Wolfram Erlhagen‡

Abstract

We propose a cognitive control architecture, based on neuro-plausible principles, for autonomous robots in the context of a joint search task. It endows the robots with cognitive skills like memory, prediction and decision making. As a mathematical framework we use a coupled system of dynamic neural fields. It represents a simplified model of a joint action circuit in which task-relevant information is represented by self-stabilized activation patterns of local neural populations. The architecture is validated using a robot simulation environment. In particular, we show the impact of memory and prediction on the ability to infer the action goal of the partner even if occluding surfaces temporally disrupt the continuity of sensory information. This goal inference capacity is essential for an efficient team behavior since it allows the observer to select an adequate complementary action.

Keywords: Dynamic Field Theory, Neuronal Representation, Cognitive Robotics.

1 Introduction

One of the major challenges for today's robotics is the development of autonomous agents that are able to engage in joint action tasks with humans or other robots in natural work environments. Successful joint action requires a set of high-level cognitive functions. Most importantly, the robot should take into account the consequences of an action displayed by a partner when deciding about its own future actions in a joint task. To allow for a fluent team behavior, this decision process very often requires to register the intention of the partner way before the observed motor act is completed. The selection and timing of an appropriate

*The present research was conducted in the context of the fp6-IST2 EU-project JAST (proj.nr. 003747)

†Dept of Industrial Electronics, University of Minho, Portugal. E-mails: nhipolito@dei.uminho.pt, llouro@dei.uminho.pt, estela.bicho@dei.uminho.pt

‡Dept of Mathematics for Science and Technology, University of Minho, Portugal. E-mail: wolfram.erlhagen@mct.uminho.pt.

complementary action thus critically depends on the ability to predict and anticipate effects of observed actions.

Our group develops and tests robot control architectures for joint action tasks that are inspired by our understanding of the cognitive and neural processes supporting social interactions in humans and other primates. We believe that such a neuro-cognitive approach will ultimately allow to realize the ambitious goal of creating socially relevant robots. In recent years, an increasing number of neurophysiological and neuroimaging studies have directly investigated perception and action in a social context. A circuit of cortical areas defined by neuronal populations with specific functionalities has been identified that is believed to implement a mapping from an observed intentional action onto the motor representation of an action to be executed (for a discussion see [6].

For the robot control architecture in the context of a joint action task we adopt the central idea that this mapping is controlled by a representation of the partner's action goal. Specific motion cues trigger this goal representation which in turn biases the observers's decision for a complementary action. A second characteristic of the proposed control architecture, which reflects insights from neurophysiology, concerns the nature of the representations. A common feature of many neural populations in higher brain areas is that the population activity may persist over extended periods of time upon cessation of the external input which has initially triggered the activity pattern. It is commonly believed that self-sustained activity is linked to cognitive functions like working memory and decision making [9]. A robot working in cluttered and dynamic environments is frequently faced with the problem that task relevant objects or the partner, move out of sight due to occluding surfaces for instance. The temporarily lack of direct sensory information should of course not disrupt the capacity to plan and maintain a goal-directed action (e.g., heading toward the hidden object) or to infer the action goal of the partner.

As a mathematical framework to model the evolution of the neural activity patterns in the joint action circuit, we use Dynamic Neural Fields (DNFs) ([1, 13]; for a recent review of the mathematical analysis see [4] ; for previous applications in the robotics domain see, e.g., [11, 3, 10, 7]). The attractor dynamics of DNFs is well suited to model the integration over time of input from connected populations and sources external to the circuit. When a certain activation threshold is reached, the recurrent excitatory interactions between neurons start to dominate the field dynamics leading to self-sustained patterns. The capacity to form decisions is in turn mediated by lateral inhibition.

Here we present simulation results of a validation of the dynamic control architecture in a joint search task that involves two mobile robots equipped with a vision system, a high-degree of freedom arm and a hand. The joint search task consists in finding different objects distributed in the workspace with the goal to transport them to a predefined place. The primary challenge for the team is an efficient division of the search space among the part-

ners. However, depending on the size of the object a direct physical interaction of the two robots may be necessary. We do not allow for a direct communication of intentions among the robots since we want to use the same control architecture also in joint action tasks that include humans.

The paper is organized as follows: Section 2 introduces the proposed neuro-plausible architecture. Section 3 presents the coupled system of dynamic neural fields for the implementation of the cognitive functionalities. The results of the joint search task are described in section 4. The paper ends with a discussion and a short outlook.

2 Neuro-inspired control architecture

In the search task, multiple objects may be simultaneously sensed. As a consequence, each robot must be able to make a decision about the object to-be-fetched next. To guarantee for an efficient team strategy, the control architecture should establish sensory-motor mappings that go beyond the simple rule to move towards the most salient, that is, closest object. The decision process has to take into account the interpretation of the observed motion of the partner robot in terms of its current goal. As a simple strategy for the present search task, the goal inference may be based on estimating the change over time of the relative distance between the moving robot and objects in the workspace. In cluttered and dynamic environments this necessarily requires the ability to extrapolate past trajectory information to future positions that may be occluded from view. Once a potential target object has been inferred, the size of the object is an important cue that biases the decision about an adequate complementary behavior. For a small size object, which can be carried alone, the observing robot should look for another small object in a different part of the search space. For a large object, which can only be manipulated together, the decision should be to head towards that object to help carrying it.

Fig 1 presents a sketch of the multi-layered architecture that controls the action of the observing robot R1. In the object memory layer (ML) the information about the location of large and small objects is stored, whereas the action observation layer (OL) represents the position of the partner robot R2. Both information sources are integrated in the action simulation layer (ASL) which encodes the prediction about the object the partner is currently heading to. This prediction biases the decision represented in the goal layer (GL) about the next action of the observer. Finally, GL is linked to an action execution layer (AEL) which contains a sequence of movement primitives necessary to achieve the desired end state (e.g., approaching, grasping, transporting and placing the object at a predefined position). In the present simulation examples, our focus is on the approaching phase since we are mainly interested in studying how inferred goals of the partner robot affect the object selection process.

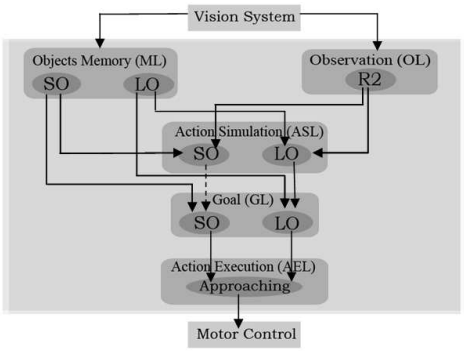


Figure 1: Schematic view of the multi-layered control architecture together with the connectivity between layers (arrows). In the ***Observation*** layer, the visual description of the motion displayed by the partner robot R2 is represented and stored. The ***Objects Memory*** layer contains populations which memorize the visual information about the location of small (sublayer SO) and large objects (sublayer LO). The ***Action Simulation*** layer combines these information to make a prediction about the object R2 is currently heading to. The ***Goal*** layer represents the decision of the observer R1 which object to fetch next. This decision takes into account the inferred goal of R2. The mapping between the representations of small objects in layers ASL and GL is inhibitory (dashed arrow), meaning that R1 should not select the same object. The mapping between large objects is excitatory (solid arrow), meaning that the two robots should directly cooperate in manipulating these objects. The ***Action Execution*** layer contains a representations of the movement primitive "approaching a goal object". This primitive controls the motor commands for the mobile platform.

3 Dynamic field model

To model the self-stabilized representations in each layer, we use the mathematical framework of dynamic neural fields. It has been originally introduced to describe the formation of localized spatio-temporal patterns in neural tissue [1]. For the robotics application we assume that the fields encode the parameter direction in the horizontal plane relative to a fixed frame of reference. Our motivation comes from the so-called populations of head-direction cells originally found in freely moving rats [12, 14]. Each individual head-direction cell has its maximum firing rate at only one particular direction, and firing rates decrease monotonically on either side as the angular distance to the preferred direction increases. Thus for a whole population of head-direction cells a particular direction is represented by a localized firing pattern in the direction space. We adopt this population coding idea for the robotics applications. It is important to note, however, that according to the functionality of the different layers of the control architecture, the self-sustained patterns have different meanings (e.g., representing object memory or decisions).

In each layer i, the activity $u_i(\phi,t)$ at time t of a neuron representing direction ϕ is described by the following integro-differential equation of Amari-type [1]:

$$\begin{aligned}\tau_{\mathrm{i}}\frac{\delta u_{\mathrm{i}}(\phi,t)}{\delta t} &= -u_{\mathrm{i}}(\phi,t) + S_{\mathrm{i}}(\phi,t) \\ &+ \int_0^{2\pi} w_{\mathrm{i}}(\phi-\phi')f_{\mathrm{i}}(u_{\mathrm{i}}(\phi',t))d\phi' + h_{\mathrm{i}} \end{aligned} \tag{1}$$

where the constants $\tau_i > 0$ and $h_i < 0$ define the time scale and the resting level of the of the field dynamics, respectively. The integral term describe the intra-field interactions. It is assumed 1) that interaction strength, $w_{\mathrm{i}}(\phi,\phi')$, between any two neurons ϕ and ϕ' depends only on the distance of their preferred directions, and 2) that nearby cells excite each other, whereas separated pairs of cells have a mutually inhibitory influence. For the present implementation we have used the following kernel of lateral inhibition type:

$$w_{\mathrm{i}}(\phi) = A\exp(-\phi^2/2\sigma^2) - w_{inhib} \tag{2}$$

where $w_{inhib} > 0$ is a constant and $A > 0$ and $\sigma > 0$ describe the amplitude and the standard deviation of a Gaussian, respectively. The transfer function $f(u)$ is chosen of sigmoidal shape with slope parameter β and threshold u_O:

$$f_{\mathrm{i}}(u_{\mathrm{i}}) = \frac{1}{1+exp\left(-\beta(u_{\mathrm{i}}-u_0)\right)}. \tag{3}$$

The parameters describing the interaction kernel are adjusted to guarantee for a bi-stable regime of the field dynamics in which a homogeneous solution coexists with a localized pulse solution. Starting from the homogeneous resting state, the field dynamics may evolve in response to a localized input $S_i(\phi)$ of adequate intensity a self-stabilized pulse. The input $S_i(\phi,t)$ to field i comes from connected fields j or from sources external to the circuit. To simulate the input from the real vision system to layers OL and ML, Gaussians are used for the examples shown here. It is assumed for simplicity that 1) intra-field connections exist only between corresponding neurons representing the same preferred direction ϕ, and 2) that the input strength is proportional to the activity $f(u_j(\phi,t))$. The summed input to a field in layer i from all connected fields is given as $S_i(\phi,t) = k\sum_j \pm f_j u(\phi,t)$. All intra-field connections are excitatory (+) except the connections between the fields representing small objects (SO) in layers ASL and GL, which are inhibitory (-). The parameter k scales the total input relative to the threshold for triggering a self-sustained pattern. This guarantees that the coupling is weak and the field dynamics is dominated by the recurrent interactions.

In the following we briefly summarize the adaptations to the basic field structure we have made to reflect the specific functionality of the various layers (for mathematical details see [5]). The evolution of a single pulse in layer GL implements a decision making capacity since lateral inhibition suppresses the activation of other pools of neurons which may also get input

from connected layers. To represent and memorize multiple objects at the same time, the interaction kernel for layer ML is adapted to allow for multi-pulse solutions. Note that the two types of objects, large and small, are represented within the different layers by separated fields.

For the robotics applications, two characteristics of the standing pulse solutions are important. First, the translation invariance of the fields allows to update the memorized information. Changes in the external input (e.g., due to self-motion) may displace the waveform stable pulse to encode this new information. Second, memory and decisions should be updated from time to time. A forgetting mechanisms can be implemented by defining a proper dynamics for global inhibition $h_i < 0$. For sufficiently large values of $|h_i|$, the field dynamics becomes mono-stable. As a result, a (multi-) pulse solution decays back to resting level [3].

The *Observation* layer represents a prediction about the position of the other robot even if the partner is temporarily out of sight. This prediction is implemented as a self-stabilized traveling wave in direction space. A specific choice of an asymmetric interaction kernel causes a localized pulse to travel without disturbing its shape:

$$w(\phi, t) = w(\phi) + \eta(t) w'(\phi) \tag{4}$$

where $w'(\phi)$ denotes the derivative of the Gaussian weight function used in equ. 1. The instantaneous angular velocity of this travelling wave is given by

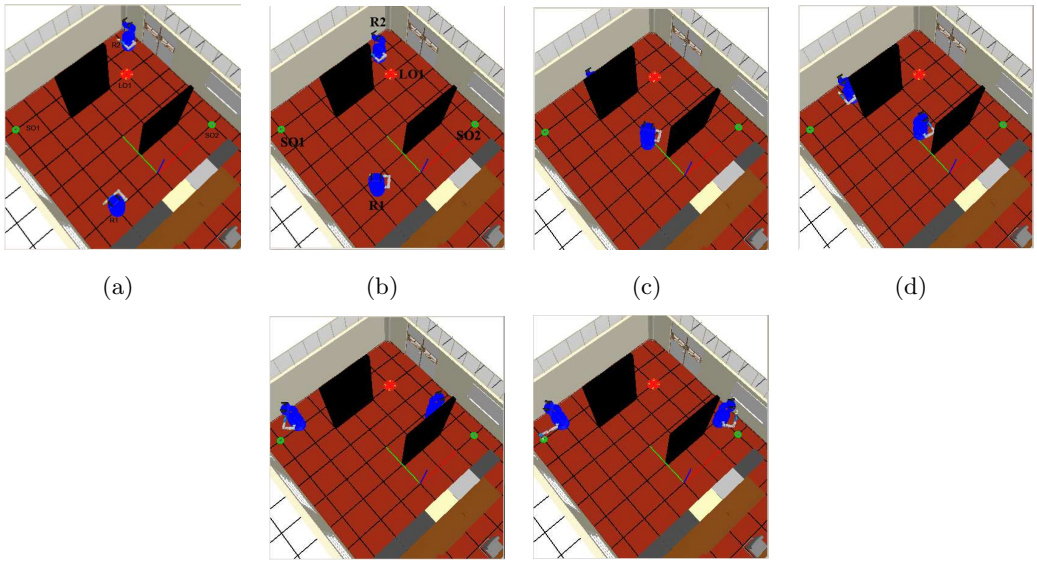
$$\varpi_{wave}(t) = -\frac{\eta(t)}{\tau}. \tag{5}$$

Since anticipation is a fundamental capacity for joint action tasks, the wave velocity is chosen larger than the directional change per time unit of the partner robot which is estimated initially from direct sensory information.

4 Simulation Results

In the simulations, each neural field as well as the dynamics of heading direction and path velocity are integrated numerically using the forward Euler method. Each field is sampled spatially along ϕ with a sampling distance of 2°. This value is sufficient to guarantee that the behavior in discrete case approximates quite well the behavior of the continuous field equation.

Figure 2 and figures 3 to 6 show snapshots of the overt behavior of the robots and the neural field representations of the observer R1, respectively. The robots are initially placed as illustrated in panel (a). R1 senses the three objects SO1, SO2 and LO1, and also detects robot R2. In this example, R1 infers that object S01 is the current action goal of R2 and decides to select the other small object S02. Initially, however, robot R1 moves towards object LO1, panels (b) to (c), since R2's presence near the large object triggers an



(a) (b) (c) (d)

(e) (f)

Figure 2: The robot simulation environment for the joint search task is shown. In this example, the two robots, R1 and R2, have to find and carry two small objects (S01 and S02) and one large object (L01), which are distributed in the workspace, to a common place (white square). The 6 snapshots of the robot show the motion of the two robots towards S01 and S02, respectively. Note that the spatio-temporal continuity of visual information about the partner robot appears to be disrupted due to occluding surfaces. For a detailed description of the snapshots see the text.

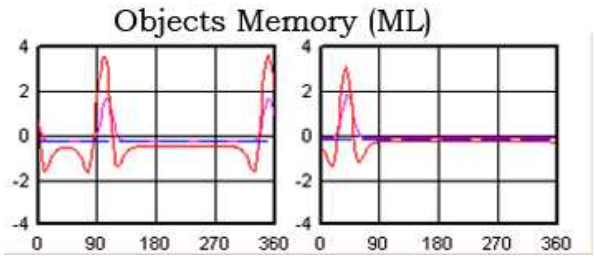


Figure 3: Neural field representation of objects seen by R1 at the beginning of the experiment shown in Fig.2. Two small objects (S0-field left) and one large object (L0-field right) are detected and memorized by self-stabilized pulses.

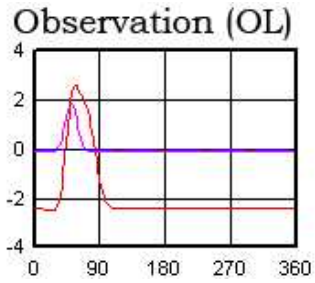


Figure 4: R1's representation of the motion of the partner R2. The visual information about the change in direction of R2 triggers a travelling wave in direction space. As shown by the snapshot of the wave (red plot), the internal representation is ahead of the actual position of R2 (magenta plot) sensed by the visual system.

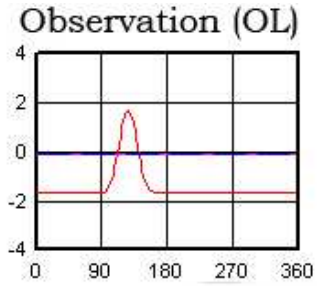


Figure 5: Snapshot of the self-stabilized travelling wave after R2 has become occluded from view at about 70°. The wave mechanism allows to extrapolate past trajectory information into the future.

activation pulse at that direction in the SO-field of the *Action Simulation* layer)(not shown). Subsequently, the integration of the motion information triggers an anticipatory traveling wave in the *Observation* layer that represents future positions of R2 (see fig.4). When R2 becomes occluded by the wall (panel (c)) the wave continues to travel with a constant speed (see fig. 5). Via the inter-field couplings, the activity in layers ML and OL trigger the evolution of a pulse solution centered at the location of S01 (at about 135°) in the S0-field of the *Action Simulation* layer (fig.6). This pulse represents the prediction of robot R1 that SO1 is the current goal of R2. Since the couplings between the S0-fields in the *Action Simulation* and the (*Goal* layer) are inhibitory, the existence of the pulse in ASL causes a suppression of activity below resting level at corresponding sites of the S0-field in the *Goal* layer. As a result, the decision of R2 represented as a pulse in that field (not shown) is biased towards the selection of the other small object S02. It is important to note that when R2 starts to move, object S02 becomes hidden from view due to the occluding surface. The capacity to

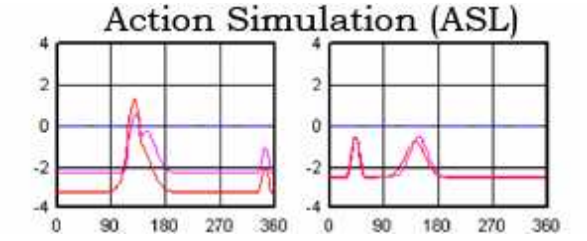
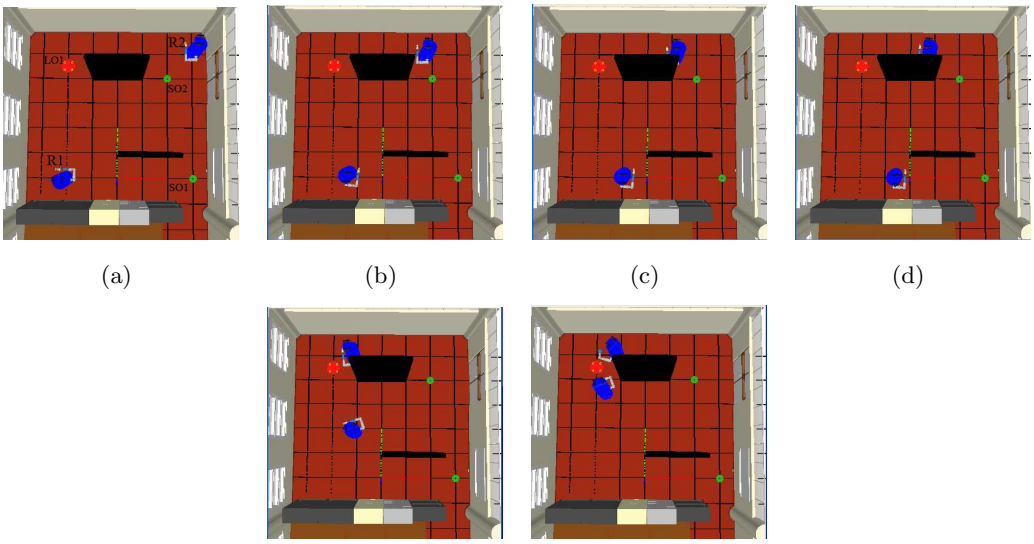


Figure 6: The input form the object memory layer (ML) together with the predictive information about the position of R2 in layer ML trigger the evolution of a a suprathreshold pulse solution in the SO-field of layer ASL. It represents the anticipated action goal of R2, which is in the present example object S01.

memorize object location is thus crucial for a successful task execution.



(a) (b) (c) (d)

(e) (f)

Figure 7: Snapshots of the motion of the two robots are shown in a different scenario. They shall illustrate the decision of robot R1 to help carrying the large object L01 which R1 has inferred as a the current action goal of R2. Again, the self-stabilized nature of the internal representations is essential to make and maintain the decision in the presence of occlusion. For a detailed description of the snapshots see the text.

The manipulation of large objects requires the direct physical interaction between the two robots. A simple team strategy to avoid the part of the search space which is currently the focus of intention of the partner is thus not feasible. Inferring that a large object is the current action goal should bias the decision of the observer to head towrd the same object.

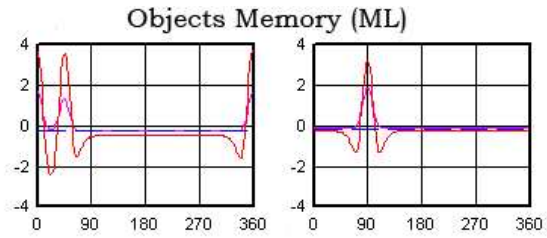


Figure 8: Neural field representation of objects seen by R1 at the beginning of the experiment shown in Fig.7. Two small objects (S0-field left) and one large object (L0-field right) are detected and memorized by self-stabilized pulses.

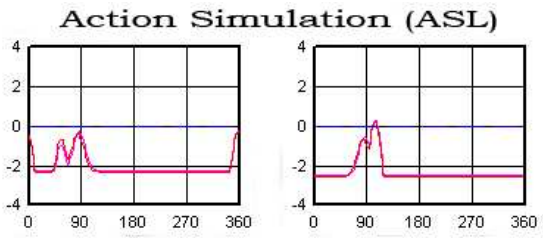


Figure 9: As shown by a snapshot of the evolving suprathreshold pulse in the LO-field of layer ASL, R1 predicts that R2 is heading towards object L01. When the wave in layer ML reaches the position of L01, the input from ML and OL to ASL is sufficient to trigger a self-stabilized pulse in that layer.

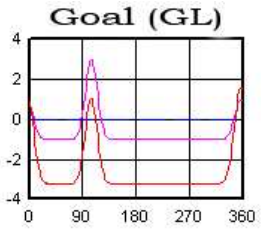


Figure 10: The pulse in the L0-field of layer GL represents the decision of R1 to head also towards object L01.

The snapshots in figure 7 illustrate this scenario. As can be seen in the *Object Memory* layer, initially, observer R1 detects two small object, SO1 at 0° and SO2 at about 45°, a large object LO1 at 90° and robot R2 located in the direction 45° (not shown). The relative proximity of R2 and S02 lead to a decision of R1 to move in the direction of S01. However, the sensed motion of R2 triggers the evolution of an anticipatory travelling wave in the direction of object LO1 about 100° relative to the new position of R1. As a result, the total input to the LO-field of the *Action Simulation* layer creates a pulse at that direction (see the snapshot in 9). Since the couplings between the L0-fields of the *Action Simulation* and the *Goal* layer are excitatory, the input to GL represented by the wave activity changes the initial selection of object S01 and consequently the motor behavior of R1 (compare snapshots (d)and (e). Its final decision represented by the pulse in layer GL (figure 10) is to also focus on object L01. This change in behavior reflects the high priority for cooperative behavior which is implemented in the control architecture. By changing the relative strength of the couplings between layer ASL and layer GL, it is also possible to maintain the initial decision to pick up object S02 which is closest to the initial position of R1.

5 Conclusion

We have presented a control architecture for autonomous robots engaged in a joint search task, that is based on neuro-plausible principles. The distributed architecture reflects the notion that cognitive processes in the brain unfold over time under the influence of multiple internal and external influences [2]. The self-stabilized nature of the neural population representations allows to guide complex behavior which goes beyond a simple input-output information processing scheme. Memorized information about the location of objects or the capacity to predict that a partner continues moving behind an occluding surface with the intention to reach a certain goal (object) are crucial cognitive skills for joint action tasks.

The choice of neural fields as a mathematical description of population dynamics simplifies or ignores many aspects of processing in the brain. However, the fact that they can be mathematically analyzed is an important advantage when trying to build cognitive robots [4].

The architecture is formalized as a system of weakly coupled dynamic neural fields. The weak coupling guarantees that the input acts essentially as an external perturbation of the dynamics which is dominated by the interactions within the field. In the present simulation examples, the couplings were set by hand. Recently, we have started to study correlation based learning rules to establish the inter-field connections during learning and practice [8]. New mathematical challenges concern the convergence and stability of the learning dynamics. In the context of the JAST project we are currently adapting and testing the proposed architecture for a a joint construction scenario which starts when all objects have been carried by the team to the predefined place. This scenario contains a relatively large number of

distinct action sequences. Consequently, the mapping from action observation onto action selection becomes much more complex.

References

[1] S Amari. Dynamics of pattern formation in lateral-inhibitory type neural fields. *Biological Cybernetics*, 27:77–87, 1977.

[2] R Beer. Dynamic approaches to cognitive science. *Trends in Cognitive Science*, 4:91–98, 2000.

[3] E Bicho, P Mallet, and G Schöner. Target representation on an autonomous vehicle with low-level sensors. *The International Journal of Robotics Research*, 19:424–447, 2000.

[4] S Coombes. Waves, bumps, and patterns in neural field theories. *Biological Cybernetics*, 93:91–108, 2005.

[5] W Erlhagen and E Bicho. The dynamic neural field approach to cognitive robotics. *Journal of Neural Engineering*, 5:36–54, 2006.

[6] W Erlhagen, A Mukovskiy, and E Bicho. A dynamic model for action understanding and goal-directed imitation. *Brain Research*, 1083:174–188, 2006.

[7] W Erlhagen, A Mukovskiy, E Bicho, G Panin, C Kiss, A Knoll, H van Schie, and H Bekkering. Goal-directed imitation for robots: a bio-inspired approach to action understanding and skill learning. *Robotics and Autonomous Systems*, 54:353–360, 2006.

[8] W. Erlhagen, A. Mukovskiy, F. Chersi, and E. Bicho. On the development of intention understanding for joint action tasks. In *6th IEEE Int. Conf. on Development and Learning*. Imperial College London, 11-13 July , 2007.

[9] J I Gold and M N Shadlen. Banburismus and the brain: the relationship between sensory stimuli, decisions, and reward. *Neuron*, 36:299–308, 2002.

[10] M Quoy, S Moga, and P Gaussier. Dynamical neural networks for planning and low-level robot control. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33:523–532, 2003.

[11] G Schöner, M Dose, and C Engels. Dynamics of behavior: theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16:213–245, 1995.

[12] J S Taube and J P Bassett. Persistent neural activity in head direction cells. *Cerebral Cortex*, 13:1162–1172, 2003.

[13] J G Taylor. Neural bubble dynamics in two dimensions: foundations. *Biological Cybernetics*, 80:393–409, 1999.

[14] K Zhang. Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory. *Journal of Neuroscience*, 16:2112–2126, 1996.

Ant–swarm robotics for a dynamic cleaning problem

Yaniv Altshuler* Vladimir Yanovsky* Israel A. Wagner*, †
Alfred M. Bruckstein†

Abstract

Several recent works considered multi a(ge)nt robotics in static environments. In this work we examine ways of operating in dynamic environments, in which changes take place independently of the agents' activity. The work focuses on a dynamic variant of the known *Cooperative Cleaners* problem (described and analyzed by Wagner and Bruckstein in [10]). This problem assumes a grid, having "dirty" pixels or tiles, that form a connected region of the grid. Several agents move in this dirty region, each having the ability to "clean" the place it is located in. The dynamic variant of the problem involves a deterministic expansion of dirt in the environment, simulating a spreading of *contamination*, or *fire*. A cleaning protocol for the problem is presented, as well as several analytic lower and upper bounds on its performance.

Keywords : Cooperative cleaning, Dynamic environments, Swarm analysis.

Introduction

In the world of living creatures, "simple minded" animals like ants or birds cooperate to achieve common goals with surprising performance. It seems that these animals are "programmed" to interact locally in such a way that the desired global behavior is likely to emerge even if some individuals of the colony die or fail to carry out their task for some other reasons. It is suggested to consider a similar approach to coordinate a group of robots without a central supervisor, by using only local interactions between the robots. When this decentralized approach is used, much of the communication overhead (characteristic to centralized systems) is saved, the hardware of the robots can be fairly simple, and better modularity is achieved. A properly designed system should achieve reliability through redundancy. Significant research effort has been invested during the last few years in design and simulation of multi-agent robotics and intelligent swarm systems. (see e.g. [10, 7, 2, 9, 5, 8]). Unfortunately, the mathematical \ geometrical theory of such multi-agents systems is far from being satisfactory, as

*Computer Science Department, Technion, Haifa, Israel. E-mails: {yanival, volodyan, wagner, freddy}@cs.technion.ac.il

†IBM Haifa Labs, MATAM, Haifa, Israel.

pointed out in [3] and many other papers. In this work we will examine a problem in which the agents must work in a dynamic environment — where changes may take place, that are independent and certainly not caused by the agents' activity. In the spirit of [4] we consider simple robots with only a bounded amount of memory (i.e. *finite-state-machines*).

The dynamic cooperative cleaners problem

We shall assume that the time is discrete. Let G denote a two dimensional integer grid $\mathbf{Z}^2$, whose vertices (or "*tiles*") have a binary property called '*contamination*'. Let $cont_t(v) \in \{on, off\}$ state the contamination state of the tile v at time t. Let $F_t = \{v \in G \mid cont_t(v) = on\}$ be the contaminated sub-graph of G at time t. We assume that F_0 is a single connected component. Our algorithm will preserve this property along its evolution. Let a group of k agents that can move on the grid G (moving from a tile to its neighbor in one time step) be placed at time t_0 on F_0. Each agent is equipped with a sensor capable of telling the *contamination* status of all tiles in the digital sphere of diameter 7, which surrounds the agent. An agent is also aware of other agents which are located in these tiles, and all the agents agree on a common direction. Each tile may contain any number of agents simultaneously. Each agent is equipped with a memory of size limited to $O(k)$ bits. When an agent moves to a tile v, it can clean this tile (i.e. $cont(v) \leftarrow off$). The agents do not have any prior knowledge of the shape or size of F_0 except that it is a single and simply connected component.

The contaminated region F_t is assumed to be coated at its boundary by a rubber-like elastic barrier, dynamically reshaping itself to fit the evolution of the contaminated region over time. The purpose of this barrier is to guarantee the preservation of the simple connectivity of F_t. When an agent cleans a contaminated tile, the barrier simply moves back, in order to fit the void previously occupied by the cleaned tile. This is demonstrated in Figure 1. Every d time steps the contamination spreads. That is, if $t = nd$ for some positive integer n, then : $\forall v \in F_t \ \forall u \in 4Neighbors(v)$, $cont_{t+1}(u) = on$. The order in which contaminated tiles influence their clean neighbors is as follows — first, all the clean tiles located to the right of a contaminated tile are becoming contaminated. Then, the clean tiles located below contaminated tiles, followed by clean tiles located to the left of such tiles. Finally, clean tiles located above a contaminated tile are affected. This process is demonstrated in Figure 2. While the contamination spreads, the elastic barrier stretches accordingly, as demonstrated in Figures 2 and 3. For the agents who travel along the tiles of F, the barrier signals the boundary of the contaminated region. When an agent detects a contaminated tile which is "on the other side" of the barrier it treats it as though it is a clean tile.

The agents' goal is to clean G by eliminating the contamination entirely, meaning that the agents must ensure that : $\exists t_{success} \ s.t \ F_{t_{success}} = \emptyset$ In addition, it is desired that the time $t_{success}$ will be minimal.

In this work we impose the restriction of no central control and full 'de-centralization',

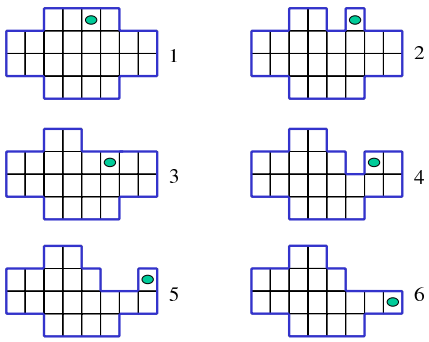


Figure 1: A demonstration of the evolution of the elastic barrier as a result of the movement and cleaning of an agent according to the **SWEEP** protocol, described later.

i.e. all agents are identical and no explicit communication between the agents is allowed. An important advantage of this approach, in addition to the simplicity of the agents, is fault-tolerance — even if almost all the agents cease to work before completion, the remaining ones will eventually complete the mission, if possible.

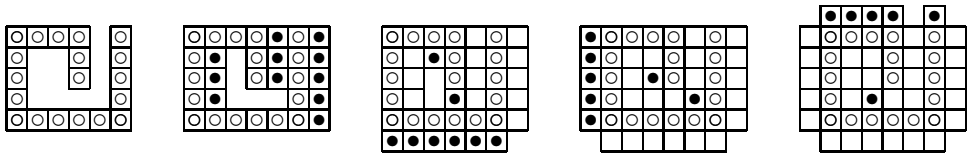


Figure 2: A demonstration of the spreading process. The left chart represents the contaminated region prior to the spread, while the rest of the charts represents the different sub-phases of the first phase of the contamination process. The tiles marked with hollow circles represent the "original" contaminated tiles (meaning, tiles that were already contaminated already, prior to the spread). Tiles marked with filled circles represent the *temporary-contaminated* tiles of each sub-phase. Notice that a tile can be marked as *temporary-contaminated* only once through the spreading process.

The cleaning protocol

In order to solve the *Dynamic Cooperative Cleaners* problem we propose the **SWEEP** cleaning protocol. Generalizing an idea from computer graphics (presented in [6]), this protocol preserves the connectivity of the *contaminated* region by preventing the agents from cleaning *critical points* — points which when cleaned disconnect the contaminated region. At each time step, each agent cleans its current location (assuming it is not a critical point), and moves according to a local movement rule, creating the effect of a clockwise traversal along

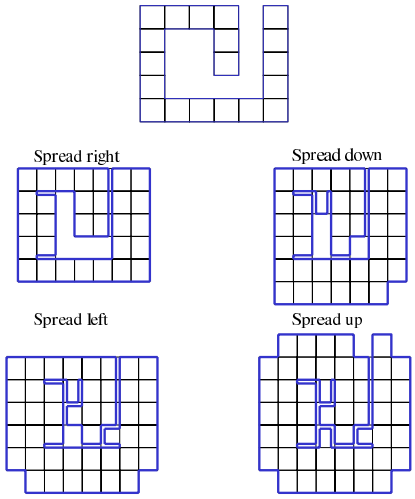
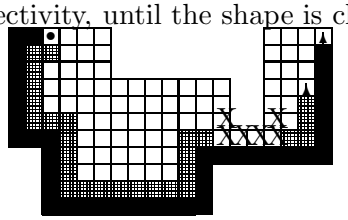


Figure 3: The top chart represent the initial region prior to the contamination spread. The four charts below demonstrate the evaluation of the barrier throughout the spreading process.

the boundaries of the contaminated region. As a result, the agents "peel" layers from the region, while preserving its connectivity, until the shape is cleaned entirely.



An example of two agents using the **SWEEP** protocol, at $t = 40$, when $d > 40$. All the tiles were contaminated at time 0. The black dot denotes the starting point of the agents. The X's mark the *critical points*.

Overview of the results

Motivation

Since we know no easy way to decide whether k agents can successfully clean an instance of the *Dynamic Cooperative Cleaners* problem, producing bounds for the proposed cleaning protocol is important for estimating its efficiency. Analyzing the performance of the above defined protocol is quite difficult. Due to the preservation of the *critical points*, such points can be visited many times by the agents before being cleaned. Furthermore, due to the dynamic nature of the problem, the shape of the contaminated region can change dramatically during the cleaning process.

Definitions

Let S_t denote the size of the contaminated region F at time t, namely the number of tiles (i.e. grid points) in F. Actually, F defines a dichotomy of $\mathbf{Z}^2$ into F and $\overline{F} = \mathbf{Z}^2 \setminus F$. The boundary of the contaminated region F is denoted as ∂F, defined as $\partial F = \{(x,y) \mid (x,y) \in F \wedge (x,y)$ has an $8Neighbor$ in $(G \setminus F)\}$

A *path* in F is defined to be a sequence $(v_0, v_1, \ldots, v_n)$ of tiles in F such that every two consecutive tiles are 4 *connected* (the *Manhattan* distance between them is 1). The *length* of a *path* is defined to be the number of tiles in it. Let tile v be called a *critical point* if there exist $v_1, v_2 \in 4Neighbors(v)$ for which all paths connecting v_1 and v_2, included in $8Neighbors(v)$, necessarily pass through v (where v, v_1, v_2 and all said paths are from F). We shall denote by c_t the circumference of F at time t, defined as follows: let v_0 and v_n be two 4 *connected* tiles in ∂F_t, and let $C_t = (v_0, v_1, \ldots, v_n)$ be the shortest *path* connecting v_0 and v_n, which contains all the tiles of ∂F_t and only such tiles. If there are several different shortest paths then let C_t be a one of them. Notice that C_t may contain several instances of the same tile, if this tile is a *critical point*(meaning that C_t is an ordering of the tiles of ∂F_t, in which multiple instances of tiles that are *critical points* are allowed). c_t will be defined as the length of C_t.

For some $v \in F_t$ let $Strings_t(v)$ denote the set of all *paths* in F_t that begin in v and end at any *non-critical point* in ∂F_t, and let $w(F_t, v)$ denote the *depth* of v — the length of the shortest *path* in $Strings_t(v)$ (unless v is a *critical point* in which case its *depth* is defined to be zero). Let $W(F_t)$ denote the *width* of F_t, defined as the maximal depth of all the tiles in F_t, i.e. : $W(F_t) = \max\{w(F_t, v) \mid v \in F_t\}$.

Results

In order to demonstrate the hardness of the problem, we first present below lower bounds for the cleaning time which will be required of *any* cleaning protocol which might be used by the agents (Theorems 1 and 2).

Theorem 1. *Using any cleaning protocol, the area of the contaminated region at time t can be recursively lower bounded, as follows :*

$$S_{t+d} \geq S_t - d \cdot k + 2\sqrt{2 \cdot (S_t - d \cdot k) - 1}$$

Corollary 1. *Using any cleaning protocol, k agents cleaning a contaminated region of size S_0, which spreads every d time steps, could not clean it if :*

$$S_0 > \left\lceil \frac{1}{8}d^2k^2 + dk + \frac{1}{2} \right\rceil$$

Theorem 2. *Using any cleaning protocol, a lower bound for the area of the contaminated region at time $t = i \cdot d$ for some $i \in \mathbb{N}$ is the minimal positive t solution of the following*

implicit equation :

$$2t = \sqrt{2(S_t - dk) - 1} - \sqrt{2(S_0 - dk) - 1} + \ln\left(\frac{\sqrt{2(S_t - dk) - 1} - \frac{dk}{2}}{\sqrt{2(S_0 - dk) - 1} - \frac{dk}{2}}\right)^{\frac{dk}{2}}$$

The efficiency of the **SWEEP** protocol is demonstrated by the upper bound on its cleaning time, presented in Theorems 3, 4 and 5 below.

Theorem 3. *Assume that k agents start cleaning a simple connected region F_0 at some boundary point p_0 and work according to the **SWEEP** protocol, and denote by $t_{success}(k)$ the time needed for this group to clean F_0. Then it holds that:*

1. *If $(t = \frac{8(|\partial F_0|-1)\cdot(W(F_0)+k)}{k} + 2k)$ is not greater than d, then $t_{SUCCESS} = \lceil t \rceil$. This also holds for static environments, since in such cases $d \to \infty$.*

2. *Otherwise $(t > d)$: if F_0 is convex, find the minimal t for which :*

$$\sum_{i=d+1}^{t} \frac{1}{S_0 - 1 + 2\lfloor \frac{i}{d} \rfloor^2 + (c_0 + 2)\lfloor \frac{i}{d} \rfloor} \geq \gamma + \frac{8}{k} \cdot \left\lfloor \frac{t}{d} \right\rfloor$$

where

$$\gamma \triangleq \frac{8(k + W(F_0))}{k} - \frac{d - 2k}{|\partial F_0| - 1}$$

3. *Otherwise ($t > d$ but F_0 is not convex), find the minimal t for which :*

$$\sum_{i=d+1}^{t} \frac{1}{S_0 - 1 + 2\lfloor \frac{i}{d} \rfloor^2 + (c_0 + 2)\lfloor \frac{i}{d} \rfloor} \geq$$

$$\geq \alpha + \frac{8}{2k}\sqrt{\beta + 4\left(\left\lfloor \frac{t}{d} \right\rfloor^2 + \left\lfloor \frac{t}{d} \right\rfloor\right)}$$

where

$$\alpha \triangleq 8 + \frac{8}{2k} - \frac{d - 2k}{|\partial F_0| - 1} \quad \textit{and} \quad \beta \triangleq 2S_0 + 2c_0 - 1$$

Theorem 4. 1. *If $(t = \frac{8(|\partial F_0|-1)\cdot(W(F_0)+k)}{k} + 2k)$ is not greater than d, then $t_{SUCCESS} = \lceil t \rceil$. This also holds for static environments, since in such cases $d \to \infty$.*

2. *Otherwise $(t > d)$: if F_0 is convex, find the minimal μ for which :*

$$\psi\left(\mu + \frac{c_0 + 2 - \gamma_2}{4}\right) - \psi\left(\mu + \frac{c_0 + 2 + \gamma_2}{4}\right) + \gamma_1 - \frac{\gamma_2 \cdot \gamma}{d} - \frac{8 \cdot \gamma_2}{d \cdot k} \cdot \mu \geq 0$$

where :

$$\gamma_1 \triangleq \psi\left(1 + \frac{c_0 + 2 + \gamma_2}{4}\right) - \psi\left(1 + \frac{c_0 + 2 - \gamma_2}{4}\right)$$

and :

$$\gamma_2 \triangleq \sqrt{(c_0 + 2)^2 - 8 \cdot (S_0 - 1)}$$

and :

$$\gamma \triangleq \frac{8(k+W(F_0))}{k} - \frac{d-2k}{|\partial F_0|-1}$$

3. *Otherwise ($t > d$ but F_0 is not convex), find the minimal μ for which :*

$$\psi\Big(\mu + \frac{c_0+2-\gamma_2}{4}\Big) - \psi\Big(\mu + \frac{c_0+2+\gamma_2}{4}\Big) - \gamma_4\sqrt{\beta+4(\mu^2+\mu)} + \gamma_3 \geq 0$$

where :

$$\gamma_1 \triangleq \psi\Big(1 + \frac{c_0+2+\gamma_2}{4}\Big) - \psi\Big(1 + \frac{c_0+2-\gamma_2}{4}\Big)$$

and :

$$\gamma_2 \triangleq \sqrt{(c_0+2)^2 - 8\cdot(S_0-1)}$$

and :

$$\gamma_3 \triangleq \gamma 1 - \frac{\gamma_2}{d}\cdot\Big(8 + \frac{8}{2k} - \frac{d-2k}{|\partial F_0|-1}\Big)$$

and :

$$\gamma_4 \triangleq \frac{8}{2}\cdot\frac{\gamma_2}{k\cdot d} \qquad\qquad \beta \triangleq 2S_0 + 2c_0 - 1$$

then $t_{SUCCESS} = (\mu_{SUCCESS} - 1)\cdot d$

where ψ is the Digamma *function, defined as $\psi(x) = \frac{d}{dx}\ln\Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}$, where $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ (see for example [1]).*

Theorem 5. *Given a contaminated region of properties S_0, $|\partial F_0|$ and $W(F_0)$, spreading every d time steps, then in order to guarantee a successful cleaning of the region before the contamination will be able to spread even once, the minimal number of cleaning agents required is bounded from above as follows :*

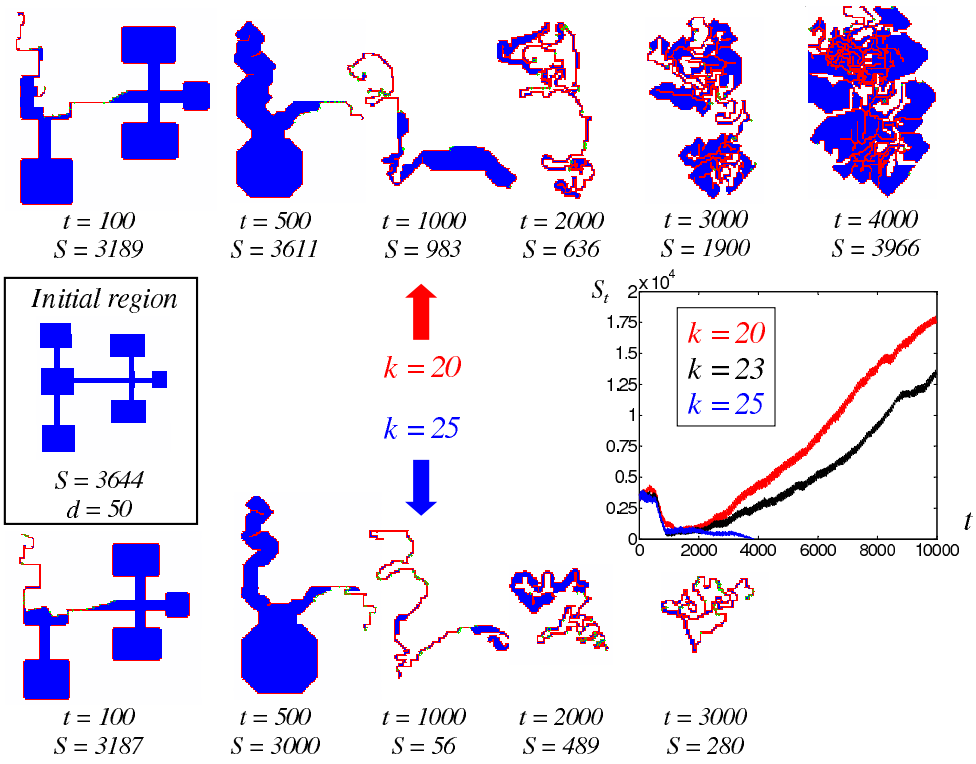
$$(k_1 \leq k \leq k_2) \wedge (k > 0)$$

where :

$$k_{1,2} = 2\left(-|\partial F_0| + 1 + \frac{d}{8}\right) \pm 2\sqrt{\left(|\partial F_0| - 1 - \frac{d}{8}\right)^2 - (|\partial F_0|-1)W(F_0)}$$

Experimental results

Figure demonstrates the change in the geometric properties of the contaminated region as a result of the cleaning process. Exhaustive simulations were performed with the **SWEEP** protocol on various dynamic domains, and compared with the lower and upper bounds. The latter upper bounds are obviously not very tight, and the search for a tighter bound remains a challenge.



References

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Number 55 in Applied Mathematics Series. National Bureau of Standards, 1964.

[2] R. C. Arkin and T. Balch. Cooperative multi agent robotic systems. *Artificial Intelligence and Mobile Robots*, 1998. AAAI.

[3] G. Beni and J. Wang. Theoretical problems for the realization of distributed robotic systems. In *Proc. of 1991 IEEE Internal Conference on Robotics and Automation*, pages 1914–1920, Sacramento, CA, April 1991.

[4] V. Braitenberg. *Vehicles*. MIT Press, 1984.

[5] D. C. Conner, A. Greenfield, P. N. Atkar, A. A. Rizzi, and H. Choset. Paint deposition modeling for trajectory planning on automotive surfaces. *IEEE Transactions on Automation Science and Engineering*, 2(4):381–392, 2005.

[6] D. Henrich. Space efficient region filling in raster graphics. *The Visual Computer*, 10:205–215, 1994. Springer.

[7] E. Pagello, A. D'Angelo, C. Ferrari, R. Polesel, R. Rosati, and A. Speranzon. Emergent behaviors of a robot team performing cooperative tasks. *Advanced Robotics*, 2002.

[8] I. Rekleitisy, V. Lee-Shuey, A. Peng Newz, and H. Choset. Limited communication, multi-robot team based coverage. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004.

[9] S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. *AAAI-94*, pages 426–431, 1994.

[10] I. A. Wagner and A. M. Bruckstein. Cooperative cleaners: A case of distributed ant-robotics. In *Communications, Computation, Control, and Signal Processing: A Tribute to Thomas Kailath*, pages 289–308. Kluwer Academic Publishers, 1997.

Q-learning applied to a stochastic timed automaton with deterministic transitions and clock resets

Gonçalo Neto* Pedro U. Lima*

Abstract

Supervisory Control of Discrete Event Systems and Reinforcement Learning are two frameworks suited for robot control but focusing on different aspects of it. We combine ideas from the two fields and come up with an event based controller that incorporates hard restrictions and adaptation. We apply this method to a Stochastic Timed Automaton, with some simplifications, and show the problem is equivalent to a Semi Markov Decision Process. We derive the optimality equations and Q-learning update rule for the system and show the stochastic approximation theorems that guarantee Q-learning convergence on MDPs can still be applied, under certain conditions on the firing times of events. We present an application example of the type of problems we believe this method is most suited for.

1 Introduction

The modeling of robotic tasks can be addressed in several ways, according to different frameworks. A possible choice is Discrete Event Systems (DES) [4] which rely on the concept of event, rather than time, as the driver of the system dynamics, an adequate abstraction in many setups. The goal of a robot is, most of the times, to exert some control over such system to accomplish a certain task.

For the particular class of DES only concerned about the untimed (or logical) behavior of the system, Supervisory Control (SC), as introduced in [12], is an adequate tool to steer a robotic task modeled to accomplish the specifications. A detailed description of such tools is made in [4, 12].

Another popular choice to address the problems of decision making in a robotic setup is that of Markov Decision Processes (MDP) [9]. This kind of systems provide the basis on which to apply reinforcement learning (RL) techniques [10] that allow the robot to have no knowledge about the environment (apart from the state space in which it lives and the

*Institute for Systems and Robotics, Instituto Superior Técnico, Portugal. E-mails: gneto@isr.ist.utl.pt, pal@isr.ist.utl.pt

actions available), learning to act by experience. Q-learning [11] is one of the most popular algorithms to address the problem of decision making in such a setup.

Although there are points of contact between the two mentioned approaches, usually SC is more concerned about controlling a system whose model is fully known, in order to avoid logical problems. An example is avoiding deadlocks in a system modeled as an automaton, which could correspond to a potentially hazardous situation on a manufacturing plant.

On the other hand, reinforcement learning, when applied without any other support methods, usually ends up being a very time consuming technique, and it is mainly applied to simulation, software agents or know toy-problems (although there are successful cases of practical applications of the algorithms).

Our approach on this work aims for combining the ideas of both SC and RL, in order to build controllers that are mainly specified at the logical level, but leaving room for adaptation and optimization, using learning tools.

2 Background

2.1 Supervisory Control of Discrete Event Systems

A Discrete Event System (DES) can be roughly described as a system which is driven by a sequence of events, rather than time. This can correspond to a inherently discrete system or to one that has a mix of both discrete and continuous dynamics.

In such a system, the strings of events produced are said to represent the behavior of the system. Considering the event set E, the behavior of a system whose dynamics are ruled by the events on that set represents a language L, with strings composed of elements of E.

Modeling such a system using the complete description of the possible behaviors is not only tedious but often impossible. Several formalisms are usually used, that correspond to different degrees of language complexity, to represent the system in a more compact way. In particular, systems that generate regular languages can be represented by a Finite State Automaton (FSA), described as a tuple $G = (X, E, f, \Gamma, x_0, X_M)$ where:

- E represents an event set
- X represents a state space
- $f : X \times E \to X$ is a possibly partial function representing the state transitions.
- $\Gamma(x)$ is the set of enabled events in state x.
- x_0 is the initial state of the system.
- X_M is a set of marked states.

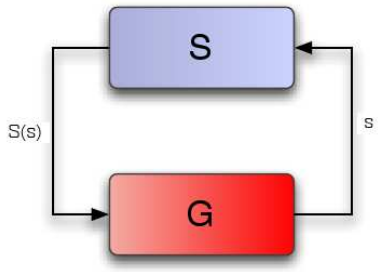


Figure 1: Supervisory Control scheme

In order to control a system of this kind, the event set is divided in a controllable event set E_C and an uncontrollable event set E_{UC} such that $E = E_C \cup E_{UC}$. A supervisor is then defined as a function:

$$S : \mathcal{L}(G) \rightarrow 2^E$$

The supervisor basically chooses a set of events to enable for each string generated by the system. For a system starting in state x_0 and after string s has occurred, the set of enabled events is given by $S(s) \cap \Gamma(f(x_0, s))$ with f being defined recursively for strings from the initial automaton definition.

Figure 1 shows the control scheme for this kind of systems.

It's important to note that the supervisor, in order to be *admissible*, is not allowed to disable uncontrollable events:

$$\forall_{s \in \mathcal{L}(G)} E_{UC} \cap \Gamma(f(x_0, s)) \subseteq S(s)$$

The behavior of the supervised system can also be represented by a language, usually denoted by $L(S/G)$, and in some cases generated by a FSA. We focus particularly on the finite state systems, whose restricted and unrestricted behaviors generate regular languages, and can be represented by FSA.

The controllability theorems for supervisory control systems can be found in [4, 12].

2.2 Q-learning

Markov Decision Processes [1, 6, 2, 9] are, in fact, the foundation for much of the research on agent control. They can be defined as a tuple (X, A, T, R) where:

- A is an action set.
- X is a state space.
- $T : X \times A \times X \longrightarrow [0, 1]$ is a transition function defined as a probability distribution over the states. Hence, we have $T(x, a, x') = Pr\{x_{t+1} = x' | x_t = x, a_t = a\}$. x_{t+1} represents

the state of the process at time $t+1$, x_t the state at time t and a_t the action taken after observing state x_t.

- $R : X \times A \times X \longrightarrow \mathcal{R}$ is a reward function representing the expected value of the next reward, given the current state x and action a and the next state a': $R(x, a, x') = E\{r_{t+1}|x_t = x, a_t = a, x_{t+1} = x'\}$. In this context r_{t+1} represents the immediate payoff of the environment to the agent at time $t + 1$.

A fundamental concept of algorithms for solving MDPs is the *state value function*, which is nothing more than the expected reward (in some reward model) for some state, given the agent is following policy π:

$$V^{\pi}(x) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| x_t = x, \pi\right\} \tag{1}$$

Similarly, the expected reward given the agent takes action a in state x and following policy π could also be defined:

$$Q^{\pi}(x, a) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| x_t = x, a_t = a, \pi\right\} \tag{2}$$

This function is usually known as *Q-function* and the corresponding values as *Q-values*.

From Equation (1) a relation can be derived, which will act as the base of much of the ideas behind dynamic programming and reinforcement learning algorithms to solve MDPs.

$$V^{\pi}(x) = \sum_{a} \pi(x, a) \sum_{y \in X} T(x, a, y) \left[R(x, a, y) + \gamma V^{\pi}(y)\right] \tag{3}$$

The resulting equation, called the *Bellman equation*, has a unique solution for each policy which is the state value function for that policy [6].

It can be shown [9] that there is always at least one deterministic policy whose state values are not dominated by other policies, which leads to the optimal version of the previous equation:

$$V^{*}(x) = \max_{a} \sum_{y \in X} T(x, a, y) \left[R(x, a, y) + \gamma V^{*}(y)\right] \tag{4}$$

or in terms of the Q-values:

$$Q^{*}(x, a) = \sum_{y \in X} T(x, a, y) \left[R(x, a, y) + \gamma \max_{b \in A} Q^{*}(y, b)\right] \tag{5}$$

Solving this equations can be accomplished using dynamic programming but, when there is no access to information about the transition function and the expected value of the reward, a sample based strategy can be used to obtain the state values, using stochastic approximation

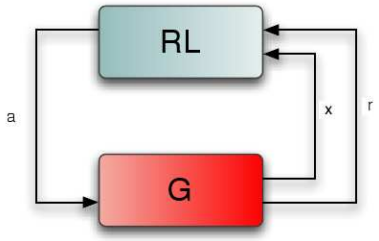


Figure 2: Q-learning control scheme

results. This is the base for reinforcement leaning algorithms, from which Q-learning is a popular example. The update rule for Q learning is:

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha_t\big(r_t + \gamma \max_{b \in A} Q_t(x_{t+1}, b) - Q_t(x_t, a_t)\big) \tag{6}$$

Figure 2 shows the agent/environment loop for reinforcement learning systems.

3 Building the Model

We start by assuming our system can be modeled as a Stochastic Timed Automaton (STA) [5] which is defined as a tuple $(X, E, \Gamma, p, p_0, \mathcal{T})$ where:

- E represents an event set
- X represents a state space
- $\Gamma(x)$ is the set of enabled events in state x
- $p(x', x, e)$ is a transition function defining the probability of reaching state x', starting in state x and after the occurrence of event e. The functions is possibly partial, not defined for events $e' \notin \Gamma(x)$.
- $p_0(x)$ is a probability distribution over X that represents the knowledge about the initial state of the system.
- $\mathcal{T} = \{\mathcal{T}_i : i \in E\}$ is a stochastic clock structure.

Note that the difference between this model and an FSA is the introduction of uncertainty in the transitions and in the time between event occurrences. Nevertheless, the system can still be controlled logically in exactly the same way as an FSA.

As for the deterministic and untimed case, we consider the event set is divided in a set of controllable events E_c and a set of uncontrollable events E_{uc}. We the define a supervisor S.

Considering the goal of the work is, ultimately, to allow for the unrestricted part of the system behavior to evolve to some optimum point, according to some utility function, we need to set the conditions to apply reinforcement learning to it. We make the following assumptions:

Assumption 1. *The initial state of the automaton is univocally determined:*

$$\exists_{x_0 \in X}\, p_0(x_0) = 1$$

Assumption 2. *The outcomes of (state, event) transition are deterministic:*

$$\forall_{x \in X} \forall_{e \in \Gamma(x)} \exists_{x' \in \mathcal{X}}\, p(x', x, e) = 1$$

This way, each transition univocally identifies the state to which the system moves, and we can even replace p_0 for x_0 and p for $f : X \times E \to X$, representing a transition partial function where $x' = f(x, e)$.

Furthermore, an observer is needed to identify such state. The focus on this work is not on observability, so, all events are assumed to be completely observable and the observer is assumed to completely identify the state: $G^{Ob} = (X, E, \Gamma, x_0, f)$.

As for the supervisor previously defined, we also need to have access to a state description in order to apply learning to the supervised system. We make another assumption:

Assumption 3. *The supervisor can be represented by a finite state automaton:*

$$G^{Su} = (X^{Su}, E^{Su}, \Gamma^{Su}, x_0^{Su}, f^{Su})$$

The product of the supervisor with the observer will create another FSA that represents the logic behavior of the supervised system (assuming full observability). Hence, the state space on which the learning algorithm works (we call it X^M) will be a subset of the cartesian product of both automata state spaces: $X^M \subset X^{Ob} \times X^{Su}$.

Actions available to the learning system are associated to events:

$$A_{x^M} = \left\{ \{e\} : e \in \Gamma^{Su \times Ob}(x^M) \cap E_c \right\} \cup \{\emptyset\}$$

The learner either picks one event to enable, representing an action, or chooses to wait until the system itself produces a reaction. We could assume other action definitions, which wouldn't make a difference to the equations derived from this point on. The set of enabled events for choice of action a (with a being picked from A_{x^M}) is denoted as:

$$\Gamma_a(x^M) = \left(\Gamma^{Su \times Ob}(x^M) \cap E_{uc} \right) \cup a$$

The introduction of idling actions is an interesting consequence of a continuous time system – the notion that sometimes waiting for some event to occur might be preferable to choosing something to do.

Finally, for the reinforcement learning methods to work, the system needs to be Markovian w.r.t the decision points and, with a general clock structure associated with the STA, that might not happen. For this reason, we make the following assumption:

Assumption 4. *At each state change, a reset is made to the firing times of every event.*

The expressions that characterize the transition probabilities on this system can be calculated from the model parameters. Particularly, we have:

$$\begin{aligned} H(t, x'|x, a) &= P\left[Y \leq t, X' = x'|X = x, A = a\right] \\ &= \sum_{e \in \Gamma_a(x)} P\left[Y \leq t|E = e\right]\ p(x'|x, e)\, p(e|x, a) \\ &= \sum_{e \in \Gamma_a(x)} P\left[Y_e \leq t\right]\ p(x'|x, e)\, p(e|x, a) \end{aligned}$$

with $H(t, x'|x, a)$ representing a joint probability of moving to a new state, within a certain time frame and Y being the random variable that represents the time of the next event firing and Y_e the firing time of event e.

We can now make use of Assumption 4 and, in this case, $P\left[Y_e \leq t\right]$ can be replaced with the stationary time distribution for event e (defined initially for the STA), $F_{\mathcal{T}_e}(t)$. We obtain:

$$H(t, x'|x, a) = \sum_{e \in \Gamma_a(x)} F_{\mathcal{T}_e}(t)\, p(x'|x, e) \int_0^\infty F_{\mathcal{T}_e}(\tau) dF_{W^x_{a\bar{e}}}(\tau)$$

where

$$F_{W^x_{a\bar{e}}}(t) = 1 - \prod_{\substack{j \in \Gamma_a(x) \\ j \neq e}} \left(1 - F_{\mathcal{T}_j}(t)\right)$$

4 Optimality concepts

With the simplifications made constructing the model, it can be identified with a Semi-Markov Decision Process (SMDP) [7]. We build a reward structure depending on the events enabled at each moment, and the state in which the system is, which can be rewritten as depending on state and action.

For each state-action pair, we have a constant reward rate $c(x, a)$ and a lump reward $k(x, a)$. We use a discounted reward model and follow the derivation of [9] to obtain an

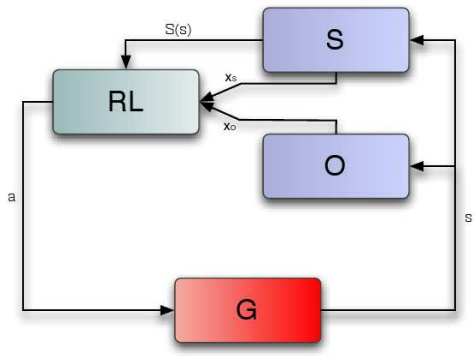


Figure 3: Supervised event-based Q-learning control scheme

optimality equation similar to the Bellman equation for MDPs.

$$Q^*(x,a) = r(x,a) + \sum_{y \in X} M(x,a,y) \max_{b \in A_x} Q^*(y,b)$$

where $M(x,a,y) = \int_0^\infty e^{-\beta\tau} H(d\tau, y|x,a)$ and $r(x,a) = k(x,a) + \int_0^\infty \frac{1}{\beta}(1-e^{-\beta\tau})c(x,a)dF_{W_a^x}(\tau)$ and the constant β is a discount factor. In [9], conditions on the function $H(t,x'|x,a)$ and the rewards are presented that guarantee convergence of the method. These can be obtained from similar conditions on the probability distributions of the events and their rewards.

This optimality equation can be approximated using a stochastic approximation strategy, much like what it was done for the MDP case. We apply the modified Q-learning rule described on [3] for SMDPs:

$$Q_{t+1}(x_t,a_t) = (1-\alpha_t)Q_t(x_t,a_t) + \alpha_t \tilde{Q}_{t+1}(x_t,a_t)$$

with

$$\tilde{Q}_{t+1}(x_t,a_t) = r(x_t,a_t,\tau_t) + e^{-\beta\tau_t} \max_{b \in A_{x_{t+1}}} Q_t(x_{t+1},b)$$

$$r(x_t,a_t,\tau_t) = k(x_t,a_t) + \frac{1-e^{-\beta\tau_t}}{\beta} c(x_t,a_t)$$

where τ_t is the time it took for the state to change from x_t to x_{t+1}.

The control scheme obtained from combining the supervisory control approach with Q-learning for a continuous time system can be seen in Figure 3.

4.1 Convergence notes

Another important assumption on the model, that will ensure the convergence of both the dynamic programming and Q-learning algorithms, is that fact that, in any interval of finite

length, the number of decision epochs to occur will be finite. This is discussed in **??** and can be expressed in the following way:

Assumption 5. *There is a* $\epsilon > 0$ *and a* $\delta > 0$ *such that, every* $x \in X$ *and* $a \in A_x$*:*

$$F(\delta|x,a) < 1 - \epsilon$$

where $F(t|x,a) = \sum_{y \in X} H(t,y|x,a)$.

This condition ensures that the decisions are not done infinitely fast, and will be important to prove Q-learning converges. Furthermore, it can be derived for the actions from similar conditions on the event distributions.

For the proof of the convergence of Q-learning applied to this kind of systems, we follow [8] by using the same convergence theorems for stochastic approximation presented in that work. Particularly, a crucial part of the proof is showing that the operator:

$$(\mathbb{H}q)(x,a) = r(x,a) + \sum_{y \in X} M(x,a,y) \max_{b \in A_x} q(y,b)$$

is a contraction mapping on the sup-norm. We have:

$$\begin{aligned}
&\|\mathbb{H}q_1 - \mathbb{H}q_1\|_\infty = \\
&= \max_{x,a} \left| \sum_{y \in X} M(x,a,y) \left(\max_{b \in A_y} q_1(y,b) - \max_{b' \in A_y} q_2(y,b') \right) \right| \leq \\
&\leq \max_{x,a} \sum_{y \in X} M(x,a,y) \cdot \max_{z,b} |q_1(z,b) - q_2(z,b)| = \\
&= \max_{x,a} \sum_{y \in X} M(x,a,y) \cdot \|q_1 - q_2\|_\infty
\end{aligned}$$

It remains to show that $\max_{x,a} \sum_{y \in X} M(x,a,y)$ is strictly bounded by 1. This condition is met because of the Assumption 5, in the following way:

$$\begin{aligned}
&\max_{x,a} \sum_{y \in X} M(x,a,y) = \\
&= \max_{x,a} \sum_{y \in X} \int_0^\infty e^{-\beta t} H(dt,y|x,a) = \\
&= \max_{x,a} \int_0^\infty e^{-\beta t} F(dt|x,a) = \\
&= \max_{x,a} \int_0^\delta e^{-\beta t} F(dt|x,a) + \int_\delta^\infty e^{-\beta t} F(dt|x,a) \leq \\
&\leq \max_{x,a} F(\delta|x,a) + e^{-\beta\delta}(1 - F(\delta|x,a))
\end{aligned}$$

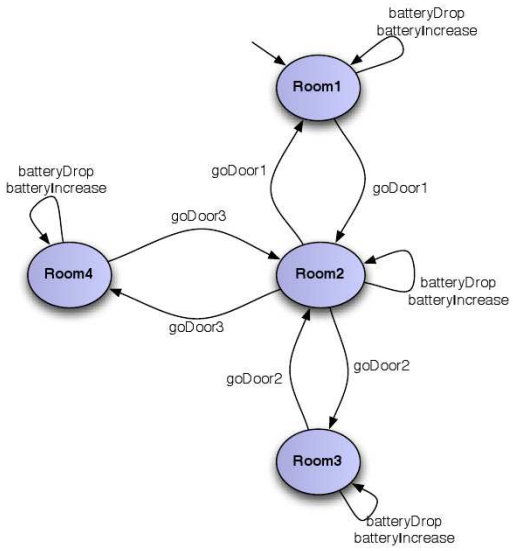


Figure 4: Navigation automaton

and from Assumption 5

$$\max_{x,a} \sum_{y \in X} M(x,a,y) \leq$$
$$\leq e^{-\beta\delta} + (1-\epsilon)(1-e^{-\beta\delta}) < 1$$

If we set $\gamma^* = e^{-\beta\delta} + (1-\epsilon)(1-e^{-\beta\delta})$, we have:

$$\|\mathbb{H}q_1 - \mathbb{H}q_1\|_\infty \leq \gamma^* \|q_1 - q_2\|_\infty$$

with $0 \leq \gamma^* < 1$, which proves that operator $\mathbb{H}$ is, in fact a contraction mapping of the sup-norm.

5 Application example

We'll start by assuming we have a mobile robot in a simple navigation problem: there are 4 rooms connected with 3 doors and the robot has to learn to navigate in an optimal way in them. The exact nature of the task will depend on the rewards assigned to each state and action. Figure 4 represents the automaton that modes the navigation environment.

Note that the navigation is performed using specialized navigation functions (represented by events of type *goDoorX*). This is a bit different from what it is usually done in reinforcement learning problems (N,S,E,W type of actions) and shows the typical higher level decision making task that we think our approach is suited for. Lets call the navigation automaton G_n.

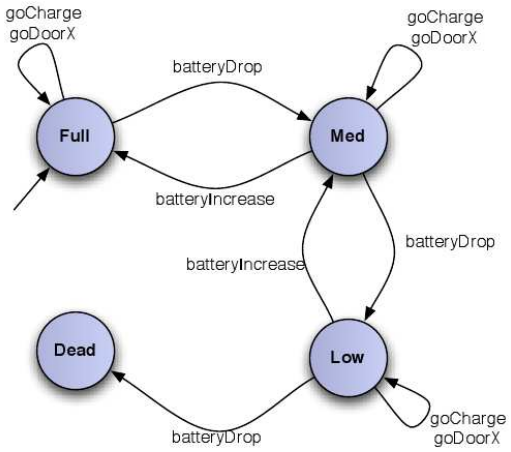


Figure 5: Battery automaton

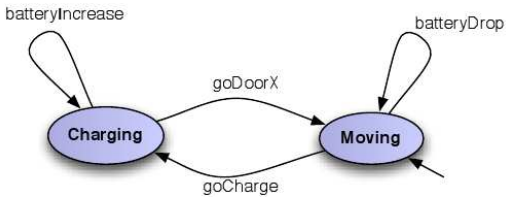


Figure 6: Charger automaton

Parallel to the navigation automaton, there's a battery meter running that reacts to drops or increases in the battery level and monitors the battery state accordingly. We'll call this automaton G_b and it is graphically represented in Figure 5.

An important note about the battery meter is that, besides modeling the state of the battery, it also models the fact that, without battery, the robot won't be able to do anything. This is represented by having disabled all possible events once the state *Dead* is reached. Clearly, this will be something to avoid when acting in this environment, as it will function as a dead lock and force the robot to stay there indefinitely.

The way the robot gets its charge back is represented by a third automaton, G_c, that is related to another navigation function, *goCharge*, that steers the robot to the nearest charger and keeps him there until some other navigation function is chosen. We assume that in every room there is a charger (for example a wall dock where the robot can plug to).

Figure 6 represents the third automaton.

The product of this 3 automata, with the additional association of a stochastic clock

structure to the events, will give us the STA with deterministic transitions used as a base for the application of the method. We have:

- $$E = \{goDoor1, goDoor2, goDoor3, \ldots$$
  $$\ldots goCharge, batteryDrop, batteryIncrease\}$$
  represents the event set.
- $X_G \subseteq \{Room1, Room2, Room3, Room4\} \times \{Charging, Moving\} \times \{Full, Medium, Low, Dead\}$ represents the state space. The actual state space is not exactly equal to the full cartesian product of all the 3 automata state spaces but a bit smaller.
- $\Gamma(x) = \Gamma_n(x_n) \cap \Gamma_c(x_c) \cap \Gamma_b(x_b)$ with $x = (x_n, x_c, x_b)$.
- The transition function is defined as:
  $$f(x, e) = \begin{cases} (f_n(x_n, e), f_c(x_c, e), f_b(x_b, e)) & e \in \Gamma(x) \\ \text{undefined} & \text{otherwise.} \end{cases}$$
  with $x = (x_n, x_c, x_b)$.
- $x_0 = (x_{0n}, x_{0c}, x_{0b})$ is the initial state.
- $\mathcal{T} = \{\mathcal{T}_i : i \in E\}$ is a stochastic clock structure.

The idea is now, before applying reinforcement learning, to introduce some safety constraints in the system. In a typical reinforcement learning application, we would assign a negative reward to the *Dead* state, and let the robot learn it shouldn't allow himself to reach that state.

However, the problem is that, once the state has been reached, the robot itself will not be able to continue to function properly. The idea of combining supervisory control is, exactly, to be able to introduce some *a priori* knowledge, that will allow us to restrict the problem to one that satisfies the safety restrictions (or any other kind of restrictions that can be modeled using supervisory control).

To accomplish that, we'll start by noting that not all events are controllable. We have $E_c = \{goDoor1, goDoor2, goDoor3, goCharge\}$ and $E_{uc} = \{batteryDrop, batteryIncrease\}$. Depending on the time distribution for each of these events, we may or may not be able to fully guarantee the system won't reach the deadlock. However, we can force the robot to go to the next charger as soon as it reached the medium state. That is accomplished by the use of the supervisor represented in Figure 7.

The **Idling** event is not an actual event but a way of representing the supervisor will allow idling actions in that state.

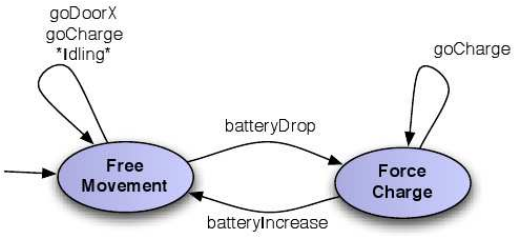


Figure 7: Supervisor automaton

The system has a good probability of never reaching the *Dead* state if the event *goCharge* has higher density at lower time than the event *batteryDrop*, which makes sense since the battery is supposed to last more per drop than the completion of one navigation primitive.

The actions available to the learning system are now defined in terms of the events in each state. For example:

- In state $(Room1, Full, Moving, FreeMovement)$ the supervisor doesn't restrict the system behavior. So, considering we identified each action with a controllable event, the action set would be $A = \{\{goDoor1, batteryDrop\}, \{goCharge, batteryDrop\}, \{batteryDrop\}\}$. Note how each action is identified with a set that has one or none controllable events plus all the uncontrollable ones – these are the events that can happen once an action is picked. Furthermore, important so note we included an idling action because there actually are uncontrollable events. This wouldn't be possible if there were no uncontrollable events because it would force the system to a lock.

- In state $(Room1, Medium, Moving, ForceCharge)$ the actions available would be the same but the supervisor caught a *batteryDrop* event and, for that reason, will only allow the charging action. So: $A = \{\{goCharge, batteryDrop\}\}$

With all this defined we could now apply the modified Q-learning update rule for SMDPs presented previously and reach the optimal behavior for the robot, within the security specifications.

6 Conclusions

The main focus of this work was to combine the supervisory control approach with the reinforcement learning approach to the modeling of robotic tasks. We show we can apply a modified Q-learning rule with guarantees of convergence in a stochastic timed fully observable system, with a logical supervisor, under some simplifications. We also derive the optimality equations for such system, which could be used to obtain dynamic programming algorithms.

We present a toy problem that illustrates the kind of situations we think this approach is suited for in a robotics setup: higher level decision sequential decision making with security restrictions.

As future work, we plan to extend this approach to setups with partial observability in 3 forms:

- Unobservable events: $E_o \neq \emptyset$
- Incomplete state observer: $G^{Ob} \neq (X, E, \Gamma, x_0, f)$
- Non-deterministic state-event transitions: $p(x'|x, e) \in [0, 1]$

Acknowledgements

This work was partially supported by Fundação para a Ciência e Tecnologia, through grant SFRH / BD / 13950 / 2003 and ISR / IST pluriannual funding, from the POS_Conhecimento Program, that includes FEDER funds.

References

[1] Richard Ernest Bellman. *Dynamic Programming*. Princeton Press, 1957.

[2] Dimitri P. Bertsekas. Distributed asynchronous computation of fixed points. *Mathematical Programming*, 27:107–120, 1983.

[3] Steven J. Bradtke and Michael O. Duff. Reinforcement learning methods for continuous-time markov decision problems. In *Advances in Neural Information Processing Systems*, 1994.

[4] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, Boston, 1999.

[5] Peter W. Glynn. A GSMP formalism for discrete event systems. In *Proceedings of the IEEE*, volume 77, pages 14–23, 1989.

[6] Richard Author Howard. *Dynamic Programming and Markov Decision Processes*. MIT Press, 1960.

[7] Richard Author Howard. Semi-markovian decision processes. In *Proceedings International Statistical Institute*, 1963.

[8] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. In *Neural Computation*, volume 6, 1994.

[9] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[10] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. MIT Press, 1998.

[11] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[12] W.M. Wonham. Supervisory control of discrete-event systems. Technical report, Systems Control Group, University of Toronto, 1997.

Algorithmic approach for the sampling of six degrees of freedom information using floating 2-D coordinate frame

Milan Kvasnica*

Abstract

The objective of this paper is an algorithmic approach of optoelectronic system for the six degrees of freedom (DoF) measurements in robotics, human-machine interface, assistive technologies and for the sampling of static and dynamic properties of materials, engineering constructions and machines and for the control operation in space. Described sensory system is based on the sampling and information processing used in the conversion of a 2-D CCD array image into three axial shiftings and three angular displacement values. The algorithm for the computation of three axial shiftings and three angular displacement is based on inverse transformation of final trapezoidal light spots position, related to the original square light spots position on the 2-D CCD array.

1 Introduction

The explanation of the six DoF sampling is introduced on the force-torque transducer. Laser diodes 1 emit the light rays 2 creating the edges of a pyramid intersecting the plane of the 2-D CCD array, here alternatively the focusing screen 8 with light spots 3. The unique light spots configuration changes under axial shifting and angular displacements between the inner flange 5 and the outer flange 6 connected by means of elastic deformable medium 7. An alternatively inserted optical member 9 (for the magnification of micro-movement, or the reduction of macro-movement) projects the light spots configuration from the focusing screen onto the 2-D CCD array 4. Four light rays simplify and enhance the accuracy of the algorithms for the evaluation of the six-DoF information. The algorithms for the computation of three axial shiftings and three radial displacements values is based on the inverse transformation of the final trapezoidal position of four light spots related to the original square light spots

*Tomas Bata University in Zlin, Faculty of Applied Informatics, Nad Stranemi 4511, CZ-76005 Zlin, Czech Republic, E-mail: kvasnica@fai.utb.cz. The support from the grant Vyzkumne zamery MSM 7088352102 "Modelovani a rizeni zpracovatelskych procesu, prirodnich a syntetickych polymeru" is gratefully acknowledged.

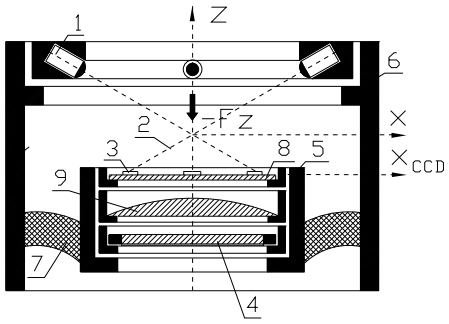


Figure 1: The six component force-torque sensor based on one 2-D CCD array with an acting force component -Fz

position in the plane floating coordinate system x_{CCD}, y_{CCD} on the 2-D CCD array. This algorithm determines the relative location and orientation of a floating 2-D coordinate system against a fixed 3-D coordinate system corresponding to the apex of the pyramid shape, or contrary. The information about three axial shiftings and three angular displacements is sampled and converted according to a calibration matrix to acting forces Fx, Fy, Fz and torques Mx, My, Mz.

2 Direct determination of normal vector components

Let us choose the rectangular coordinate frame x, y, z, hereafter called the pyramid coordinate frame, according to Figure 2. The vertex $P(0;0;0)$ of the pyramid is located in the beginning of the coordinate frame x, y, z. The basis of the pyramid (the plane κ) is created by means of the 2-D CCD array. The basic position of the 2-D CCD array is perpendicular and cutting the z coordinate at the distance $-q_0$ from the vertex P. Light beams create in their basic position on the 2-D CCD array the light spots of a square shape, designated in the coordinate frame x_{CCD}, y_{CCD}

$$\underline{A}^0_{CCD}(\underline{a}_{01};\underline{a}_{02}),\ \underline{B}^0_{CCD}(\underline{b}_{01};\underline{b}_{02}),\ \underline{C}^0_{CCD}(\underline{c}_{01};\underline{c}_{02}),\ \underline{D}^0_{CCD}(\underline{d}_{01};\underline{d}_{02}),\ \underline{S}^0_{CCD}(0;0)$$

and designated in the pyramid coordinate frame x, y, z

$$A^0_p(a_1;a_2;-q_0),\ B^0_p(b_1;b_2;-q_0),\ C^0_p(c_1;c_2;-q_0),\ D^0_p(d_1;d_2;-q_0), S^0_p(0;0) \quad \equiv (0;0;-q_0).$$

Let us designate the positions of the light spots in the CCD coordinate frame in the plane κ at mutual changes of the position k between the coordinate frames x_{CCD}, y_{CCD} and x, y, z

$$\underline{A}^k_{CCD}(\underline{a}_1;\underline{a}_2),\ \underline{B}^k_{CCD}(\underline{b}_1;\underline{b}_2),\ \underline{C}^k_{CCD}(\underline{c}_1;\underline{c}_2),\ \underline{D}^k_{CCD}(\underline{d}_1;\underline{d}_2),\ \underline{S}_0(0;0)$$

and in the pyramid coordinate frame x, y, z

$$A^k_p(a_1;a_2;a_3),\ B^k_p(b_1;b_2;b_3),\ C^k_p(c_1;c_2;c_3),\ D^k_p(d_1;d_2;d_3),\ S^k_p(x_0;y_0;z_0).$$

The angle σ of the light beams, passing through opposite edges of the pyramid with crossing point P, facilitating the proportionality of the shiftings in the direction of the axis z with the shiftings in the direction of the axes x, y is equal $\arctan(\frac{1}{2})$. Let us suppose, that any sequence of the force acting caused the change of the mutual position between the pyramid coordinate frame x, y, z of the light beams and the 2-D CCD array coordinate frame x_{CCD}, y_{CCD}. It changed the position of the center $\underline{S}^0_{CCD}(0;0)$ in the directions of the coordinates x, y, z of the pyramid coordinate frame x, y, z into a new position $S_p(x_0; y_0; z_0)$ determining axial shiftings

$$x = x_0, \quad y = y_0, \quad z = z_0$$

from the beginning P, or the shiftings $x_0, y_0, z_0 - q_0$ from basic position. Let us suppose, that any sequence of the torque acting caused angular displacements of the intersection of the plane κ (of the 2-D CCD array) with the cutting plane y, z by the angle Θ and with the cutting plane x, z by the angle Θ, and in respect to the plane x, y by the angle Ω around the axis z. Please take regard, that the angles Φ, Θ are situated in cutting planes. Here is analyzed a final (statical) position caused by any sequence of the roll, pitch, yaw, characterized by the standardized normal vector $\mathbf{n}^k_{CCD}$

$$\mathbf{n}^k_{CCD} = \begin{bmatrix} \mathbf{n}^k_x \\ \mathbf{n}^k_y \\ \mathbf{n}^k_z \end{bmatrix}$$

of the plane κ in position k. Following relationships enables to determine the components of the vector $\mathbf{n}^k_{CCD}$

$$\mathbf{n}_x = \frac{q_0}{u}\frac{d_1-1}{d_1+1}\mathbf{n}_z$$

$$\mathbf{n}_y = \frac{q_0}{u}\frac{d_2-1}{d_2+1}\mathbf{n}_z$$

$$\mathbf{n}_z = \frac{(d_1+1)(d_2+1)}{\sqrt{\frac{q_0^2}{u^2}[(d_1-1)^2(d_2+1)^2+(d_1+1)^2(d_2-1)^2]+(d_1+1)^2(d_2+1)^2}}.$$

The angle δ which contains the planes κ and τ and the angle μ which contains the intersection of the plain κ and τ with the x axis is possible to eliminate by following relationship

$$\mathbf{n}^k_{CCD} = \begin{bmatrix} \mathbf{n}_x \\ \mathbf{n}_y \\ \mathbf{n}_z \end{bmatrix} = \begin{bmatrix} \mathbf{i} & \sin\delta & \cos\mu_x \\ \mathbf{j} & \sin\delta & \sin\mu_x \\ \mathbf{k} & \cos\delta & \end{bmatrix}.$$

The determination of normal vector components using angular displacements Φ, Θ in cutting planes y, z of the 2D CCD is expressed in following relationships

$$\Phi = \arctan(\tan\delta \sin\mu)$$

$$\Theta = \arctan(\tan\delta\cos\mu)$$

$$\delta = \arctan\sqrt{(\tan^2\Phi + \tan^2\Theta)}$$

$$\mu = \arctan\frac{\tan\Theta}{\tan\Phi}.$$

3 Dilatation, constriction and rotation matrix and angular displacement Ω

The final position of the normal vector $\mathbf{n}^k_{CCD}$ caused by three axial and three angular displacements is possible to substitute by the pitch δ of the plane κ of the 2-D CCD array around the intersection line $y_{\kappa\tau}$ and the roll μ of the intersection line around the axis z, see Figure 2. This final position is possible to decompose into three elementary motion: The rotation around the axis z by the angle $-\mu$

$$\mathbf{Rot}(z;-\mu) = \begin{bmatrix} \cos\mu & \sin\mu & 0 \\ -\sin\mu & \cos\mu & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

The pitch around the axis $y_{\kappa\tau}$ by the angle δ

$$\mathbf{Rot}(y_{\kappa\tau};\delta) = \begin{bmatrix} \cos\delta & 0 & \sin\delta \\ 0 & 1 & 0 \\ -\sin\delta & 0 & \cos\delta \end{bmatrix};$$

The rotation around the axis z by the angle μ

$$\mathbf{Rot}(z;\mu) = \mathbf{Rot}(z;-\mu)^T = \begin{bmatrix} \cos\mu & -\sin\mu & 0 \\ \sin\mu & \cos\mu & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The product of transformation matrixes with the coordinates of general point ${}^k_p\mathbf{X}_p$ causes the rotation of this point around the axis $y_{\kappa\tau}$ into a new position ${}^{k+1}_p\mathbf{X}_p$

$${}^{k+1}_p\mathbf{X}_p = \mathbf{Rot}(z;-\mu)\cdot\mathbf{Rot}(y_{\kappa\tau};\delta)\cdot\mathbf{Rot}(z;\mu)\cdot{}^k_p\mathbf{X}_p = \mathbf{Rot}(\mu;\delta)\cdot{}^k_p\mathbf{X}_p,$$

where

$$\mathbf{Rot}(\mu;\delta) = \begin{bmatrix} \frac{1+n_z-n_x^2}{1+n_z} & -\frac{n_x n_y}{1+n_z} & n_x \\ -\frac{n_x n_y}{1+n_z} & \frac{1+n_z-n_y^2}{1+n_z} & n_y \\ -n_x & -n_y & n_z \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}.$$

The Monge's projection enables to convert the 3-D operation in space to 2-D operation in plane. The transformation of the plane κ points from the pyramid coordinate frame into the coordinate frame κ, τ is given by the dilatation matrix

$$\mathbf{Dil}(\delta;\mu) = \mathbf{Rot}(y_{\kappa\tau};\delta)\cdot\mathbf{Rot}(z;-\mu) = \begin{bmatrix} \frac{\cos(\mu)}{\cos(\delta)} & \frac{\sin(\mu)}{\cos(\delta)} \\ -\sin(\mu) & \cos(\mu) \end{bmatrix}.$$

Following transformation into the pyramid coordinate frame is given by the rotation around the axis z by the angle μ

$$\mathbf{Rot}(z;\mu) = \begin{bmatrix} \cos\mu & -\sin\mu \\ \sin\mu & \cos\mu \end{bmatrix} = \mathbf{Rot}(z;-\mu)^T.$$

Then the relationship $\mathbf{Rot}(\mu;\delta)$ is simplified by the conversion into the 2-D operation by means of the dilatation matrix expressed in the normal vector components

$$\mathbf{Rot}(\mu;\delta) = \mathbf{Rot}(z;\mu)\cdot\mathbf{Dil}(\delta;\mu) =$$

$$= \begin{bmatrix} \frac{\cos^2\mu}{\cos\delta} + \sin^2\mu & \frac{\sin\mu\cos\mu}{\cos\delta} - \sin\mu\cos\mu \\ \frac{\sin\mu\cos\mu}{\cos\delta} - \sin\mu\cos\mu & \frac{\sin^2\mu}{\cos\delta} - \cos^2\mu \end{bmatrix} = \begin{bmatrix} \frac{n_x^2 - n_y^2 n_z}{n_z(1-n_z^2)} & \frac{n_x n_y}{n_z(1+n_z)} \\ \frac{n_x n_y}{n_z(1+n_z)} & \frac{n_y^2 + n_x^2 n_z}{n_z(1-n_z^2)} \end{bmatrix}.$$

The transformation of the plane τ points from the pyramid coordinate frame into the coordinate frame κ, τ is given by the constriction matrix

$$\mathbf{Con}(-\delta;\mu) = \mathbf{Rot}(y_{\kappa\tau};-\delta)\cdot\mathbf{Rot}(z;-\mu) = \begin{bmatrix} \cos(\mu)\cos(\delta) & \sin(\mu)\cos(\delta) \\ -\sin(\mu) & \cos(\mu) \end{bmatrix}.$$

Following transformation into the pyramid coordinate frame is given by rotation around the axis z by the angle μ according to relationship $\mathbf{Rot}(z;\mu)$. Then the relationship $\mathbf{Rot}(\mu;\delta)$ is simplified by the conversion into the 2-D operation by means of the dilatation matrix expressed in the normal vector components

$$\mathbf{Rot}(\mu;-\delta) = \mathbf{Rot}(z;\mu)\cdot\mathbf{Con}(-\delta;\mu) =$$

$$= \begin{bmatrix} \cos^2\mu\cos\delta + \sin^2\mu & \sin\mu\cos\mu\cos\delta - \sin\mu\cos\mu \\ \sin\mu\cos\mu\cos\delta - \sin\mu\cos\mu & \sin^2\mu\cos\delta + \cos^2\mu \end{bmatrix} = \begin{bmatrix} \frac{n_x^2 n_z + n_y^2}{1-n_z^2} & \frac{-n_x n_y}{1+n_z} \\ \frac{-n_x n_y}{1+n_z} & \frac{n_y^2 n_z + n_x^2}{1-n_z^2} \end{bmatrix}.$$

Let us analyze the position of the normal vector $\mathbf{n}_{CCD}^k$ and diagonals A_p^k, S_p, B_p^k and C_p^k, S_p, D_p^k in 3-D space. Diagonal A_p^k, S_p, B_p^k belongs to the plane x, z and diagonal C_p^k, S_p, D_p^k belongs to the plane y, z. Both diagonals belong to the plane κ of the 2-D CCD array. Normal vector $\mathbf{n}_{CCD}^k$ is perpendicular to both diagonals and to axes x_{CCD}, y_{CCD}. Then the the direction of

the straight line passing through the points A_p^k, S_p, B_p^k is given by the cross product of the axis $\mathbf{y} = [0; 1; 0]^T$ and normal vector $\mathbf{n}_{CCD}^k$

$$\frac{\mathbf{y} \times \mathbf{n}_{CCD}^k}{|\mathbf{y} \times \mathbf{n}_{CCD}^k|}.$$

The straight line passing through the points A_p^k, S_p, B_p^k include an angle Ω with the axis x_{CCD} in the plane of the 2-D CCD array in the pyramid coordinate frame. Let us express this straight line slope in the coordinates of the pyramid frame

$$\underline{k}_{AB} = \tan(\underline{\Omega}_{AB}) = \frac{y_B - y_A}{x_B - x_A}. = \frac{\sin \underline{\Omega}_{AB}}{\cos \underline{\Omega}_{AB}} = \frac{\cos(\frac{\pi}{2} - \underline{\Omega}_{AB})}{\cos \underline{\Omega}_{AB}} = \frac{\mathbf{y}_{CCD}^k \cdot (\mathbf{y} \times \mathbf{n}_{CCD}^k)}{\mathbf{x}_{CCD}^k \cdot (\mathbf{y} \times \mathbf{n}_{CCD}^k)},$$

where axes of the 2-D CCD array $\mathbf{x}_{CCD}^k, \mathbf{y}_{CCD}^k$ are expressed in the pyramid coordinate frame, normal vector $\mathbf{n}_{CCD}^k = [\mathbf{n}_x; \mathbf{n}_y; \mathbf{n}_z]^T$ and the cross product is

$$\mathbf{y} \times \mathbf{n}_{CCD}^k = [\mathbf{n}_z; 0; -\mathbf{n}_x]^T.$$

The angular displacement Ω of the axes $\mathbf{x}_{CCD}, \mathbf{y}_{CCD}$ is given by the relationships

$$\mathbf{x}_{CCD}^{k-1} = \mathbf{Rot}(\Omega) \cdot \mathbf{x}_{CCD}^{k-2} = \begin{bmatrix} \cos \Omega \\ \sin \Omega \\ 0 \end{bmatrix}$$

$$\mathbf{y}_{CCD}^{k-1} = \mathbf{Rot}(\Omega) \cdot \mathbf{y}_{CCD}^{k-2} = \begin{bmatrix} -\sin \Omega \\ \cos \Omega \\ 0 \end{bmatrix}$$

where

$$\mathbf{Rot}(\Omega) = \begin{bmatrix} \cos \Omega & -\sin \Omega & 0 \\ \sin \Omega & \cos \Omega & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and $\mathbf{x}_{CCD} = [1; 0; 0]^T$; $\mathbf{y}_{CCD} = [0; 1; 0]^T$.

$$\mathbf{x}_{CCD}^k = \mathbf{Rot}(\mu; \delta) \cdot \mathbf{x}_{CCD}^{k-1} = \begin{bmatrix} H_{11} \cos \Omega + H_{12} \sin \Omega \\ H_{21} \cos \Omega + H_{22} \sin \Omega \\ H_{31} \cos \Omega + H_{32} \sin \Omega \end{bmatrix}$$

$$\mathbf{y}_{CCD}^k = \mathbf{Rot}(\mu; \delta) \cdot \mathbf{y}_{CCD}^{k-1} = \begin{bmatrix} -H_{11} \sin \Omega + H_{12} \cos \Omega \\ -H_{21} \sin \Omega + H_{22} \cos \Omega \\ -H_{31} \sin \Omega + H_{32} \cos \Omega \end{bmatrix}$$

Let us in following steps to prepare the numerator

$$\mathbf{y}_{CCD}^{k} \cdot (\mathbf{y} \times \mathbf{n}_{CCD}^{k}) = (H_{13}H_{31} - H_{11}H_{33}) \sin\Omega + (H_{12}H_{33} - H_{13}H_{32}) \cos\Omega$$

and the denominator

$$\mathbf{x}_{CCD}^{k} \cdot (\mathbf{y} \times \mathbf{n}_{CCD}^{k}) = (H_{11}H_{33} - H_{13}H_{31}) \cos\Omega + (H_{12}H_{33} - H_{13}H_{32}) \sin\Omega$$

and after the substitution

$$\tan(\underline{\Omega}_{AB}) = \frac{(H_{12}H_{33} - H_{13}H_{32})\cos\Omega - (H_{11}H_{33} - H_{13}H_{31})\sin\Omega}{(H_{11}H_{33} - H_{13}H_{31})\cos\Omega - (H_{12}H_{33} - H_{13}H_{32})\sin\Omega}$$

$$\tan(\underline{\Omega}_{AB}) = \frac{G - \tan(\Omega)}{1 + G\tan(\Omega)}.$$

Let us define dividing ratio expressed in the normal vector components

$$G = \frac{H_{12}H_{33} - H_{13}H_{32}}{H_{11}H_{33} - H_{13}H_{31}} = \frac{n_x n_y}{n_x^2 + n_z(1+n_z)}.$$

And after elimination is the angle Ω

$$\Omega = \arctan \frac{G - \tan(\underline{\Omega}_{AB})}{1 + G\tan(\underline{\Omega}_{AB})}$$

The transformation of the coordinates $\underline{x}, \underline{y}$ of the square CCD array into the pyramid coordinate frame x, y is described by the equations, where $\underline{S}_0(x_{S0}; y_{S0})$ are the coordinates of the beginning of the coordinate frame x_{CCD}, y_{CCD}, designated in the pyramid coordinate frame x, y and Ω is the angle between these coordinate frames

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\Omega & -\sin\Omega \\ \sin\Omega & \cos\Omega \end{bmatrix} \cdot \begin{bmatrix} \underline{x}_{CCD} \\ \underline{y}_{CCD} \end{bmatrix} + \begin{bmatrix} x_{S0} \\ y_{S0} \end{bmatrix}.$$

The transformation of the pyramid coordinates x, y into the x_{CCD}, y_{CCD} coordinate frame where $S(\underline{s}_1; \underline{s}_2)$ are the coordinate of the beginning of the pyramid coordinate frame x, y designated in the coordinate frame x_{CCD}, y_{CCD}

$$\begin{bmatrix} \underline{x}_{CCD} \\ \underline{y}_{CCD} \end{bmatrix} = \begin{bmatrix} \cos\Omega & \sin\Omega \\ -\sin\Omega & \cos\Omega \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \underline{s}_1 \\ \underline{s}_2 \end{bmatrix}.$$

The beginning $\underline{S}_0(x_{s0}; y_{s0})$ of the coordinate frame of the 2-D CCD array designated in the pyramid coordinate frame x, y by means of the turn through the angle μ determines the point $S_0(x_{0\mu}; y_{0\mu})$. The following pitch from the plane τ through the angle δ into the plane κ determines the point $\underline{S}_0(x_{0\mu\delta}; y_{0\mu})$, by means of the constriction matrix

$$\begin{bmatrix} x_{\mu\delta} \\ y_{\mu} \end{bmatrix} = \begin{bmatrix} \cos\mu\cos\delta & \sin\mu\cos\delta \\ -\sin\mu & \cos\mu \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}.$$

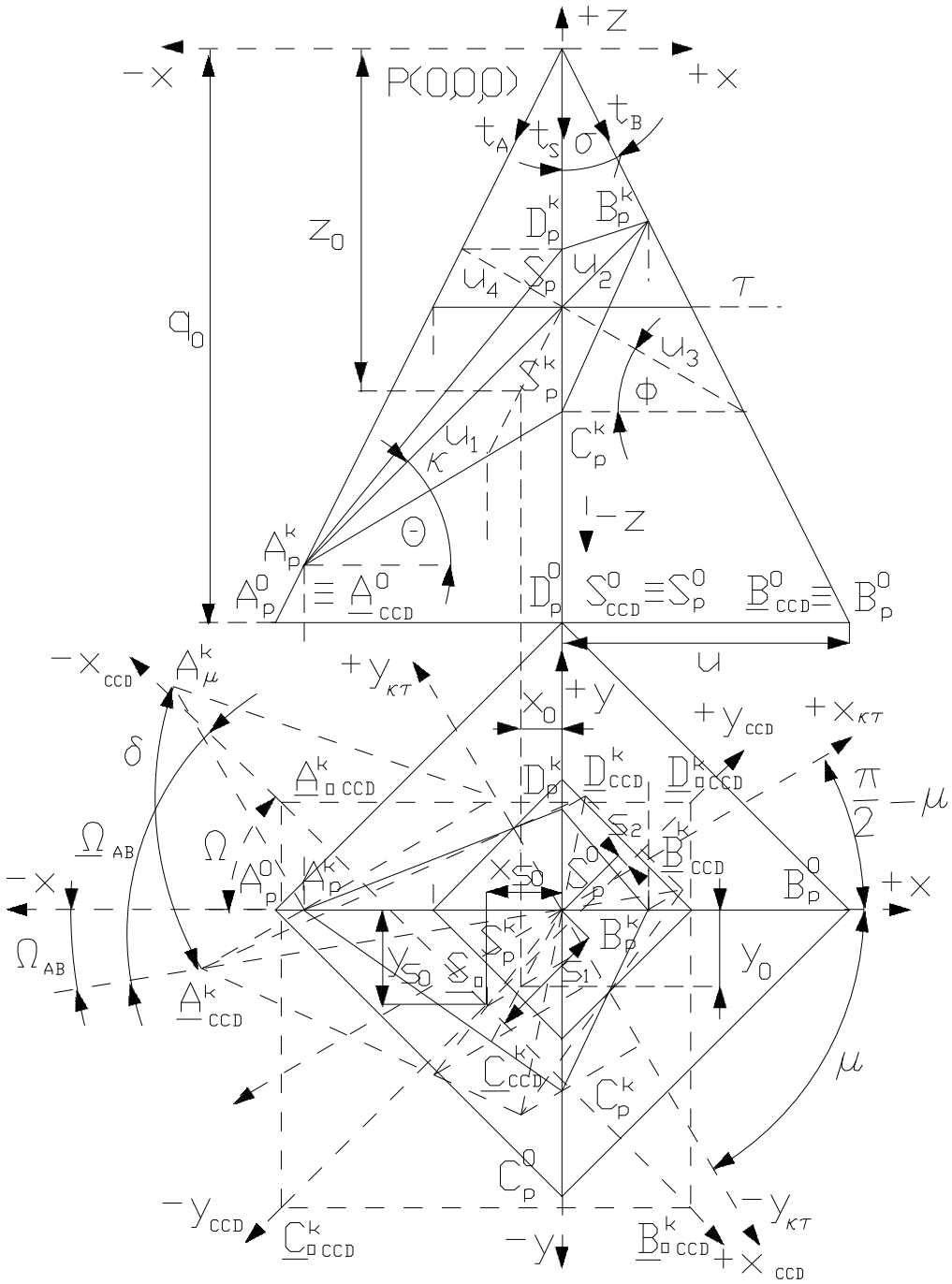


Figure 2: Geometric approach for the sampling of six component information using 2-D CCD array and four light rays in Monge's projection.

The inverse transformation of the point $\underline{S}_0^k(x_{0\mu\delta}; y_{0\mu})$ from the $x_{\kappa\tau}, y_{\kappa\tau}$ coordinate frame into the pyramid coordinate frame x, y in the plane τ is determined by means of the relationship

$$\begin{bmatrix} x_{\frac{\mu}{\delta}} \\ y_\mu \end{bmatrix} = \begin{bmatrix} \frac{\cos\mu}{\cos\delta} & \frac{\sin\mu}{\cos\delta} \\ -\sin\mu & \cos\mu \end{bmatrix} \cdot \begin{bmatrix} x^k \\ y^k \end{bmatrix}$$

with the dilatation matrix which determines the coordinates of the point $S_p^k(x_0; y_0)$. The coordinates of the point $S_p^k(x_0; y_0)$ (as the projection from the plane κ into the plane view) determine the position of the midpoint of the inner flange of the six component transducer, see Figure 3.1 and Figure 3.2. The intersection of the straight line passing parallel with the axis z against the plane κ of the 2-D CCD array into the point x_0, y_0 determines the coordinate z_0. The coordinate z_0 contains the information about the magnitude and the direction of axial shifting in the z direction. The coordinates x_0, y_0 contains the information about the magnitude and the direction of the axial shiftings in the x and y direction

$$x = x_0 \qquad y = y_0 \qquad z = z_0 - q_0.$$

References

[1] E. Dobrocka. Geometrical principles of the monolithic X-ray magnifier. *Journal Appl. Cryst.*, 24, 1991.

[2] G. Hirzinger, J. Dietrich, J. Gombert, J. Heindl, K. Landzettel, and J. Schott. The sensory and telerobotic aspects of space robot technology experiment ROTEX. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Toulouse, Labege, France, 1992.

[3] M. Kvasnica. Intelligent sensors for the control of autonomous vehicles. In *Proceedings of the 6th International Conference and Exposition on Engineering, Construction and Operation in Space and on Robotics for Challenging Environments Space and Robotics'1998*, Albuquerque, New Mexico, USA, 1998.

[4] M. Kvasnica. Measurement in engineering construction and control operations in space. In *ASCE Multiconference on Engineering, Construction, Operations, and Business in Space and on Robotics for Challenging Situations and Environment Space and Robotics'2000*, Albuquerque, New Mexico, USA, 2000.

[5] M. Kvasnica. Improvement of positioning accuracy in multi-pod parallel structures. In *ASCE Multiconference on Engineering, Construction, Operations, and Business in Space and on Robotics for Challenging Situations and Environment Space and Robotics'2002*, Albuquerque, New Mexico, USA, 2002.

Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators

Matthew Howard* Sethu Vijayakumar*

Abstract

We consider the problem of direct policy learning in situations where the policies are only observable through their projections into the null-space of a set of dynamic, non-linear task constraints. We tackle the issue of deriving consistent data for the learning of such policies and make two contributions towards its solution. Firstly, we derive the conditions required to exactly reconstruct null-space policies and suggest a learning strategy based on this derivation. Secondly, we consider the case that the null-space policy is conservative and show that such a policy can be learnt more easily and robustly by learning the underlying potential function and using this as our representation of the policy.

1 Introduction

Redundant manipulators are characterised as having degrees of freedom in excess of those needed to perform some task. In the control of such systems a popular paradigm is to utilise redundancy through secondary movement policies that complement the primary task goals in some way. Such policies prefer actions that, for example, avoid joint limits [1], kinematic singularities [11] or self-collisions [9]. Traditionally these policies were implemented as optimising some carefully selected *instantaneous cost function* or potential. The approach was first proposed by Liégeois [3] in the context of Resolved Motion Rate Control (RMRC) [10], but has since been extended to other control regimes (notably force based control) and a wider variety of secondary policies. For example Nakamura [5] studied the problem extensively with particular emphasis on optimal RMRC and Resolved Acceleration Control (RAC) of arms with pre-defined tasks and time-integral cost functions.

However, the secondary policy need not be the result of optimisation and the formalism extends to a variety of constraint-based control scenarios. For example, in humanoid robots, a secondary goal might be to maintain some posture when performing some task [2], to perform multiple prioritised tasks at once [8] or perform control subject to contact constraints [6].

*School of Informatics, University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom. E-mail: matthew.howard@ed.ac.uk.

Furthermore, many tasks are best described as constrained with respect to certain variables. Consider running on a treadmill: the centre of mass and tilt of the torso are constrained and the policy controlling the gait is projected into the null-space of these constraints.

The focus in this paper is on modelling secondary control policies from observations of constrained motion using statistical learning methods. This is a form of direct policy learning, with the difference that the policy is only partially observable. For the learning, most supervised learning techniques require consistent, convex training data. In this paper, we look at the problem of deriving this data and offer two contributions for its solution.

The first concerns the general case where the policy can be any non-conservative vector function of the state. We take a geometric approach to reconstructing the policy based on Euclid's theorem and outline the necessary conditions for exact reconstruction at any given point. Furthermore, we show that this approach suggests an iterative training algorithm for our learner.

The second contribution is to show that, by restricting the class of permissible policies to those that optimise some potential, the policy can be inferred in a simpler and more robust way using a form of inverse optimal control. In this approach, identifying the unconstrained policy from constrained observations, is equivalent to seeking the potential being optimised. We show that by modelling the policy through its potential we can side-step several of the restrictions of the geometric approach.

Finally, we present experimental results for a simulated robot arm in which the reconstructed null-space policy is used to replace that of the original and the resultant behaviour is compared across a variety of consistent task goals.

2 Problem formulation

We consider control policies of the form

$$\mathbf{u} = \mathbf{u}_{task}(\mathbf{x}, t) + \mathbf{u}_{null}(\mathbf{x}, t) = \mathbf{u}_{task}(\mathbf{x}, t) + \mathbf{N}(\mathbf{x}, t)\mathbf{a}(\mathbf{x}) \tag{1}$$

where $\mathbf{u}$ is the control signal, $\mathbf{u}_{task}$ is the component of $\mathbf{u}$ that satisfies a set of non-linear, time-varying task constraints, $\mathbf{a}(\mathbf{x})$ is a policy pursuing secondary movement goals, and $\mathbf{N}(\mathbf{x}, t)$ is a projection matrix. $\mathbf{N}(\mathbf{x}, t)$ prevents violation of the task constraints by projecting the policy into the null-space. Our goal is to model $\mathbf{a}(\mathbf{x})$ from observations of $\mathbf{u}_{null}$.

In general, $\mathbf{a}(\mathbf{x})$ can be any arbitrary vector field. According to the Helmholtz decomposition, any vector field may be comprised of rotational and divergent components

$$\mathbf{a}(\mathbf{x}) = \nabla_{\mathbf{x}} \times \mathbf{\Phi}(\mathbf{x}) + \nabla_{\mathbf{x}}\phi(\mathbf{x}) \tag{2}$$

where $\mathbf{\Phi}$ and ϕ are vector and scalar potentials. Assuming that $\mathbf{a}(\mathbf{x})$ is conservative[1] an equivalent goal is to model $\phi(\mathbf{x})$. Policies of the form (1) occur in both velocity ($\mathbf{u} \equiv \dot{\mathbf{q}}$) and

[1] A necessary and sufficient condition for this is that $\nabla_{\mathbf{x}} \times \mathbf{a}(\mathbf{x}) = 0, \forall \mathbf{x}$.

force ($\mathbf{u} \equiv \boldsymbol{\tau}$) based control [4].

Example 2.1. *Velocity-based Control*
A standard velocity-based control scheme is RMRC [10, 3]

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q}, t)^{\dagger}\dot{\mathbf{r}} + \mathbf{N}(\mathbf{q}, t)\mathbf{a} \tag{3}$$

where $\mathbf{r}, \dot{\mathbf{r}} \in \mathbb{R}^k$ *and* $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^n$*, denote the task- and joint-space positions and velocities and* $\mathbf{J}(\mathbf{q}, t)$ *is the Jacobian with* $\mathbf{W}$*-weighted pseudoinverse* $\mathbf{J}^{\dagger} = \mathbf{W}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T)^{-1}$*.* $\mathbf{N}(\mathbf{q}, t) = (\mathbf{I} - \mathbf{J}^{\dagger}(\mathbf{q}, t)\mathbf{J}(\mathbf{q}, t))$ *is the null-space projection matrix (where* $\mathbf{I}$ *is the identity matrix). Note that, in general, the Jacobian and the projection matrix are time-dependent reflecting the fact that the task-space may change in time* [1].

Example 2.2. *Force-based Control*
A general formulation for force-based control is [7]

$$\boldsymbol{\tau} = \mathbf{W}^{-1/2}(\mathbf{A}\mathbf{M}^{-1}\mathbf{W}^{-1/2})^{\dagger}(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}) + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}, t)\mathbf{a} \tag{4}$$

where $\boldsymbol{\tau} \in \mathbb{R}^n$ *is the applied torque/force,* $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^n$ *are joint-space positions, velocities and accelerations,* $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ *is an inertia/mass matrix and* $\mathbf{F} \in \mathbb{R}^n$ *describes perturbing forces such as centrifugal, Coriolis and gravity forces. The weighting matrix* $\mathbf{W} \in \mathbb{R}^{n \times n}$ *determines the control paradigm used, such as RAC (*$\mathbf{W} = \mathbf{M}^{-2}$*) or the Operational Space Formulation (*$\mathbf{W} = \mathbf{M}^{-1}$*)* [7]*. The task is described through constraints of the form* $\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}}, t)\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}, t) \in \mathbb{R}^k$ *and the null-space projection matrix is given by* $\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{W}^{-1/2}(\mathbf{I} - (\mathbf{A}\mathbf{M}^{-1}\mathbf{W}^{-1/2})^{\dagger}\mathbf{A}\mathbf{M}^{-1}\mathbf{W}^{-1/2})\mathbf{W}^{1/2}$.

The correspondence between (1), (3) and (4) can easily be shown by appropriate substitution of variables. In both cases, the second term arises when there is redundancy, i.e. the task dimensionality is lower than that of the action space ($k < n$), allowing secondary goals to be pursued. Fig. 1 shows examples of how different null-space policies affect behaviour.

3 Reconstructing nullspace policies

Theorem 3.1. *Reconstruction of Projected Policies*
Given observations $\mathbf{a}^{(i)} = \mathbf{N}^{(i)}(\mathbf{x})\mathbf{a}(\mathbf{x})$*,* $i = 1, \ldots, n$ *of a policy* $\mathbf{a}(\mathbf{x})$ *projected into the null-space of a set of* n *task constraints which that span the action space, the unconstrained policy is given by*

$$\mathbf{a}(\mathbf{x}) = \mathbf{x}^{\times} - \mathbf{x} \tag{5}$$

where $\mathbf{x}^{\times}$ *is the solution to the linear system*

$$\mathbf{A}\mathbf{x}^{\times} = \mathbf{d} \tag{6}$$

where $\mathbf{A} \equiv (\mathbf{a}', \ldots, \mathbf{a}^{(m)})^T$ *and the elements of* $\mathbf{d}$ *are given by* $d_i = \mathbf{a}^{(i)T}(\mathbf{x} + \mathbf{a}^{(i)})$.

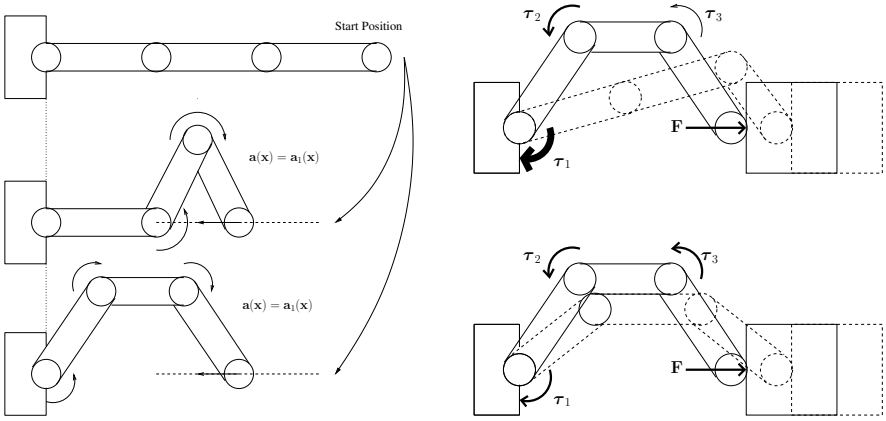


Figure 1: Effect of different null-space policies $\mathbf{a}(\mathbf{x})$ on behaviour when tracking a linear task-space trajectory in RMRC (left) and applying a force $\mathbf{F}$ to a mass in force control (right).

Proof Consider the RMRC control of a manipulator with two-dimensional joint space, $\mathbf{q} \equiv (q_1, q_2)^T$, and one-dimensional task space $r^{(i)}$, $i = 1, \ldots, n$. The Jacobian of this system

$$\mathbf{J}^{(i)}(\mathbf{q}) = (\alpha_1, \alpha_2)^{(i)} = \boldsymbol{\alpha}^{(i)} \tag{7}$$

is locally linear in the region of $\mathbf{q}$. Under task constraint i the null-space policy is constrained to a line in joint-space with intersection $r^{(i)}$ (Fig. 2, left). When the active constraint changes the rotation of this line changes so that the observed projections lie inscribed within a circle (hypersphere in n-d space) of diameter $||\mathbf{a}(\mathbf{q})||$ (Fig. 2, centre). Euclid's theorem states that any triangle inscribed in a semi-circle is a right-angle triangle. Hence $\mathbf{a}(\mathbf{q})$ is given by the intersection of the lines orthogonal to any two projections $\mathbf{a}', \mathbf{a}''$ (Fig. 2, right). By the same argument, in n-dimensional space, if observations are such that they form a basis set of the space, we can construct planes normal to the projections and solve for the intersection point $\mathbf{q}^{\times}$. This yields the linear system (6) with the unprojected vector given by (5). □

Theorem 3.1 also suggests the following lemma.

Lemma 3.1. *Given observations* $\mathbf{a}^{(i)} = \mathbf{N}^{(i)}(\mathbf{x})\mathbf{a}(\mathbf{x})$, $i = 1, \ldots, n$ *of a constrained policy* $\mathbf{a}(\mathbf{x})$, *the observation with the largest norm* $||\mathbf{a}^{(i)}||$ *lies closest to the unconstrained policy.*

Proof By inspection of Fig. 2, or by considering that $\mathbf{N}(\mathbf{x}, t)$ is a projection matrix, with k eigenvalues of value 0 and $n - k$ eigenvalues of value 1. Fewer constraints (smaller k) results in larger norms. □

Lemma 3.1 suggests an iterative approach to training whereby if multiple observations are made around the same point, those with the largest norm should be used for learning. This is particularly true of highly redundant systems ($k << n$) where there the policy is much less constrained. Furthermore, in the limit that observations are made under a single, constant constraint, a consistent policy $\mathbf{u}_{null}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\mathbf{a}(\mathbf{x})$ will be learnt.

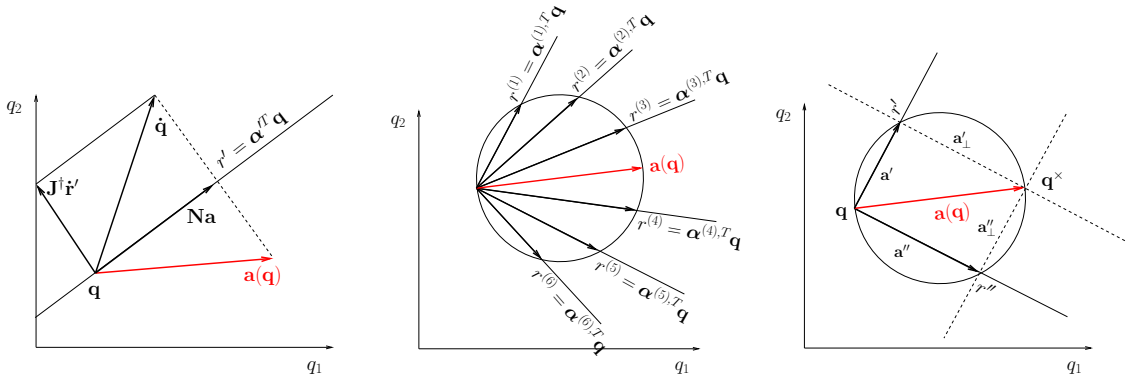


Figure 2: Under the task constraints (7), the null-space policy is projected onto a manifold $r = \boldsymbol{\alpha}^{(t)T}\mathbf{q}$ (left), orthogonal to the task space motion. Under multiple constraints the projected policy vectors lie inscribed in a hypersphere in state-space (centre). Euclid's Theorem can be used to reconstruct $\mathbf{a}(\mathbf{q})$ given observations under different constraints (right).

The condition in Theorem 3.1 that a spanning set of projections are required to exactly reconstruct the policy is somewhat restrictive, and in real data sets unlikely. However if the policy is conservative (i.e. the first term of (2) is zero) we can side-step these restrictions with the following proposition.

Proposition 3.1. *Reconstruction of Conservative Policies*
Under the same conditions as Theorem 3.1, a conservative policy $\mathbf{a}(\mathbf{x})$ *can be represented by its underlying potential function, which can be learnt without the need for multiple observations or iterative training.*

Consider again the case of RMRC of a redundant manipulator. The potential underlying a conservative $\mathbf{a}(\mathbf{q})$ can be reconstructed through inverse optimal control [4]. The simplest method requires trajectories sampled at some rate ρ resulting in a set of via-points $(\mathbf{q}_1 \dots \mathbf{q}_{\rho\tau})^T$ where

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \mathbf{N}(\mathbf{q}_t)\nabla_{\mathbf{q}}\phi(\mathbf{q}_t) \tag{8}$$

for a trajectory of duration τ and $\mathbf{u}_{task} = 0$. Training samples of $\phi(\mathbf{x})$ can be generated by integrating along trajectories using, for example, the Euler method

$$\phi(\mathbf{q}_{t+1}) = \phi(\mathbf{q}_t) + (\mathbf{q}_{t+1} - \mathbf{q}_t)^T\mathbf{N}(\mathbf{q}_t)\nabla_{\mathbf{q}}\phi(\mathbf{q}_t). \tag{9}$$

The key observation is that the integration in (9) occurs in the direction locally orthogonal to the constraints. We refer the reader to results reported in [4] for empirical evidence supporting Proposition 3.1.

In Fig. 3 the left-hand plot shows the true (blue) and reconstructed (cyan) potential along trajectories under a variety of constraints. Contours show the true (quadratic) potential function over two of the joints of the arm. The trajectories are reconstructed up to a translation

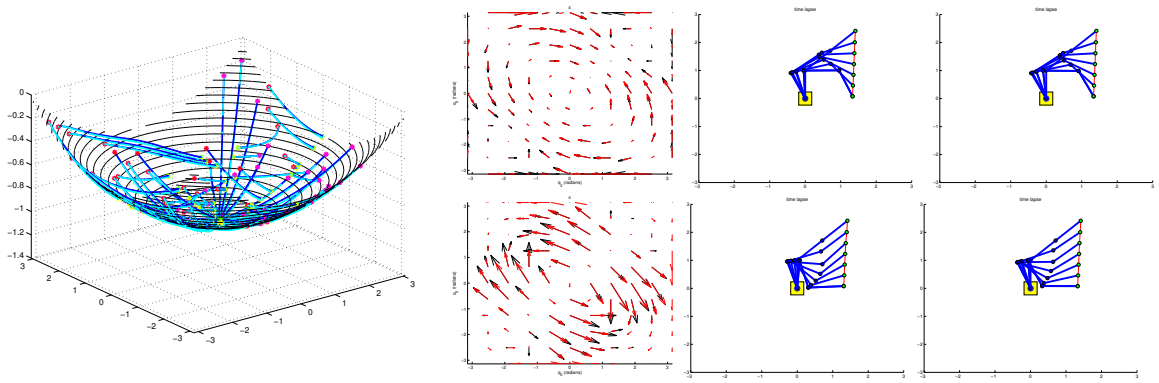


Figure 3: True (blue) and reconstructed (cyan) values of the quadratic potential (contours) along trajectories subject to different constraints (constraints on the hand, wrist, elbow, and unconstrained trajectories shown). True (black) and learnt (red) null-space policies subject to hand constraints for the quadratic (top) and sinusoidal (bottom) potentials. Time-lapse of the arm tracking a linear trajectory using the true (left) and learnt (right) null-space policies.

in the ϕ-dimension (trajectories have been translated in Fig. 3 for comparison). In the middle and right-hand plots a modified Euler method was used to learn two policies; that derived from a quadratic potential (top row) and a sinusoidal one (bottom row); The middle plots show the true and reconstructed policy subject to constraints on the hand. The right-hand plots show a time-lapse of the arm tracking a linear trajectory using the true and learnt policy in the null-space.

4 Conclusion

We have presented the mathematical basis for direct policy learning of policies subject to dynamic, non-linear constraints. We have shown that in the general case of non-conservative policies exact reconstruction of the policy requires solution of a system of equations constructed from observations under task constraints that span the state-space. We have noted that this suggests an iterative training scheme based on the norm of observed projections. Finally, we have suggested a more robust approach to learning conservative policies through numerical integration techniques and simulation results have been presented for the learning of such policies for a kinematically-controlled three link arm.

References

[1] M. Gienger, H. Janssen, and C. Goerick. Task-oriented whole body motion for humanoid robots. In *IEEE-RAS Int. Conf. on Humanoid Robots*, 2005.

[2] O. Khatib, J. Warren, V. De Sapio, and L. Sentis. Human-like motion from physiologically-based potential energies. In J. Lenarcic and C. Galletti, editors, *On Advances in Robot Kinematics*. Kluwer Academic Publishers, 2004.

[3] A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. In *IEEE Trans. Syst., Man, Cybern.*, volume 7, 1977.

[4] Howard M., M. Gienger, C. Goerick, and S. Vijayakumar. Learning utility surfaces for movement selection. In *IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, 2006.

[5] Y. Nakamura. *Advanced Robotics: Redundancy and Optimization*. Addison Wesley, Reading, MA, 1991.

[6] J. Park and O. Khatib. Contact consistent control framework for humanoid robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.

[7] J. Peters, M. Mistry, F. Udwadia, R. Cory, J. Nakanishi, and S. Schaal. A unifying methodology for the control of robotic systems. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[8] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.

[9] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick. Real-time self collision avoidance for humanoids by means of nullspace criteria and task intervals. In *IEEE-RAS Int. Conf. on Humanoid Robots*, 2006.

[10] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. 10(22), 1969.

[11] T. Yoshikawa. Manipulability of robotic mechanisms. *Int. J. Robotics Research*, 4(2), 1985.

On the geometry of rolling maps and applications to Robotics

K. Hüper* M. Kleinsteuber† F. Silva Leite‡

Abstract

The main objective of our paper is to introduce rolling maps using the formalism of differential geometry, as done in Sharpe [3], while illustrating the mathematical concepts using simple examples of manifolds which are particularly important in Robotics.

Rolling maps describe how one smooth manifold rolls on another, without twist or slip. The most consagrated of these nonholonomic mechanical systems is the rolling sphere. There are two types of constraints for a rolling sphere on a surface. A holonomic one, which insures that the sphere remains tangent to that surface, and two nonholonomic constraints which ensure that the sphere rolls without slip or twist.

Examples of mechanical systems which contain rolling elements, and are naturally modeled as nonholonomic systems, are: aircraft nose wheels, motorcycles, automotive systems and trailer systems. Rolling motions appear in the robotics context associated with rigid bodies moving while keeping point contact. As a consequence, understanding the geometry of rolling from a theoretical point of view is an important step towards applications in Robotics. This methodology may be applied in several situations, such as: wheeled mobile robots; a single robot arm maintaining controlled contact against a moving environment; two arms manipulating an object with rolling contact between each arm and the object.

We consider Riemannian manifolds, equipped with a metric induced by the Euclidean metric from the embedding space, and rolling as a rigid body on their affine tangent space at a point. The equations of motion can be described by a control systems with constraints on velocities and evolving on the Euclidean group of rigid motions, describing simultaneously rotations and translations in space. In previous work [2] we considered the case when the rolling manifold is the group of rotations SO_n or a Grassmann manifold of all k-dimensional subspaces of an n-dimensional real Euclidean space. In [1] we studied rolling maps for real Stiefel manifolds, which are the sets of all orthonormal k-frames of an n-dimensional real Euclidean space. We now extend the previous work to the case when

*NICTA, Canberra Research Laboratory, Australia, and Department of Information Engineering, Research School of Information Sciences and Engineering, Canberra, Australia. E-mail: knut.hueper@nicta.com.au.

†Mathematisches Institut, Universität Würzburg, Germany. E-mail:kleinsteuber@mathematik.uni-wuerzburg.de.

‡Institute of Systems and Robotics, University of Coimbra, and Department of Mathematics, University of Coimbra, Portugal. E-mail:fleite@mat.uc.pt.

the Euclidean group SE_n rolls without slip or twist on its tangent space at the identity. The main difficulty when extending notions of rolling to a particular manifolds is that, contrary to the sphere case, we loose geometric intuition. But the properties of rolling motions call for new developments in this area. We will highlight the great advantages of rolling motions through applications to solutions of interpolation problems on our favorite non-Euclidean spaces.

References

[1] K. Hüper, M. Kleinsteuber, and F. Silva Leite. Rolling Stiefel manifolds. International Journal of Systems Science. To appear in 2007.

[2] K. Hüper and F. Silva Leite. On the geometry of rolling and interpolation curves on s^n, $\mathfrak{so}_n$ and Graßmann manifolds. Journal and Dynamical and Control Systems. To appear in 2007.

[3] R. W. Sharpe. *Differential Geometry.* Springer, New York, 1996.

On the computation of the Karcher mean on spheres and special orthogonal groups

Krzysztof A. Krakowski* Knut Hüper† Jonathan H. Manton‡

Abstract

This paper is concerned with computation of the *Karcher mean* on the unit sphere $\mathbf{S}^n$ and the special orthogonal group $\mathbb{SO}(n)$. The Karcher mean, or the *Riemannian centre of mass*, is defined as the point minimising the sum of the squared distances from that point to each of the given points. By its definition, the mean always belongs to the same space as the given points, however, it may not be unique. Motivated by applications in control, vision and robotics, this paper studies the numerical computation of the Karcher mean. We propose simpler and computationally more efficient gradient-like and Newton-like algorithms. We give explicit forms of these algorithms and show that if the set of points lie within a particular open ball, the algorithms are guaranteed to converge to the Karcher mean.

1 Introduction

The concept of an average, center of gravity or centroid of a set of points generalized to spaces other than $\mathbb{R}^n$ appears in the literature in a number of different contexts. Since the paper of Karcher [3] the average is often referred to as the Karcher mean, the term adopted here by the authors. This paper deals with a final set of points but the results extend to the case of a mass distribution.

Let $\mathbf{M}$ be a complete Riemannian manifold and Ω a final set of points in $\mathbf{M}$. Define a generalized *variance* to be the function $\Psi\colon \mathbf{M} \to \mathbb{R}$ given by

$$\Psi(x) := \frac{1}{2\#\Omega} \sum_{p\in\Omega} \operatorname{dist}(x,p)^2, \tag{1}$$

*School of Mathematics, Statistics and Computer Sciences, University of New England, Armidale, NSW 2350, Australia. E-mail:kris@mcs.une.edu.au.

†National ICT Australia, Canberra Research Laboratory, Locked Bag 8001, Canberra ACT 2601, Australia, and Department of Information Engineering, RSISE, The Australian National University, Canberra, ACT 0200, Australia. E-mail:knut.hueper@nicta.com.au.

‡Department of Information Engineering, Research School of Information Sciences and Engineering (RSISE) The Australian National University, Canberra, ACT 0200, Australia. E-mail:j.manton@ieee.org.

where $\mathrm{dist}(\cdot,\cdot)$ is the Riemannian (geodesic) distance in $\mathbf{M}$ and $\#\Omega$ is the cardinal, the number of points, of Ω. The *Karcher mean* $\bar{x} \in \mathbf{M}$ is a point where Ψ attains its minimum. It is known (*cf.* [3]) that if $\Omega \subset \mathscr{B}_\varrho$, where $\mathscr{B}_\varrho \subset \mathbf{M}$ is the open Riemannian ball of radius ϱ, and ϱ is sufficiently small so that $\mathscr{B}_\varrho$ is convex[1] and if $\varrho < \pi/4\,C^{-1/2}$, where C is a sectional curvature on $\mathscr{B}_\varrho$, then Ψ is convex on $\mathscr{B}_\varrho$. As a consequence Ψ has a unique point of local minimum in $\mathscr{B}_\varrho$, therefore the Karcher mean $\bar{x}$ is unique and belongs to $\mathscr{B}_\varrho$.

The motivation for this paper is twofold. The recent paper [6] considers $\mathbb{SO}(3)$, so it is natural to consider $\mathbb{SO}(n)$ for $n > 3$. In fact, $\mathbb{SO}(3)$ is a very special case because it has constant sectional curvature, whereas $\mathbb{SO}(n)$ doesn't, for $n > 3$. Therefore, not only do not the methods in [6] extend, it is not even clear in advance whether or not results valid for $n = 3$ remain true if $n > 3$. It is also remarked that no numerical algorithm for finding the Karcher mean was proposed in [6]. The main contribution of this paper are the new iterative algorithms whose convergence rate are the same as the gradient descent and the Newton, but are simpler and computationally more efficient than, say the intrinsic Newton and gradient descent algorithms for *spherical weighted averages* in $\mathbf{S}^n$, *cf.* [1].

The rest of this paper is organised as follows. Section 2 describes properties of the Karcher mean. Here we highlight the connection of the Karcher mean with convexity of a set containing the points. The algorithms to compute the Karcher mean in the two spaces, $\mathbb{SO}(n)$ and $\mathbf{S}^n$ are presented in Section 3. Finally, Section 4 concludes the paper.

2 Existence and uniqueness of the Karcher mean

It turns out that existence and uniqueness of the Karcher mean depends on the space and its convexity. There are possible problems arising from the curvature of a space. Firstly, since the distance depends on minimizing Riemannian geodesics, one has to make sure that the minimising geodesic exists and is unique — injectivity radius, and then that it lies entirely in the ball containing the set of points Ω — convexity radius. It is also of the interest to know whether the Karcher mean is unique — convexity of Ψ.

Table 1 summarizes geometric properties of the two spaces considered in this paper. The both spaces are symmetric and the unit sphere is a quotient space $\mathbf{S}^n = \mathbb{SO}(n+1)/\mathbb{SO}(n)$. Because the unit sphere is more curved, it poses more theoretical problems related to the existence and uniqueness of the Karcher mean than $\mathbb{SO}(n)$.

Since the paper of Karcher [3], mentioned in the introduction, another significant development has been brought by the work on *convex geometry* by Kendall [4]. There the maximum radius of the convex Riemannian ball containing Ω, for which the Karcher mean exists and is unique, has been improved. For the unit sphere $\mathbf{S}^n$ the radius of the open ball is $\pi/2$. That

[1] We say that $\mathcal{U} \subset \mathbf{M}$ is *convex* if for any $p, q \in \mathcal{U}$ there is a unique in $\mathbf{M}$ minimizing geodesic γ from p to q and $\gamma \subset \mathcal{U}$.

	Special Orthogonal Group $\mathbb{SO}(n)$	**Unit Sphere** $\mathbf{S}^n$
tangent vector	XA — tangent at X, where A — skew-symmetric matrix	V — tangent at x, $\langle V, x\rangle = 0$, product in the ambient $\mathbb{R}^{n+1}$
the inner product	$\langle U, V\rangle := \frac{1}{2}\mathrm{tr}\left(U^{\mathrm{T}}V\right)$, where $U, V \in \mathfrak{so}(n)$	spherical, $\langle U, V\rangle := U^{\mathrm{T}}V$ (in $\mathbb{R}^{n+1}$)
exponential map	$\exp_X(XA) = Xe^A$	$\exp_x V = \cos\|V\|\ x + \frac{\sin\|V\|}{\|V\|}V$
log map	$\exp_X^{-1}(P) = X \log X^{\mathrm{T}}P$	$\exp_x^{-1}(p) = (p - x\langle x, p\rangle)\ \frac{\arccos\langle x,p\rangle}{\sqrt{1-\langle x,p\rangle^2}}$
curvature	$C(X,Y) = \frac{1}{4}\left\|[X,Y]\right\|^2$, for $\mathbb{SO}(3)$ C is $\frac{1}{4}$	$C = \frac{1}{R^2}$, for the unit sphere C is 1
injectivity radius[a]	π	π
convexity radius[b]	$\pi/2$	$\pi/2$

[a] for details on how to establish the radius for symmetric spaces *cf.* Kobayashi & Nomizu [5]

[b] follows from the Whitehead's Theorem, *cf.* [5]

Table 1: Comparison of geometric properties of $\mathbb{SO}(n)$ and $\mathbf{S}^n$.

means that if Ω is contained in an open half-sphere, the Karcher mean exists and it is unique. Buss & Fillmore's [1] investigations for the sphere relaxed slightly the conditions allowing the half-sphere to be closed as long as at least one point of Ω lies inside it.

3 Computation of the Karcher mean

In this section we introduce new iterative methods of calculating Karcher mean on $\mathbb{SO}(n)$ and $\mathbf{S}^n$. They are based on the two standard methods of deriving non-degenerate critical points, namely the gradient descent and the Newton methods. For a smooth function $f\colon \mathbb{R}^n \to \mathbb{R}$ let $x^* \in \mathbb{R}^n$ be a non-degenerate critical point of f, i.e., the Hessian $\mathrm{Hess}\, f(x^*)$ is invertible. Let $\mathfrak{N}_f\colon \mathbb{R}^n \to \mathbb{R}^n$ be given by $x \mapsto \mathfrak{N}_f(x) := x - (\mathrm{Hess}\, f)^{-1}\,\mathrm{grad}\, f(x)$. The *Newton method* is the iteration $x_{i+1} = \mathfrak{N}_f(x_i)$. Similarly, let $\mathfrak{G}_f\colon \mathbb{R}^n \to \mathbb{R}^n$ be given by $x \mapsto \mathfrak{G}_f(x) := x - \mathrm{grad}\, f(x)$. The *gradient descent method* is the iteration $x_{i+1} = \mathfrak{G}_f(x_i)$. The two methods easily extend to Riemannian manifolds. Given a smooth function $f\colon \mathbf{M} \to \mathbb{R}$ and the Riemannian normal coordinates $\varphi_{x_i}\colon \mathbb{R}^n \to \mathbf{M}$ centered at $x_i \in \mathbf{M}$ the iterations

$$x_{i+1} = \varphi_{x_i}\Big(\mathfrak{N}_{f\circ\varphi_{x_i}}(0)\Big) \quad \text{and} \quad x_{i+1} = \varphi_{x_i}\Big(\mathfrak{G}_{f\circ\varphi_{x_i}}(0)\Big) \tag{2}$$

become the intrinsic Newton and gradient methods, respectively. The gradient of the variance of $\Psi\colon \mathbf{M} \to \mathbb{R}$ given by (1) has a particularly simple form $\mathrm{grad}\,\Psi(x) = -\frac{1}{\#\Omega}\sum_{p\in\Omega}\varphi_x^{-1}(p)$, *cf.* [3], and so the intrinsic gradient descent algorithm (2) becomes

$$x_{i+1} = \varphi_{x_i}\left(\frac{1}{\#\Omega}\sum_{p\in\Omega}\varphi_{x_i}^{-1}(p)\right). \tag{3}$$

However, based on the observation in Hüper & Trumpf [2], the same rate of convergence can be obtained with algorithms where Riemannian normal coordinates φ_x are replaced with any,

possibly different, parametrizations agreeing, up to the first derivative, with φ_x. We call these new algorithms the *Newton-like* and the *gradient-like*. We now present these algorithm for the two symmetric spaces, $\mathbb{SO}(n)$ and $\mathbf{S}^n$.

The special orthogonal group

The Riemannian Hessian *operator* $\operatorname{Hess}\Psi\colon \mathcal{T}_X\mathbb{SO}(n) \to \mathcal{T}_X\mathbb{SO}(n)$ of the variance Ψ given by (1) has the following form

$$\operatorname{Hess}\Psi(XA) = \frac{1}{\#\Omega}\sum_{P\in\Omega} \mathcal{R}_{\log(XP^{\mathrm{T}})}\cdot XA, \tag{4}$$

where $\mathcal{R}_X\colon \mathfrak{so}(n) \to \mathfrak{so}(n)$ is defined by

$$Y \mapsto \frac{\operatorname{ad}_X}{2}\coth\frac{\operatorname{ad}_X}{2}\cdot Y \quad \text{with} \quad \mathcal{R}_0(Y) = Y$$

and the linear operator $\operatorname{ad}_X\colon \mathfrak{so}(n) \to \mathfrak{so}(n)$ is given by $\operatorname{ad}_X Y := [X,Y] = XY - YX$.

Theorem 3.1. *Let $\Psi\colon \mathbb{SO}(n) \to \mathbb{R}$ be defined by* (1).

1. *The critical points of* (1) *are precisely the solutions of*

$$\sum_{P\in\Omega}\log(P^{\mathrm{T}}X) = 0. \tag{5}$$

2. *At smooth points of* (1)*, the Riemannian Hessian of* (1) *is always positive definite.*

Generalized Newton methods Up to our best knowledge there is no known closed form solution of (5) in general. We start this section by proposing Newton-like methods for computing the Karcher mean in $\mathbb{SO}(n)$. Let $X \in \mathbb{SO}(n)$ and consider a smooth parametrization around X $\mu_X\colon \mathbb{R}^{n(n-1)/2} \to \mathbb{SO}(n)$ with $\mu_X(0) = X$, i.e., a local diffeomorphism around $0 \in \mathbb{R}^{n(n-1)/2}$. If there exists an open neighborhood $\mathcal{U} \subset \mathbb{SO}(n)$ of X^* and a smooth map

$$\mu\colon \mathcal{U}\times\mathbb{R}^{n(n-1)/2} \to \mathbb{SO}(n)$$

such that $\mu(X,v) = \mu_X(v)$, for all $X \in \mathcal{U}$ and $v \in \mathbb{R}^{n(n-1)/2}$, we will call $\{\mu_X\}_{X\in\mathbb{SO}(n)}$ a *locally smooth family of parametrizations around X^**. We now present three different families of this kind. In what follows we will identify $\mathfrak{so}(n) \cong \mathbb{R}^{n(n-1)/2}$.

$$\mu^{\mathrm{GS}}\colon \mathcal{U}\times\mathfrak{so}(n) \to \mathbb{SO}(n) \qquad \text{given by} \qquad (X,A) \mapsto Q\left(X(\mathbf{I}+A)\right), \tag{6}$$

$$\mu^{\mathrm{cay}}\colon \mathcal{U}\times\mathfrak{so}(n) \to \mathbb{SO}(n) \qquad \text{given by} \qquad (X,A) \mapsto X\left(\mathbf{I}+\frac{A}{2}\right)\left(\mathbf{I}-\frac{A}{2}\right)^{-1}, \tag{7}$$

$$\mu^{\exp}\colon \mathcal{U}\times\mathfrak{so}(n) \to \mathbb{SO}(n) \qquad \text{given by} \qquad (X,A) \mapsto X\exp A, \tag{8}$$

here $Q(X(\mathbf{I}+A))$ denotes the Q-factor of the unique QR-decomposition of the matrix $X(\mathbf{I}+A)$, i.e., the orthogonal matrix produced from the Gram-Schmidt process applied to the columns of $X(\mathbf{I}+A)$. The following Lemma is easily verified.

Lemma 3.1. *Let $\mathcal{U} \subset \mathbb{SO}(n)$ be an open neighborhood of $X \in \mathbb{SO}(n)$. Let $\mu_X^{\mathrm{GS}}, \mu_X^{\mathrm{cay}}$, and $\mu_X^{\exp}$ be defined as* (6), (7), *and* (8), *respectively. Then for arbitrary $A \in \mathfrak{so}(n) \cong \mathbb{R}^{n(n-1)/2}$*

$$\mathrm{D}\,\mu_X^{\exp}(0) \cdot A = \mathrm{D}\,\mu_X^{\mathrm{cay}}(0) \cdot A = \mathrm{D}\,\mu_X^{\mathrm{GS}}(0) \cdot A = XA. \tag{9}$$

Algorithm 3.1 (Newton-like Method)**.** *Given a final set of points $\Omega \subset \mathbb{SO}(n)$, compute a local minimum of Ψ given by* (1).

Step 1 *Set $X \in \mathbb{SO}(n)$ to an initial estimate of the Karcher mean, such as to any point of Ω.*

Step 2 *Compute* $\dfrac{1}{\#\Omega} \displaystyle\sum_{P \in \Omega} \log(P^{\mathrm{T}} X)$.

Step 3 *Stop if* $\left\| \dfrac{1}{\#\Omega} \displaystyle\sum_{P \in \Omega} \log(P^{\mathrm{T}} X) \right\|$ *is sufficiently small.*

Step 4 *Solve the linear equation* $\operatorname{Hess} \Psi(XA) = -\operatorname{grad} \Psi(X)$ *for A.*

Step 5 *Set $X := \mu_X(A)$.*

Step 6 *Goto* **Step 2**.

We have the following result.

Theorem 3.2. *If Algorithm 3.1 converges then it converges locally quadratically fast.*

The unit sphere

From now on we assume that Ω is contained in an open half-sphere. For any $x \in \mathbf{S}^n$ and $V \in \mathcal{T}_x \mathbf{S}^n$ the Riemannian Hessian of the variance $\Psi \colon \mathbf{S}^n \to \mathbb{R}$ is given by

$$\operatorname{Hess} \Psi(V, V) = \frac{1}{\#\Omega} \sum_{p \in \Omega} \left(\frac{1}{\sin^2 \theta_p} \left(1 - \theta_p \cot \theta_p\right) \langle V, p \rangle^2 + \theta_p \cot \theta_p \, \|V\|^2 \right),$$

where $\theta_p = \operatorname{dist}(x, p) = \arccos \langle x, p \rangle$. There are the two important properties of $\operatorname{Hess} \Psi$ that are essential to analysis of convergence of the algorithms (2).

Lemma 3.2. *The Hessian of Ψ satisfies* $\operatorname{Hess} \Psi(V, V) \leq \|V\|^2$, *for any $x \in \mathbf{S}^n$ and $V \in \mathcal{T}_x \mathbf{S}^n$.*

Lemma 3.3. *The Hessian* $\operatorname{Hess} \Psi$ *is positive definite at the Karcher mean $\bar{x}$. Therefore there exists a neighbourhood $\mathcal{U} \subset \mathbf{S}^n$ of $\bar{x}$, where Ψ is convex.*

The intrinsic gradient descent and the Newton algorithms in the sphere studied in Buss & Fillmore [1] are given by (2). These algorithms are computationally expensive because at each iteration they require calculations of normal coordinates of every point of Ω, *cf.* (3). As in the case of $\mathbb{SO}(n)$, we may replace the normal coordinates φ_x with another mapping, and preserve the rate of convergence. For the sphere a good choice is the orthogonal projection

$$\pi_x \colon \mathbb{R}^{n+1} \to \mathcal{T}_x \mathbf{S}^n \quad \text{given by} \quad z \mapsto \pi_x(z) := z - \langle x, z \rangle \, x = (\mathbf{I} - xx^{\mathrm{T}}) z.$$

We conclude this section with a new gradient-like algorithm.

Generalized Gradient Method The gradient-like algorithm (10) in the unit sphere $\mathbf{S}^n$ converges to the Karcher mean locally linearly fast

$$x_{i+1} = \varphi_{x_i}\left(\frac{1}{\#\Omega}\sum_{p\in\Omega}\pi_{x_i}(p)\right). \tag{10}$$

4 Conclusion

This paper presents, to the best of our knowledge, the first efficient and reliable numerical algorithm for computing the Karcher mean of points in $\mathbb{SO}(n)$. Specifically, under the same necessary condition which ensures the Karcher mean is well defined, the algorithm is proved to converge to the Karcher mean. Moreover, the local rate of convergence is quadratic. Finally, a new gradient-like algorithm in $\mathbf{S}^n$ is presented.

References

[1] Samuel R. Buss and Jay P. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics*, 20(2):95–126, 2001.

[2] Knut Hüper and Jochen Trumpf. Newton-like methods for numerical optimization on manifolds. In *Proceedings of the 38 Asilomar Conference on Signals, Systems and Computers, November 7–10 2004, California, USA*, volume 1, pages 136–139, 2004.

[3] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30:509–541, 1977.

[4] Wilfrid S. Kendall. Probability, convexity and harmonic maps with small image I: uniqueness and fine existence. *Proceedings of the London Mathematical Society*, 61:371–406, 1990.

[5] Shoshichi Kobayashi and Katsumi Nomizu. *Foundations of Differential Geometry*, volume 2 of *Interscience tracts in pure and applied mathematics*. Interscience Publishers, New York, 1969.

[6] Maher Moakher. Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):1–16, 2002.

Reorienting a quasi-rigid body using shape changes

Marie Bro* Maria Mose†

Abstract

It is well-known that a falling cat can reorient itself merely by changing its shape. We present here a ball and stick model which captures the features necessary for a cat to turn without external torque. We calculate the reorientation of this model resulting from a cyclic sequence of shape changes, or in terms of differential geometry the geometric phase obtained by following a given closed path in shape space. We further determine via numerical implementation of this model the most efficient series of shape changes leading to a given reorientation. In other words we solve numerically a version of the isoholonomic problem.

1 The isoholonomic problem

The concept of geometric phase have been studied by a number of authors [1], [6], [7], [9]. We give here a brief outline of the concept, and how it should be understood in the context of control theory.

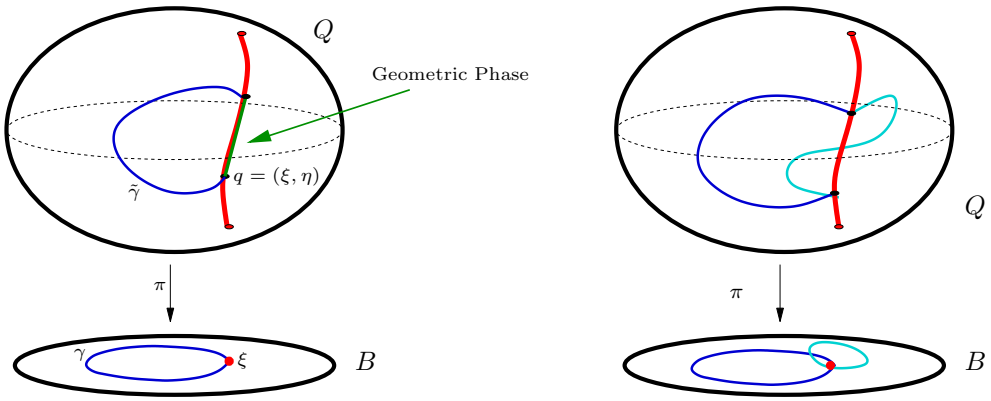
The geometric phase of a dynamical system is a certain type of phase shift. Any continuous change in the variables of the system corresponds to a curve its configuration space. If the system is a control system the variables can be split into control variables and state variables, and the configuration space can be given the structure of a fiber bundle. The set of control variables will constitute the base space and the state variables form the fibers.

One may then ask the question: 'Given a curve in base space, which curve in the configuration space will result from the changes in the control variables along this curve?'. In terms of differential geometry this question adresses the problem of lifting a curve from base space to the fiber bundle in a unique way. Lifts in general are not unique, but the so-called horisontal lifts are. They occur in the presence of a connection since a connection define the horisontal spaces. Connections often emanate from conservation laws or other constraints on the system. The phase shift in the state variables obtained from lifting a closed curve in the base space horisontally is the geometric phase (in references [7] and [8] refered to as holonomy).

*Department of Mathematics, Technical University of Denmark. E-mail:M.Bro@mat.dtu.dk.

†Department of Mathematics, Technical University of Denmark. E-mail:mariamose@hotmail.com.

When falling, the cat controls its orientation by changing its shape. Hence, the control variables are those that describe the shape, and the state of the system is the orientation. The falling cat's problem is to obtain a certain reorientation (i.e geometric phase) as efficiently as possible. It is therefore a special case of The Isoholonomic Problem [8]: *Among all curves with a fixed geometric phase, find the loop of minimum length.*



(a) The geometric phase is the phase shift in the state variables (i.e. the fiber variables) obtained when the control variables follow a closed curve in base space. The geometric phase is determined by performing a horizontal lift of the curve to the configuration space

(b) Sketch of the isoholonomic problem. Several closed curves in the base space with the same starting point might lead to the same geometric phase. The isoholonomic problem is to determine the shortest of these.

Figure 1:

This is an optimal control problem, and for the model we propose in the next section it has the formulation:

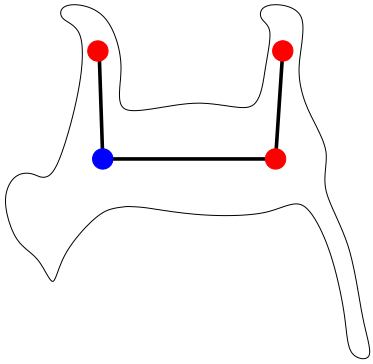
$$\begin{aligned}
\underset{\xi(t)\in B}{\text{minimize}} \quad & f(\xi) = \int_{t_0}^{t_1} \|\dot{\xi}(t)\| dt \\
\text{such that} \quad & \dot{\eta}(t) = \mathbf{g}\Big(\xi(t), \eta(t)\Big)\dot{\xi}(t), \\
& \xi(t_0) = \xi(t_1) = \xi_0, \\
& \eta(t_0) = \eta_0, \\
& \eta(t_1) = \eta_{end}, \qquad\qquad t \in [t_0, t_1]
\end{aligned} \tag{1}$$

Here, the objective function $f(\xi)$ is the length of the curve in base space; the first constraint ensures that the lift is horizontal; the second constraint guarantees a closed curve and that the starting point is kept fixed; and finally the third and the fourth constraint gives the correct initial and final orientations, hence the correct geometric phase.

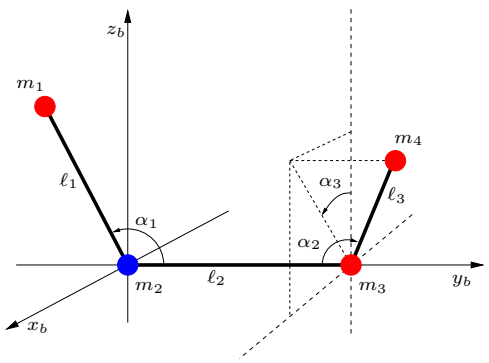
2 The mechanical model

Different models of the falling cat have been proposed most of which model the cat as two coupled rigid bodies [3], [4], [5]. These models differ in the nature of the coupling of the two bodies. Here, we propose a ball and stick model of the falling cat.

Translation plays no part in reorientation through shape changes [6], [8], and therefore we consider a quasi-rigid body, modelling the cat in a frame where origin is at the center of mass. Hence each configuration comprise a shape, described by the internal variables, and an orientation, described by the external variables. As we saw in the previous section the shape variables are the control variables and the orientation variables are the state variables. This means that when viewing the configuration space as a fiber bundle, the shape space constitutes the base space and the group of rotations in $\mathbb{R}^3$ forms the fibers. Since we are dealing with reorientation of the quasi-rigid body, it makes sense to assign $SO(3)$ as the structure group of the fiber bundle thus making it a principal bundle.



(a) A schematic drawing of a cat in relation to the proposed model.



(b) The body frame of the model. The shape is given by the three angles α_1, α_2 and α_3.

Figure 2:

The model consists of four point masses m_1, m_2, m_3, and m_4 which are connected by three massless rods ℓ_1, ℓ_2 and ℓ_3 (see Figures 2(a) and 2(b)). The rods ℓ_1 and ℓ_3 represent the forelegs and the hind legs, respectively, while ℓ_2 respresents the spine of the cat. The shape space is $S^1 \times S^2$, as indicated by figure 2(b).

Elements in shape space are denoted ξ, and elements in $SO(3)$ are denoted η. Hence, the position $\mathbf{s}_i$ of the mass m_i in the laboratory frame is given by:

$$\mathbf{s}_i(\xi, \eta) = \mathbf{R}(\eta)\big(\mathbf{b}_i(\xi) - \mathbf{b}_{CM}(\xi)\big), \quad i = 1, \ldots, 4$$

where $\mathbf{R}(\eta)$ is a rotation matrix, and $\mathbf{b}_i(\xi)$ and $\mathbf{b}_{CM}(\xi)$ are the position vectors in the body frame for m_i and the center of mass respectively.

A set of ODE's for calculating the geometric phase for this model along a path in shape space can now be derived using the conservation of angular momentum. We assume that the cat is initially at rest, and hence we impose the constraint 'angular momentum = 0' on the system. This constraint is equivalent to the mechanical connection [1], [8]. The angular momentum is calculated from the position vectors, and it turns out that it can be written as the sum of a matrix $\boldsymbol{\mathcal{K}}_\eta(\xi,\eta)$ times the time derivative of the orientation, $\dot{\eta}$ and a matrix $\boldsymbol{\mathcal{K}}_\xi(\xi,\eta)$ times the time derivative of the shape $\dot{\xi}$.

$$\mathbf{L}_{CM} \;=\; \sum_{i=1}^{3} \mathbf{s}_i \times m_i \dot{\mathbf{s}}_i \;=\; \boldsymbol{\mathcal{K}}_\eta(\xi,\eta)\dot{\eta} + \boldsymbol{\mathcal{K}}_\xi(\xi,\eta)\dot{\xi} \tag{2}$$

Given the constraint on the angular momentum, and assuming that $\boldsymbol{\mathcal{K}}_\eta(\xi,\eta)$ is regular, we obtain the following relation between small changes in the orientation and small changes in the shape:

$$d\eta = -\big(\boldsymbol{\mathcal{K}}_\eta(\xi,\eta)\big)^{-1}\boldsymbol{\mathcal{K}}_\xi(\xi,\eta)\; d\xi \tag{3}$$

The reorientation obtained by following a given closed curve in shape space (i.e. going through a series of shape changes beginning and ending with the same shape) can be calculated by integrating the above expression. On top of this, we want to determine an optimal series of shape changes which the (model) cat can perform in order to reorient itself as to land on its feet. This problem is addressed in the next section.

3 Implementation

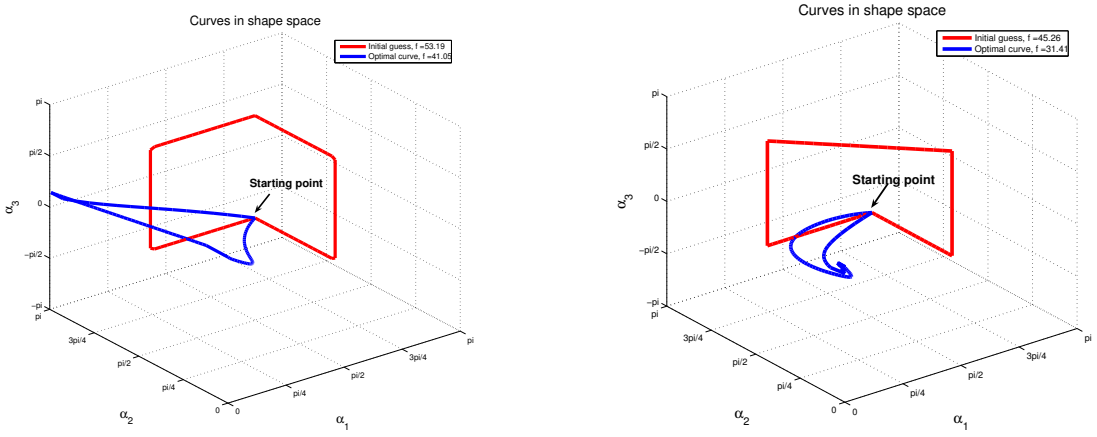
The expression relating small changes in shape to small changes in orientation, (3), can easily be integrated using e.g. a standard ODE solver in MATLAB. However, the software used in solving the isoholonomic problem numerically requires the gradients of the constraints, hence also of the ODE-solver, as input. It is, therefore, necessary to use a known and simple numerical method. To keep things relatively simple the time derivatives of ξ are discretized using simple forward differences, while the ODEs (i.e. the time derivatives of η) are discretized using Eulers Method.

The software used for solving the optimal control problem was SNOPT which is a Fortran based package for MATLAB. SNOPT solves large non-linear optimization problems using an SQP-method [2]. SQP-methods are iterative, and hence an initial guess is required. An initial guess is some closed curve in shape space which represents a series of shape changes that leads to the desired reorientation. We have constructed three, from careful studies of photos of real cats while falling. Two of these are shown as red curves in figures 3(a) and 3(b).

4 Results

We have obtained various results from optimizing the initial guesses. The main conclusion to be drawn is that there are several local minima, and that initial guesses that are quite similar can lead to very different minima when optimized. This is not very surprising from a mathematical point of view, and when studying photographs of cats flipping in the air one realizes that different cats use different methods.

Figure 3 shows two different curves in shape space each resulting from optimizing an initial curves with respect to the isoholonomic problem. The two initial curves (the red curves) are quite similar, but the results (the blue curves) are considerably different. One result (see figure 3(a)) is in good agreement with some of the motions we have inferred from photographs of cats. The other 3(b) is quite unphysical. The principal cause for this is the crudity of the model. It has no 'body', no extension. Another reason might be the low order of the discretization.



(a) This result (the blue curve) resembles fairly well some of the motions we have observed in photographs of falling cats. It corresponds to a series of motions where the cat draws its legs towards the body and at the same time twist its spine, followed by stretching out the hind legs, and finally moving both hind- and forelegs to their initial position while twisting the spine in the opposite direction.

(b) This result is quite unphysical. It corresponds to a series of motions where the cat pull in its legs and 'work a bit' with them in various directions and the unfold them. The reorientation is obtained by the small movements of the legs.

Figure 3: Initial curves (red) and the resulting optimized curves (blue) in shape space

Our model of the cat can give an idea about how a quasi-rigid-body can be reoriented. Furthermore we have discovered several curves (the initial guesses) that gives the desired reorientation, and represents a movement similar to the one performed by the cat. However, the optimization results suggest that the presented model does not capture the features of the

cat to an extent where it is adequate as a basis for an optimization where the result should resemble the actual movement of a cat.

The two models presented in [4] both take into account the body of the cat, but disregard the effect of the limbs. Our model is opposite of that in the sense that it disregards the bulk of the body and focuses on the limbs. Neither of the models fully captures all the features of the cat involved in the reorientation. Therefore, one should be careful when making biologic conclusions about the cat on the basis of either of the models, and it seems infeasible to make a refined model of that take both the body an the limps into account, given the complexity of these models. On the other hand, if the goal is to discover new principles for efficient reorientation, all three models will be relevant to study.

References

[1] A. M. Bloch. *Nonholonomic Mechanics and Control*. Interdisciplinary Applied Mathematics. Springer, 1. edition, 2003.

[2] Christof Büskens and Helmut Maurer. Sqp-methods for solving optimal control prroblems with control and state constraints: adjoint variables, sensitivity analysis and real-time control. *Journal of Computational and Applied Mathematics*, 120:85–108, 2000.

[3] C. K. Chen and N. Sreenath. Control of coupled spatial two-body systems with nonholonomic constraints. *Proceedings of the 32nd IEEE Conference on Decision and Control*, 2:949–954, december 1993.

[4] Michael J. Enos. On an optimal control problem on $so(3) \times so(3)$ and the falling cat. *Fields Institute Communications: Dynamics and Control of Mechanical systems, The Falling Cat and Related Problems*, 1993.

[5] T. R. Kane and M. P. Scher. A dynamical explanation of the falling cat phenomenon. *International Journal of Solids and Structures*, 5(7):663–670, 1969.

[6] Robert G. Littlejohn and Matthias Reinsch. Gauge fields in the separation of rotations and internal motions in the n-body problem. *Reviews of Modern Physics*, 69(1):213–274, januar 1997.

[7] Jerrold E. Marsden and Tudor S. Ratiu. *Introduction to Mechanics and Symmetry*. Texts in Applied Mathematics. Springer-Verlag, 1994.

[8] R. Montgomery. Isoholonomic problems and some applications. *Communications in Mathematical Physics*, 128:565–592, 1990.

[9] Alfred Shapere and Frank Wilczek. Gauge kinematics of deformable bodies. *American Journal of Physics*, 57(6):2051–2054, june 1989.

Path and environment information from relative crossings of landmarks

Benjamín Tovar* Fred Cohen† Steven M. LaValle*

Abstract

We describe our progress studying a minimal robot model, for a robot moving in the plane which is only able to sense the cyclic order of landmarks with respect to its current position. We model the landmarks as points in the plane. No metric information is available regarding the robot or landmark positions; moreover, the robot does not have a compass or odometers (e.g., coordinates). We establish its capabilities in terms of mapping the environment and accomplishing tasks, such as navigation. The algorithms are nicely characterized using the notion of *order type*, which is powerful enough to determine which points lie inside the convex hulls of subsets of landmarks. Some portions of this work were previously published in [7].

1 Introduction

In this paper, we consider a robot moving in the plane with very limited sensing: it knows only the cyclic ordering of landmarks as they appear from the robot's current position (no distance information can be measured and there are no other sensors). The information space is characterized using the concept of the *order type* of a configuration of points in the plane [4]. Given the sensor limitations, we avoid estimation of the position of the robot and of landmarks, and instead concentrate on the landmarks' relative orderings to construct the algorithms. This paper follows minimalist philosophy, which implies that we want the robot, its sensors, and its models to be as simple as possible. Although our problem has not been considered before, this philosophy has been successful in a number of works (e.g., [1, 2, 6, 3]). Is it really necessary for the robot to build an explicit representation of the environment? Is knowing the exact position of the robot crucial for the completion of the task? After establishing what the robot can learn from its simple sensor, we then illustrate the kinds of tasks that it can solve.

*University of Illinois, Dept. of Computer Science. Siebel Center. 201 N. Goodwin 3340, Urbana, IL. 61801. E-mail: {btovar,lavalle}@uiuc.edu

†University of Rochester, Dept. of Mathematics. Hylan Building. Rochester, NY 14627. E-mail: cohf@math.rochester.edu

[0]This work was partially supported by the DARPA SToMP program and the ONR grant N000014-02-1-0488.

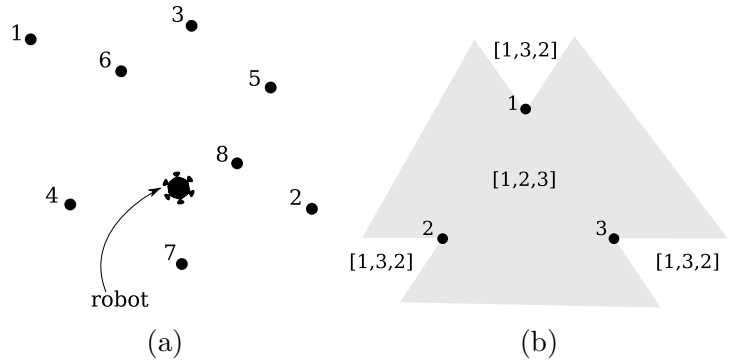


Figure 1: (a) The landmark order detector gives the cyclic order of the landmarks around the robot. Note that only the cyclic order is preserved, and that the sensed angular position of each landmark may be quite different from the real one. Thus, the robot only knows reliably, up to a cyclic permutation, that the sequence of landmarks detected is [7,2,8,5,3,6,1,4]. (b) Cyclic permutations of three landmarks. Purely by sensing, the robot cannot even know if it is inside the convex hull defined by the three landmarks. Nevertheless, the orientation of the triangle (the counterclockwise cyclic order of the landmarks as sensed from inside their convex hull) can be determined with an information state.

2 Model

The robot is modeled as a moving point in $\mathbb{R}^2$. The environment is modeled as the set E of n different labeled points in $\mathbb{R}^2$. For $p_i \in E$, for $1 \leq i \leq n$, p_i is referred to as the *landmark* with *label* i. The robot has a sensor, called the *landmark order detector*, and it is denoted with $\text{lod}_s(\mathbf{x})$, for $\mathbf{x} \in X$. The landmark order detector gives the counterclockwise cyclic order of the landmarks as seen from the current robot's position. This gives the robot the cyclic permutation of landmarks around it (see Figure 1.a). Note that no metric information is available to the robot. The robot does not have any coordinate estimate of its position, and the position of the landmarks. Moreover, we assume that the landmark order detector does respect the cyclic order of landmarks, but does not measure the angle between them. In other words, the sensor does not provide by itself any notion of front, back, left or right with respect to the robot. It is assumed, though, that the robot can choose a particular landmark label and move towards the landmark position. This motion is called *landmark tracking*, and it is denoted by track(i). For simplicity, we assume that the tracking ends when the robot arrives at the landmark. In the case of point landmarks, this means that the sensor no longer detects the landmark just tracked.

We assume that landmarks obstruct the visibility of the robot. That is, only the landmark closest to the robot is detected. In this paper we assume that the landmarks are in general position (no three point landmarks are collinear).

3 Order type and landmarks

The only information the robot receives is the changes in the cyclic permutations. For example, for three landmarks, only two sensing readings are possible. Purely by sensing, the robot cannot even know if it is inside the convex hull defined by the three landmarks (see Figure 1.b). Nevertheless, consider the robot traveling from the landmark labeled with 1 to the landmark labeled with 2. Since the reading from the landmark order detector follows a counterclockwise order, the robot can determine whether the landmark labeled with 3 is to the *left* or *right* of the directed segment that connects landmark 1 to landmark 2. Thus, the robot can combine sensing with action histories to recover some structure of the configuration of landmarks.

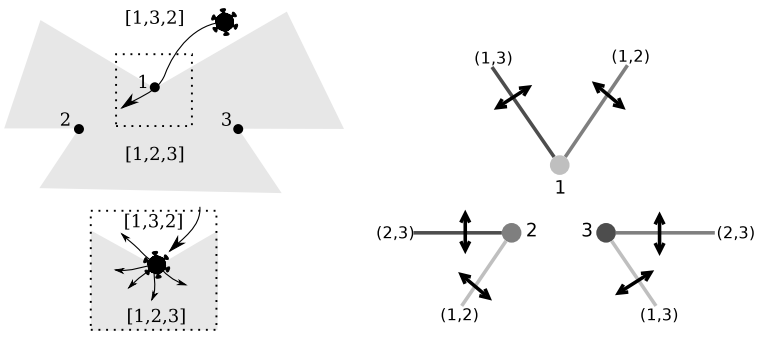
We generalize the previous idea to encode information states with the concept of *order type*. For a configuration of labeled points in the plane, the order type is defined as the relative orientation of every triple of points, that is, the signed area of every triangle with vertices in the set of points [4]. For points $p_i = (x_i, y_i)$, $p_j = (x_j, y_j)$, and $p_k = (x_k, y_k)$, this can be computed with the determinant $\det(p_i, p_j, p_k) = (x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i)$. We write $p_i p_j p_k^+$ if $\det(p_i, p_j, p_k) > 0$. The order type of the configuration of points can be encoded by a function defined as: $\Lambda(i, j) = \{k \mid p_i p_j p_k^+, \text{ for } p_i, p_j, p_k \in P\}$.

The function Λ takes the indices i, j of two points $p_i, p_j \in P$, and returns the indices corresponding to the points in $P \setminus \{p_i, p_j\}$ positively oriented with respect to p_i and p_j (in that order). For example, following Figure 1.a, $\Lambda(3, 7) = \{2, 5, 8\}$, and $\Lambda(7, 3) = \{1, 4, 6\}$. Alternatively, the order type can be specified with the function $\lambda(i, j) = |\Lambda(i, j)|$. It is not immediately clear that once the function λ is known, Λ can be deduced. It also follows that when $\lambda(i, j) = 0$, then there are no points to the left of the directed edge $\overline{p_i p_j}$, and both p_i and p_j belong to the boundary of the convex hull.

The order type definition is extended naturally to our point landmark framework, using the landmark labels as the indices for Λ. Of course, the robot cannot compute the determinants, because it lacks any coordinates. Nevertheless, it is possible to compute Λ for any pair of landmark labels. The value of $\Lambda(i, j)$ is determined as follows. The robot is commanded to track landmark p_i until p_i disappears (the robot is at p_i). Next, the robot is commanded to track p_j, and at the moment p_i is detected again, the robot is guaranteed to be on $\overline{p_i p_j}$, pointing towards p_j. From the sensor reading in this position, $\Lambda(i, j)$ and $\Lambda(j, i)$ can be found found.

4 Solving robotic tasks

In this section we present some tasks that can be solved using the concepts presented in previous sections. In the following examples, L is the set of landmarks detected in the environment E, and $n = |L|$.



(a) (b)

Figure 2: (a) Orientation error. A small control error may find the wrong orientation for the triangle. On the bottom, if the robot follows the top-left arrow, the orientation is not computed correctly. (b) Swap lines. Crossing a half-line swaps the order of the respective landmarks in the reading of the landmark order detector. Such half-lines are called *swap lines*.

4.1 Landmarks inside a triangle

The task in this section is to compute the subset of landmarks of L that are inside of the triangle defined by the landmarks labeled with i, j and k. In other words, if $k \in \Lambda(i, j)$, the robot should determine $\Lambda(i, j) \cap \Lambda(j, k) \cap \Lambda(k, i)$, or if $k \notin \Lambda(i, j)$, then $\Lambda(j, i) \cap \Lambda(i, k) \cap \Lambda(k, j)$ should be computed. These two cases correspond to the two possible orientations of a triangle, as defined before with the determinant. Since both the orientation of the triangle and the needed values of Λ can be computed easily as explained before, we use this simple example to introduce a motion strategy that deals with control uncertainty. Refer to Figure 2.a. The problem here is that the internal angle of the triangle at landmark i is obtuse. This gives little margin of error for the control, and the triangle orientation may not be computed correctly. As it can be seen for landmarks j and k, with acute angles, the error in the control should be almost π before the orientation is computed incorrectly. Given that a triangle has at most one obtuse angle, the robot repeats the orientation procedure three times, one for each edge of the triangle. If in this strategy an orientation is found more than once, it is taken as the correct orientation of the triangle. This strategy allows for a control error in the direction of the robot up to $2\pi/3$.

4.2 Navigation

In our framework, a navigation goal is a sequence g of landmark labels. Formally, the *navigation* task is defined as follows: *Move the robot such that a state* $\mathbf{x}$ *with* $lod_s(\mathbf{x}) = g$ *is reached. Report if* g *cannot be attained given the configuration of the landmarks in the plane.*

Before describing the navigation algorithm, we need to describe what is achieved by moving

the robot to a place where a particular permutation is sensed. For this purpose, consider the partition of the plane in which locations inside the same cell generate the same reading in the landmark order detector. This can be considered as an aspect graph [5], in which a cyclic permutation is an *aspect* of the configuration of landmarks. The decomposition is determined by half-lines, that if crossed, generate a change in the permutation order of a pair of landmarks in $\text{lod}_s(\mathbf{x})$. Each pair of landmarks generate a pair of half-lines, which are referred to as *swap lines* (see Figure 2.b).

Given that the robot cannot travel outside hull(L), the navigation task is only defined for cells whose intersection with hull(L) is not empty. The navigation task is only meaningful if different cells generate different cyclic permutations for the landmark order detector. To prove this, the following Lemma is proposed:

Lemma 4.1. *Let C be the set of cells of the decomposition generated by the swap lines that intersect hull(L), and let $C_i, C_j \in C$. If C_i and C_j are not the same cell, and if they are bounded by the same swap line m, then they generate different readings in the landmark order detector.*

Proof. Let $(s(a), a)$ and $(s(a'), a')$ be the landmarks that generated m. Consider a motion of the robot from C_i to C_j in a straight line arbitrarily close to m. This makes labels $s(a)$ and $s(a')$ to appear consecutive in $\text{lod}_s(\mathbf{x})$ for the duration of the motion. Since C_i and C_j are different, then at least one swap line intersects m between cells C_i and C_j. Let such swap line be generated by landmarks $(s(b), b)$ and $(s(b'), b')$. Crossing this line swaps the order of $s(b)$ and $s(b')$. This swapping could be reverted if the other swap line generated by $(s(b), b)$ and $(s(b'), b')$ is crossed, or if one of $(s(b), b)$ or $(s(b'), b')$ swaps with all the other landmarks. The first situation is not possible, since both swap lines lie in the same line, and m can only intersect one of them. The other case implies that $s(a)$ and $s(a')$ are at some instant not consecutive in $\text{lod}_s(\mathbf{x})$. This is not possible by traveling arbitrarily close to m. Thus, the readings of $\text{lod}_s(\mathbf{x})$ from C_i and C_j will differ in at least a pair of landmarks. □

The next theorem states that the landmark order detector generates different readings for cells intersecting the convex hull of the configuration of landmarks.

Theorem 4.1. *Let C be the set of cells of the decomposition generated by the swap lines that intersect hull(L). Then for any two different cells C_i and C_j, the cyclic permutations generated by $lod_s(\mathbf{x})$ when the robot is inside C_i or C_j are different.*

Proof. By induction on the the number of landmarks $n = |L|$. When $n = 3$, there is a single cell intersecting hull(L). For $n > 3$, assume the statement is true for n landmarks. Then, for $n + 1$, adding the new landmark generates $2n$ swap lines, some of which stab cells in C. Cells stabbed by the same swap line will have different cyclic permutations, by Lemma 4.1. Since the new landmark does not change the relative ordering of any other three landmarks,

by the induction assumption, cells that do not share one of the new swap lines will also have different permutations. □

By Theorem 4.1 it is known that different "places" will have different cyclic permutations associated. Given that the robot does not have the landmarks coordinates, the exact geometrical decomposition cannot be constructed. Nevertheless, the robot can navigate such that a particular cyclic permutation g appears in the landmark order detector. First of all, the robot has to decide if g is attainable in the configuration of landmarks. This can be decided using Λ as follows:

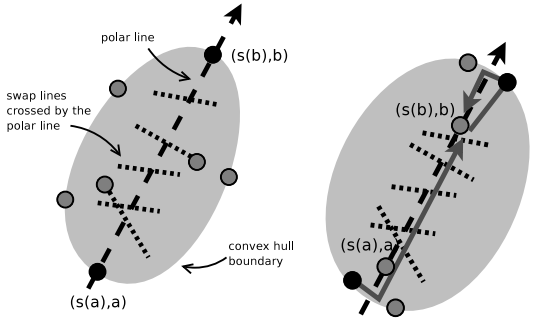
Lemma 4.2. *Let $g = [s(p_1), s(p_2), \ldots, s(p_n)]$, and let $g(s(p_i), s(p_j))$ be the set whose elements are the elements of the subsequence of g starting at $s(p_{i+1})$, ending at $s(p_{j-1})$. If g is attainable in the configuration of landmarks of L, then for each landmark label $s(p_i)$ there is a landmark label $s(p_j)$ such that $\Lambda(s(p_i), s(p_j)) = g(s(p_i), s(p_j))$. The directed line passing from p_i to p_j is called the polar line of (i, p_i), and (j, p_j) is called a pole of (i, p_i).*

Proof. Suppose that g is attained when the robot is at point p. Consider the line m passing through p and the landmark at p_i. Now rotate m clockwise, with p_i as a pivot, until m hits another landmark, say $(s(p_j), p_j)$. We have that $\Lambda(s(p_i), s(p_j)) = g(s(p_i), s(p_j))$, otherwise, the order required for g is not attained ((i, p_i) or (j, p_j) would appear in the wrong place according to g). □

While the region in which g is attained is bounded by polar lines, not all polar lines intersect such region. However, a landmark and its pole appear consecutive in $\text{lod}_s(\mathbf{x})$ if the corresponding polar line bounds the goal region. Thus, the search for the goal permutation g is reduced to such polar lines. Suppose that a polar line is determined by landmarks $(s(a), a)$ and $(s(b), b)$. If both of them belong to the boundary of the convex hull, then the robot traverses the line segment $\overline{ab}$. This is done, for example, by tracking $(s(a), a)$ and then tracking $(s(b), b)$ (see Figure 3). If the landmarks do not belong to the boundary of the convex hull, the intersection of the polar line with the convex hull is found by the robot traveling in the boundary of the convex hull, until landmark labels $s(a)$ and $s(b)$ swap places. At this point, the robot tracks any of the landmarks, which traverses at the same time the polar line. Note that the robot may not need to traverse the convex hull boundary to find this intersections, since this information may be already available from Λ. Note also that the tracking takes place once $s(a)$ and $s(b)$ swap places in $\text{lod}_s(\mathbf{x})$, thus the robot travels arbitrarily close, but not exactly on the polar line.

References

[1] R. C. Brost. *Analysis and Planning of Planar Manipulation Tasks.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1991.



(a) (b)

Figure 3: The two general cases for navigation in the polar lines. In (a) the polar line is determined by two landmarks in the boundary of the convex hull, and no further computation is required. In (b), the intersection of the polar line with the boundary of the convex hull is found by a change in the order of the landmarks which determine the polar line. With this, the robot is able to traverse the polar line.

[2] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Trans. Robot. & Autom.*, 4(4):369–379, August 1988.

[3] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[4] J. E. Goodman and R. Pollack. Multidimensional sorting. *SIAM Journal on Computing*, 12(3):484–507, August 1983.

[5] J.J. Koenderink and A.J. van Doorn. The singularities of the visual mapping. *Biological Cybernetics*, 24:51–59, 1976.

[6] T. S. Levitt and D. T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44(3):305–360, 1990.

[7] B. Tovar, L. Freda, and S. M. LaValle. Mapping and navigation from permutations of landmarks. In *AMS Contemporary Mathematics Proc*, 2006.

Robot manifolds for direct and inverse kinematics solutions

Bruno Damas* Manuel Lopes†

Abstract

We present a novel algorithm to estimate robot kinematic manifolds incrementally. We relate manifold learning with the forward and inverse kinematic of robots. The learned structure encodes this functions and so it is possible to recover them from the manifold. Our algorithm works without any knowledge of the robot kinematics and can potentially work in highly-dimensional spaces. We present some simulated examples to validate our approach.

1 Motivation

The direct and inverse kinematics are fundamental functions for robot control. There are already closed form solutions for the kinematics of most commonly used robotic manipulators, mainly the direct kinematics of serial robots and for the inverse kinematics of parallel robots when the number of degrees of freedom is less than seven [2, 9]. Serial mechanisms forward kinematics have closed form solutions, commonly obtained using the Denavitt-Hartenberg notation. A closed form for the inverse kinematics, on the other hand, is usually more difficult to get: numerical methods [12, 2] or a general approach described in [12] are widespread alternatives. Conversely, in parallel mechanisms, the inverse problem is many times trivial while the direct one is usually not that obvious.

Nowadays this topic is experiencing a renewal research interest mainly due to the existence of humanoid robots with a large number of degrees of freedom. Robots with more than forty degrees of freedom are becoming common in recent years [5, 6, 4]. In this high-dimensional setting, planning a trajectory or computing the kinematics usually becomes a difficult task. Such redundant robots can solve the same task in many different ways and the main difficulty is selecting the best option available.

*Instituto de Sistemas e Robótica, Instituto Superior Técnico, Portugal, and Escola Superior de Tecnologia de Setúbal, Portugal. E-mail:bdamas@isr.ist.utl.pt.

†Instituto de Sistemas e Robótica, Instituto Superior Técnico, Portugal. E-mail:macl@isr.ist.utl.pt. This work was partially supported by EU Project RobotCub and by the Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) through the POS_Conhecimento Program that includes FEDER funds.

Obtaining a closed form solution for the inverse or forward kinematics is very difficult when the number of degrees of freedom increases. This limitation strongly motivates to directly *learn* these functions from real data obtained during robot operation. Robot kinematics, however, is not usually an injective function: this implies that even if the forward kinematics is known, the inverse kinematics might be impossible to extract from it. And the same situation can occur if we want to evaluate the forward kinematics by having the inverse kinematics. This occurs because multiple outcomes can result from a single input.

In this paper we present an algorithm that can learn a model representing simultaneously the forward and inverse kinematics. Instead of learning both maps separately, we learn the manifold representation of the joint input-output space. Our algorithm is incremental and so every new sample can be used to improve the model. From the learned model we can recover both forward and inverse kinematics. The proposed algorithm can be divided in two different steps:

- An online algorithm that learns input-output restrictions of a generic smooth map;
- A method that, given a partial set of input-output variables, provides an estimate of the remaining ones, using the learned restrictions.

2 Problem formulation

This paper presents a new algorithm to learn the kinematic model of a generic robot. The key point of our approach is to consider the problem from an unsupervised point of view, where data points consist of vectors containing *both* controlled and observed variables, thus allowing to easily recover the relation among any set of variables. These vectors define a surface that can be seen as the graphic of a function. Consider D_c the number of controlled — or independent — variables and D_o the number of observed variables. A point belonging to the robot kinematic manifold $\mathbf{x}$ in a $D = D_c + D_o$ dimensional space will lie in a sub-space of dimension D_c. This manifold can be represented by the implicit function:

$$\mathbf{H}(\mathbf{x}) = \mathbf{0}\ , \tag{1}$$

where $\mathbf{H}(\mathbf{x})$ imposes the $D - D_c$ restrictions arising from kinematics considerations.

Learning the complete manifold can provide significant advantages over other supervised learning based techniques. It will be shown that such knowledge can provide both the forward and inverse kinematics in a straightforward manner, and can as well supply any other restrictions among controlled and observed variables. The proposed algorithm is also able to deal with situations where multiple outcomes arise from a single input. Note that all this information can be obtained with no necessity to conduct further learning, *i.e.*, one can easily

change from an inverse kinematics problem to a forward one maintaining the manifold previously learned. The manifolds resulting of several sensory-motor coordination task have been studied in [7].

Unsupervised learning of a D_c-dimensional manifold in a D-dimensional space can be interpreted as a probability density estimation problem: given a set of (possibly corrupted with noise) sample points $\mathbf{x}_i$ belonging to the manifold, $i = 1 \dots N$, estimate the probability of a point $\mathbf{x}$ belonging to the manifold, *i.e.*,

$$p(\mathbf{H}(\mathbf{x}) = \mathbf{0} \,|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \,. \tag{2}$$

For small regions the manifold can be approximately described by a hyperplane with dimension D_c. The covariance of a set of data points belonging to a small neighborhood provides a description of the manifold around that location. The D_c first principal components (in the sense of Principal Component Analysis (PCA)) of the covariance span a hyperplane given a local estimate of the manifold in that point. The remaining PCA dimensions can be neglected as they represent the noise affecting data.

In this paper the robotic kinematic manifold will be approximated by a collection of M models, where each model describes the manifold in the vicinity of its center μ_m by a local covariance matrix $\mathbf{C}_m$. This gaussian mixture representation provides a good estimate of the manifold if there are sufficient local models to appropriately cover the entire manifold, so that in the domain of each model the true manifold is approximately linear.

The quality of estimates obtained using the proposed learning scheme depends severely on the manifold learning mechanism. *ISOMAP* [11] and *LLE* [10] are two standard methods to do generic manifold estimation and both provide convergence proofs. Unfortunately, these methods need to gather all data in an offline fashion before being able to perform the estimation. Also, they do not provide a parametric model, not even a local one, making available only the manifold coordinates of the points in the dataset: for new data points a metric must be used to interpolate among the neighbors.

There are two major issues to address in this paper: first, the location and covariance of each model must be estimated. This is a classical unsupervised learning problem, where the parameters of the mixture are to be estimated. The manifold estimate can be used to obtain the direct and inverse robot kinematics, i.e. given an actuation value we would like to have an estimate of the observed variables, or, inversely, obtain the actuators position that leads to a desired observation.

3 Learning

One of the most popular methods to estimate a mixture of Gaussians is the expectation-maximization algorithm (EM) [3]. It is a method prone to local minima, specially if the data

dimensionality is high. In its original formulation EM is a batch algorithm; the sequential nature of data acquisition in real robots, however, suggests modifying it to an online version, making it able to provide estimatives while acquiring new data. This can be easily done if we keep some memory traces along the learning process, as described in [8] (to appear soon). Once in possession of a collection of local models describing the kinematics manifold, a neighborhood between models can be obtained, allowing us to define a rough measure of distance between points along the manifold. This will be very useful when the estimation process can generate multiple outcomes.

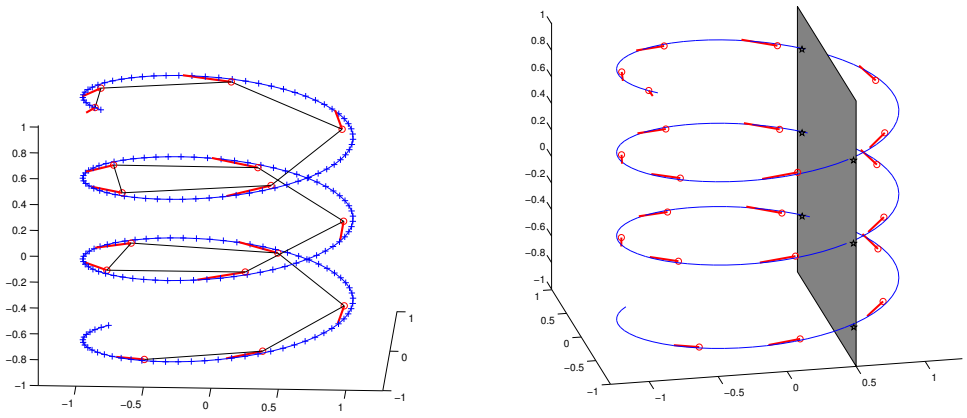
Suppose data points $\mathbf{x}$ are divided into a query component and an answer component, $\mathbf{x} = [\mathbf{x}_q^T \ \mathbf{x}_a^T]^T$, such that $D_q + D_a = D$, where D_q is the query dimension and D_a is the answer dimension, not necessarily equal to D_c and D_o. The answer component is the set $\mathbf{x}_a$ of elements of $\mathbf{x}$ to be estimated given a specific value of the remaining elements $\mathbf{x}_q$. For instance, for a forward kinematics problem $\mathbf{x}_q$ corresponds to the actuation variables, while for an inverse kinematics problem $\mathbf{x}_q$ matches the observed variables.

The key point to estimate $\mathbf{x}_a$ is to understand that, for a specific local model and for a given value of $\mathbf{x}_q$, $\mathbf{x}_a$ should be the value that maximizes the likelihood of the whole data point $\mathbf{x}$, given the model. This can be achieved by minimizing the corresponding Mahalanobis distance to the center of the model, using the respective covariance. Calculation are simple, and details can be found in [8] (to appear soon).

After obtaining an estimate $\hat{\mathbf{x}}_a$ for each local model we can merge these local solutions to obtain the definitive solution. These local solutions can be combined, for instance, using a weighted average. Multiple answers can also be obtained, for non-injective input-output relations, using the neighborhood between models to group local models into different solutions. Once again, further details can be found in [8].

4 Experiments

We performed several simulations to assess the quality of the proposed algorithm. Figure 1(a) shows a one dimensional manifold in a three dimensional space. As we can see, after the learning step the distance along the manifold prevails over the Euclidean distance between model centers when defining the neighborhood relationships. Figure 1(b) shows how correctly are estimated the multiple outcomes for $\mathbf{x}_q = 0.5$. Note that any of the remaining two dimensions could also be used as the query variable.



(a) (b)

Figure 1: (a) Learned manifold ($D = 3$, $D_c = 1$). Red circles represent model centers, red lines represent the principal components of the covariance for each model, black lines represent the neighborhood relations between models and blue crosses stand for the sample points. (b) Recovering the forward model embedded in the manifold. With $x_q = 0.5$ the six possible outcomes are successfully estimated (represented in the figure by black asterisks).

5 Conclusions

We presented a general algorithm to learn the manifold resulting from robotic kinematic structures. This algorithm is computationally very efficient: the most time consuming operations are local models covariance matrices inversions that must be performed each time a new point is incrementally incorporated by the learning mechanism. Considering only a subset of the nearest models in this update step can significantly speed up the process when the number of models starts to grow. Note also that in high dimension spaces the inversion of the updated covariance matrix can be done efficiently using the Woodbury identity.

This online learning algorithm can provide a robot with real-time learning without a previous acquisition of data, and it does not need any previous information about the underlying manifold. The convergence of this modified EM algorithm is a key issue when dealing with high dimension data. So far, the proposed learning algorithm can only deal efficiently with low dimension data. As N increases, the solution tends to be stuck in poor local maxima of the likelihood function. Also, it becomes increasingly difficult to tune the few parameters of the algorithm. Work is currently being done to deal with these severe limitations. It should be stressed, however, that the major claim of this work is the way we can efficiently use a learned mixture of gaussians in the product space of the actuated and observed variables to infer both the forward and backward kinematics. Such a scheme can even be used to infer

a mixture of both controlled and observed variables, and has enough flexibility to be easily replaced by any other mixture of gaussians learning method (see, for example, [1]). The proposed method deals naturally with the classical problem of redundancy and non-injectivity in the forward-backward maps, being able to extract multiple solutions when multiple answers do exist. These solutions can then be used by a higher level algorithm to choose the final solution, *e.g.*, taking into account obstacle avoidance or energy minimization schemes.

References

[1] Matthew Brand. Charting a manifold. *Advances in Neural Information Processing Systems*, 15:985–992, 2003.

[2] John J. Craig. *Introduction to Robotics*. Addison-Wesley Pub Co, 1989.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society B*, 39:1–38, 1977.

[4] Fujitsu. Humanoid Robot HOAP-2. http://www.automation.fujitsu.com, 2003.

[5] L. Geppert. Qrio, the robot that could. *IEEE Spectrum*, 41(5):34–37, May 2004.

[6] M. Hirose, Y. Haikawa, T. Takenaka, and K. Hirai. Development of humanoid robot ASIMO. In *Workshop on Exploration towards Humanoid Robot Applications at IROS*, Hawaii, USA, 2001.

[7] R.A. Peters II and O.C. Jenkins. Uncovering manifold structures in robonaut's sensory-data state space. In *IEEE-RSJ International Conference on Humanoid Robotics*, Tsukuba, Japan, December 2005.

[8] Manuel Lopes and Bruno Damas. A learning framework for generic sensory-motor maps. In *International Conference on Intelligent Robots and Systems*, San Diego, USA, 2007.

[9] J. P. Merlet. *Parallel Robots*. Springer, 2006.

[10] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, december 2000.

[11] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[12] Lung-Wen Tsai. *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. Wyley-Interscience, 1999.

Optimal control of cooperative multi-robot systems using mixed-integer linear programming

Christian Reinl* Oskar von Stryk*

Abstract

A new planning method for optimal control of multi-robot systems is discussed which accounts for the (continuous) physical locomotion dynamics of the robots and its tight coupling to the distribution and allocation of (discrete) subtasks to the robots to fulfill a joint mission. The point of departure is a nonlinear and nonconvex hybrid optimal control problem (HOCP) formulation which incorporates a detailed hybrid automaton model. Because of the many difficulties involved in solving this problem like large computational times and the lack of good or global convergence properties it is transcribed into a mixed-integer linear program (MILP). This can be solved much more efficiently using existing algorithms. The proposed approach is outlined for an example problem of cooperative soccer robots. The MILP solution itself may serve either as a good initial solution estimate for a method addressing the nonlinear HOCP or may later become the kernel of a model predictive control method for cooperative multi-robot systems. Despite the promising results obtained so far a variety of open questions yet remains to be answered including the "best" way of transcribing HOCP to MILP with respect to both computational efficiency and good HOCP solution approximation.

1 Introduction

In this paper multi-robot systems are considered where the individual nonlinear physical motion dynamics is of fundamental importance for the mission success which depends on optimizing physical values like the robots' positions or energy consumption.

The problem's possible combinatorial character complicates an analytical inspection and hardly theoretically proved results are available for the most general problem formulation. The key idea in this paper is to build up a centralized MILP-based (cf. [3]) controller for multi-vehicle systems, starting with a systematic HOCP description (Sect. 2) and a consistent transformation (Sect. 3) towards optimization based control (Sect. 4). Tight coupling of discrete states (e.g. actions) an respective continuous state variables (positions, velocities etc.) is a basic feature in there. Especially in an uncertain setting (failures, uncontrollable

*Technische Universität Darmstadt, CS Dept., E-mail: {reinl, stryk}@sim.tu-darmstadt.de.

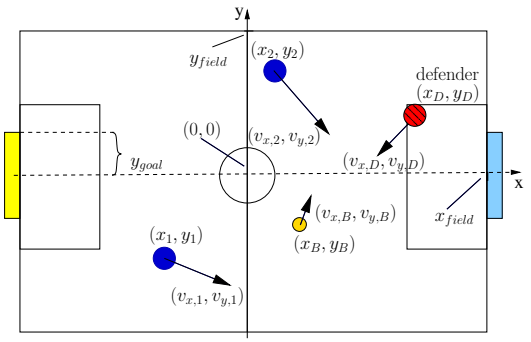


Figure 1: Setting of the soccer benchmark problem

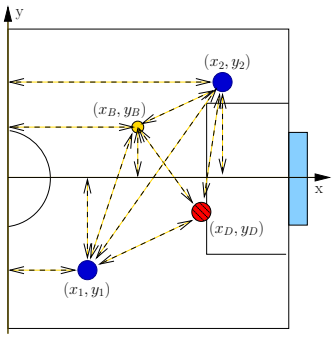


Figure 2: Contributions to objective

objects), the robustness of MILP offers an efficient (cf. [2]) way to be used in receding horizon controllers. To illustrate this approach we refer to a benchmark problem from robot soccer (cf. Fig. 1) with two strikers versus one (passive) defender, all modeled as moving point masses. The intention is to find the control which optimizes the attackers' chances for a considered time horizon $[t_0, t_f]$. Results for this representative example will be given in Sect. 4.

2 Modeling the cooperative multi-robot system

We are considering (in-)direct controllable and not controllable moving objects i in our system. Each one is characterized by its continuous dynamic state $\boldsymbol{x}_i$ (e.g. position, velocity,...) and a discrete value q_i that denotes a certain subtask or role. Together with the continuous control variable $\boldsymbol{u}_i$, the continuous state evolves subject to $\dot{\boldsymbol{x}}_i = \boldsymbol{f}_{q_i,i}(\boldsymbol{x}_i, \boldsymbol{u}_i)$. By defining (usually unknown) switching times t_s and corresponding specifications, how to connect $\boldsymbol{x}_i$ when q_i switches at t_s, the individual trajectory for an object is determined.

For the regarded soccer example, $i \in \{1, 2, B, D\}$ denotes two strikers, one ball and a defender. As modes of motion q_i for the strikers we distinguish `free_moving` and `dribbling`.

2.1 Modeling switched dynamics with hybrid automata

We are regarding multi-robot systems consisting of moving objects with specific modes of motion and rules that define feasible sequences for them. Thus we are using hybrid automata to describe the cooperative system. They are well established in the context of robot control.

A hybrid automaton [5] $H = (V, E, \mathbb{X}, \mathbb{U}, ini, \mathbf{f}, \mathbf{j}, \mathbf{i}, \mathbf{e})$ consists of a finite directed multi-graph (V, E) with knots in V (called states) and edges in E (so-called switches), a set of continuous state variables $\mathbb{X}$, a set of continuous control variables $\mathbb{U}$, a map ini which assigns an initial condition to each edge, the invariants provided by the map $\mathbf{i}$ which assigns each knot with a feasible region for the continuous states and controls using equality and inequality constraints, a map $\mathbf{f}$ which assigns a flow equation or state dynamics to each state, a map $\mathbf{j}$ which assigns jump conditions to edges and a map $\mathbf{e}$ which assigns events to edges which occur at switches. For the proposed soccer application (cf. Fig. 3) we added another hierarchy

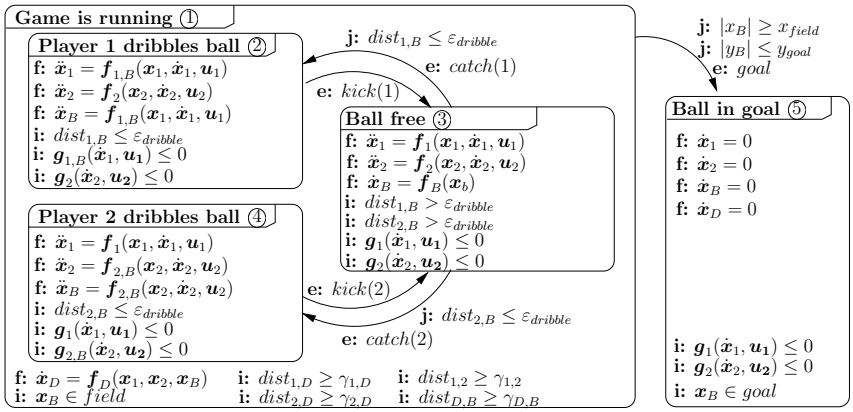


Figure 3: Hierarchical hybrid automaton model of the switched motion dynamics

there that contains conditions that are similar in the covered knots. In this model we only distinguish whether the ball is dribbled, rolls free or is inside the goal. The respective motion dynamics of a dribbling robot is indexed by "B". The initial conditions ini are defined with the position $\boldsymbol{x}_i(t_0)$ of the objects i. Catching and kicking a ball are modeled by events

$$kick(i): \ \dot{\boldsymbol{x}}_B(t_s + 0) = 3 \cdot \dot{\boldsymbol{x}}_i(t_s - 0)\,, \ catch(i): \ \boldsymbol{x}_B(t_s + 0) = \boldsymbol{x}_i(t_s - 0)\ , \tag{1}$$

$t_s \pm 0 := \lim_{\epsilon \to 0, \epsilon > 0} t_s \pm \epsilon$. All other state trajectories are required to be continuous at t_s.

The auxiliary variable $dist_{i_1,i_2}$ represents a distance measure between objects i_1, i_2 and is used to express collision avoidance (with a constant γ_{i_1,i_2}). Further constraints on state and control variables according to the specific motion modes are modelled as invariants **i**.

The dynamic of the defender is not considered to be switched here. We tested our approach with a simple model for the dynamic of robots and ball and set for all states except ⑤

$$\textbf{f: } \dot{\boldsymbol{x}}_B(t) = \boldsymbol{v}_B(t), \ \textbf{f: } \ddot{\boldsymbol{x}}_\Diamond(t) = \begin{pmatrix} \ddot{x}_\Diamond(t) \\ \ddot{y}_\Diamond(t) \end{pmatrix} = \begin{pmatrix} \dot{v}_{x,\Diamond}(t) \\ \dot{v}_{y,\Diamond}(t) \end{pmatrix} = \boldsymbol{u}_\Diamond(t) = \begin{pmatrix} u_{x,\Diamond}(t) \\ u_{y,\Diamond}(t) \end{pmatrix} \ (\Diamond \in \{1, 2\}) \tag{2}$$

For a dribbling robot the upper bound on its velocity is reduced by a factor $c^U_{v,dr}$.

The numerical method proposed in [4] for the general HOCP formulation uses piecewise polynomials and binary variables to transfer the hybrid automaton and an objective function into a finite dimensional sparse non-linear mixed-binary optimization problem. It was solved with a combination of sequential quadratic programming and Branch-and-Bound techniques. More details and results for this model are given there. Now we are interested in a MILP-formulation that provides a efficient optimization-based control considering the basic system characteristics.

2.2 The linearized model

We are introducing a fixed (not necessary) equidistant grid of time points with the sampling time $\boldsymbol{T}_s := t_k - t_{k-1}$. For the evolution of the continuous state and control variables we are defining $\boldsymbol{x}(k) := \boldsymbol{x}(t_{k+1})$ and $\boldsymbol{u}(k) := \boldsymbol{u}(t_{k+1})$. The knots in the describing automaton can

switch only at these time points and are not free any more. All (in-)equalities that were used to describe the different states and the motion dynamics in the systems will be reformulated or transformed into linear expressions now. Thus the differential flow conditions **f** must be reformulated as difference equations

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_{q,i}(\boldsymbol{x}, \boldsymbol{u}) \quad \rightsquigarrow \quad \boldsymbol{x}(k+1) = A_{q,i} \cdot \boldsymbol{x}(k) + B_{q,i} \cdot \boldsymbol{u}(k) \ . \tag{3}$$

Additional state variables and binary variables may be necessary here for case differentiations and combination of these cases with logical expressions. In the context of hybrid automata, this case differentiations are treated as new subknots. This splitting up strongly depends on the nonlinearity of the expression and the desired accuracy of the transformed model.The regions defined by the invariants **i** are approximated in a polygonal manner. Linear inequalities are combined logically therefore. If nonlinear expressions occur in the jump conditions **j** of events **e**, they have to be treated respectively. Afterwards, the automaton is clocked and covers only linear expressions and logical constraints. Application to the example results in

$$\begin{aligned} x_{\Diamond}(k+1) &= x_{\Diamond}(k) + T_s\, v_{x,\Diamond}(k) \ , \quad & v_{x,\Diamond}(k+1) &= v_{x,\Diamond}(k) + T_s\, u_{x,\Diamond}(k), \\ x_B(k+1) &= x_B(k) + T_s\, v_{x,B}(k) \ , \quad & v_{x,B}(k+1) &= \begin{cases} v_{x,\Diamond}(k) & \text{(dribbling)} \\ c_{trac}\, T_s\, v_{x,B}(k) & \text{(ball free)} \end{cases} \end{aligned}$$

($\Diamond \in \{1, 2\}$, $c_{trac} \cdot T_s < 1$, y_i, $v_{y,i}$, analogously). A simple, reactive defender that is always moving towards the current ball position is modeled by

$$x_D(k+1) = x_D(k) + Ts\, v_{x,D}(k) \ , \quad v_{x,D} = \frac{v^U_{x,D}}{D_{max}}(x_b(k) - x_D(k)), \tag{4}$$

($D_{max} \geq \max_{x,y,k}\{|x_b(k) - x_D(k)|, |y_b(k) - y_D(k)|\}$, y_D, $v_{y,D}$, analogously). The constant $v^U_{x,D}$ is the upper bound for $|v_D|$. In the investigated example the controls and velocities are constraint by quadratic expressions. Generally expressions of the form $\pm\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \leq \pm r$ can be transformed by using $n_\gamma \geq 4$ linear expressions

$$\pm \sin(\frac{i}{3}\pi)\,(x_1 - x_2) \pm \cos(\frac{i}{3}\pi)\,(y_1 - y_2) \leq \pm r \ (i = 1, \ldots, n_\gamma) \ . \tag{5}$$

Thus the invariants **i:** $\boldsymbol{g}_{q_i,i}(\dot{\boldsymbol{x}}_i) = ||(v_{x,i},\, v_{y,i})^T||_2 - v^U_{q_i,i} \leq 0$ were reformulated ($v^U_{q_i,i}$ constant, for $\boldsymbol{u}_{q_i,i}$ respectively). For the distance $dist_{i_1,i_2}$ between two objects as an auxiliary state variable the column-sum norm was used.

3 Transforming the model into a mixed-integer linear program

For each knot and each edge of the (hierarchical) automaton a time-dependent binary variable $b(t)$ is introduced so that $b = 1$ iff the state or edge is active. The structure then is transcribed with simple linear inequalities, e.g. $b_{(2)}(k) + b_{(4)}(k) + b_{(3)}(k) \leq 1$, $b_{(2)}(k) + b_{(4)}(k+1) \leq 1$,etc.

Logical relations combined with inequalities are translated using the 'Big-M'-technique (cf. [1]). Thus flows and invariants get connected with the respective binary variable, e.g.

$$\texttt{IF } b_{(3)} = 1 \texttt{ THEN } v_{x,B}(k+1) = c_{trac}\, v_{x,B}(k) \;\Leftrightarrow\; \begin{cases} (1-b_{(3)})m \le v_{x,B}(k+1) - c_{trac}\, v_{x,B}(k) \\ v_{x,B}(k+1) - c_{trac}\, v_{x,B}(k) \le (1-b_{(3)})M \end{cases}$$

$M \ge \max\{v_{x,B}(k+1) - c_{trac} v_{x,B}(k)\},\ m \le \min\{v_{x,B}(k+1) - c_{trac} v_{x,B}(k)\}$ constant.

To rate the quality of a computed attack, we mainly look at the situation at the final time t_{N+1} and primarily regard the following components (cf. Fig. 2). The positions of the attacking robots and the ball $(x_i(k), y_i(k))^T$, distances between the robots, defender and ball $dist_{i_1,i_2}(k)$ and also the events "*ball in goal*" and "*one robot dribbling*" $b_{(5)}$, $b_{(2)}$, $b_{(4)}$. With carefully determined coefficients then the objective function J is implemented as a weighted sum. Due to remaining degrees of freedom, $\boldsymbol{u}_i(k)$ is further added to it. The intention is to minimize J where the tactical behavior of the team is varied with the coefficients in J.

4 Optimization

Results for the linear implementation of the proposed benchmark problem with the parameters

$$\begin{array}{rclrclrclrclrcl} T_s &=& 0.8, & N &=& 10, & x_{field} &=& 270, & y_{field} &=& 180, & y_{goal} &=& 40, & \varepsilon_{dr} &=& 5, \\ \gamma_{2,D} &=& 30, & \gamma_{B,D} &=& 30, & D_{max} &=& 700, & c_{trac} &=& 0.88, & v^U_{B,x} &=& 135, & c_{v,dr} &=& 60.7, \\ v^U_{\Diamond,y} &=& 45, & u^U_{\Diamond,x} &=& 40, & u^U_{\Diamond,y} &=& 40, & v^U_{\Diamond,y} &=& 90, & v^U_{\Diamond,x} &=& 45, & & & \end{array}$$

and the objective function

$$J = \sum_{k=1}^{N} \left(-0.2\, dist_{B,D} - 320\, b_{(5)} + 0.001 \sum_{i\in\{1,2\}} (\,|u_{x,i}| + |u_{y,i}|\,) \right)\Bigg|_{t=k} + \left(-x_B + 0.6\,|y_B| - 0.15\, dist_{B,D} - 0.01\, dist_{1,2} - 160\,(b_{(2)} + b_{(4)}) + 0.02 \sum_{i\in\{1,2\}} (-x_i + |y_i|) \right)\Bigg|_{t=N+1} . \tag{6}$$

are given. The MILP was solved with CPLEX 10.0 (from ILOG, Inc.) on a PC (Intel(R) Pentium(R) M processor 1.86GHz; 1024 MB RAM) in 30 sec (see Fig. 4 for details).

5 Conclusion and outlook

A MILP formulation has been developed which accounts for the tight coupling of discrete decisions and continuous flow variables in optimal control of cooperative mobile robot systems. A consistent modeling towards a linearized formulation was shown. The numerical approach is applicable to a wide range of scenarios. Ongoing work considers techniques to improve the MILP by decoupling and additional constraints. Also various methods to linearize the given nonlinear description more systematically are investigated. MILP-models can cope well with the non-convexities, the combinatorial character and their efficiency hardly depends on initial

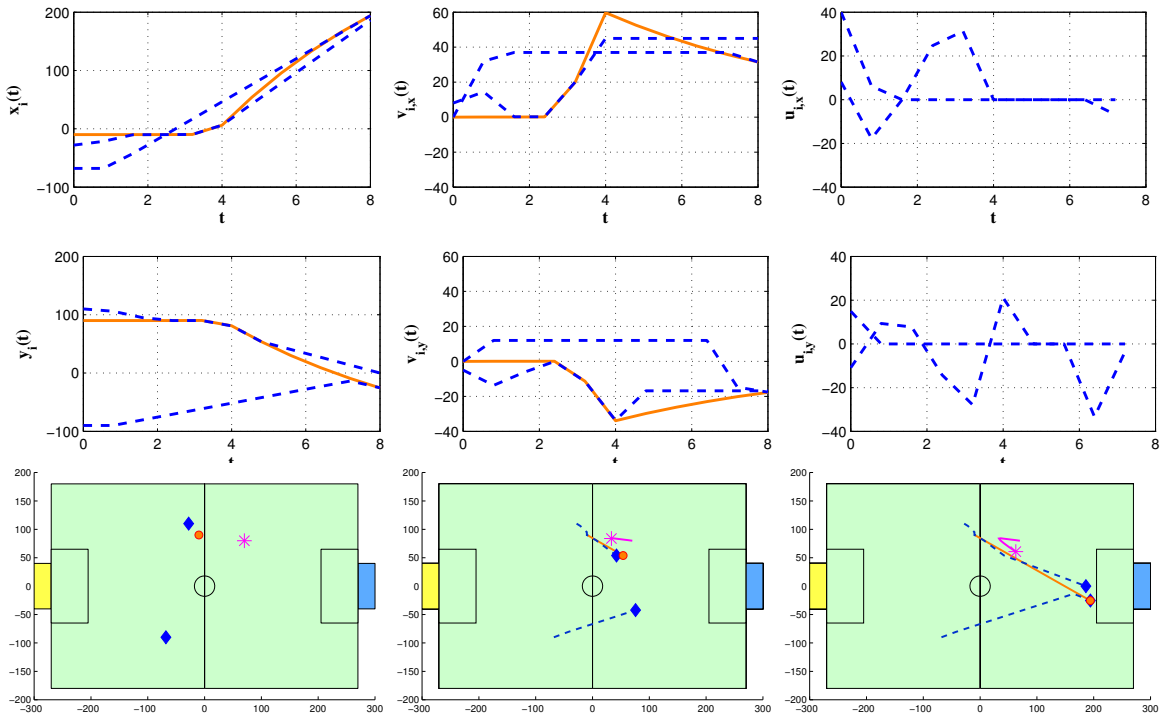


Figure 4: First two rows: Optimal positions, velocities and controls for the attackers (--♦) and the ball (—●). Third row: Computed optimal behavior shown at timesteps $k = 1, 7, 11$. Attacker 1 goes to ball, dribbles and kicks it towards the penalty area. Attacker 2 catches it there and dribbles to a promising position. The defender (—✳) follows the ball.

guesses. They are therefore well suited for repeated application to account for changes in uncertain environments.

Acknowledgement

Parts of this research have been supported by the German Research Foundation (DFG) within the Research Training Group 1362 “Cooperative, adaptive and responsive monitoring in mixed mode environments”.

References

[1] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[2] F. Borrelli, D. Subramanian, A. U. Raghunathan, and L. T. Biegler. MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles. *American Control Conference*, June 2006.

[3] M. G. Earl and R. D'Andrea. Iterative MILP methods for vehicle control problems. In *IEEE Conference on Decision and Control*, volume 4, pages 4369 – 4374, December 2004.

[4] M. Glocker, C. Reinl, and O. von Stryk. Optimal task allocation and dynamic trajectory planning for multi-vehicle systems using nonlinear hybrid optimal control. In *Proc. 1st IFAC-Symposium on Multivehicle Systems*, pages 38–43, Salvador, Brazil, 2006.

[5] T.A. Henzinger. The theory of hybrid automata. In M.K. Inan and R.P. Kurshan, editors, *Verification of Digital and Hybrid Systems*, NATO ASI Series F: Computer and Systems Sciences 170, pages 265–292. Springer, 2000.

Central pattern generator for legged locomotion: a mathematical approach

Carla M. A. Pinto*

Abstract

This paper is a survey of the work done by Pinto and Golubitsky [18] on a model of a central pattern generator (CPGs) for biped leg rhythms, leg, and its contribution to a new insight on the study of CPG models for humanoid robotic locomotion.

Leg is a network of four all-to-all coupled neurons (cells). Each leg (joint) receives signals from two cells. The symmetry of this network predicts periodic solutions associated with the rhythms of known biped gaits, such as *walk*, *run*, *two-legged hop*, *two-legged jump*, *skip*, *gallop*, *asymmetric hop*, and *one-legged hop*.

Similarly to the network leg, CPG models for locomotion in robots are commonly modeled by neural network oscillator circuits coupled to the joints of the robot. Each oscillator model consists of two simulated coupled neurons.

We find that the mathematical properties of the network leg may be considered new tools to the understanding and implementation of locomotion biped patterns in humanoid robots.

1 Introduction

Biologists often assume that vertebrate locomotion is controlled by a central pattern generator (CPG), located somewhere in the nervous system. A CPG, induced by command neurons, generates signals according to the movement patterns of specific gaits in animals [2, 7, 16].

Legged locomotion involves a large number of degrees of freedom, such as, pelvic rotation about a vertical axis, pelvic tilt, knee flexion, and many others till up to two hundred [3] and references therein. In this sense, a CPG plays an important role for having well-coordinated movements of these degrees of freedom.

Mathematically, CPGs are modeled by networks of identical systems of differential equations. A brief explanation to this fact is that CPGs consist of neurons, neurons are modeled

*Instituto Superior de Engenharia do Porto. Centro de Matemática da Universidade do Porto. Rua Dr António Bernardino de Almeida, 431, 4200-072 Porto E-mail:cpinto@fc.up.pt.

by electrical circuits (see for example [8]), and these circuits are described as systems of differential equations. Each individual system, which we call *cell*, model neurons or collections of neurons.

Golubitsky *et al.* [5] introduce two locomotor CPG models for legged movements: an eight-cell quadruped locomotor CPG model, quad, and the analogous four-cell CPG model, leg, for leg rhythms in bipeds, see Figure 1. Quad and leg are the minimal models capable of producing the usual quadruped gaits of walk, trot and pace, and the biped gaits of *walk* and *run*, respectively. A physiological interpretation can be given to the fact that these models have two cells per limb [5]. Most joints are driven by two muscle groups and the activity of these muscles must be coordinated. If we think of locomotor CPGs abstractly as controlling muscle groups rather than legs then it makes sense that minimal locomotor CPG networks should have two cells for each leg. Each graph, the *network architecture*, corresponds to a class of systems of differential equations [6]. Each of these systems can be considered to be a model CPG.

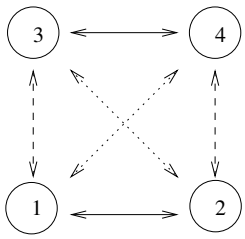


Figure 1: CPG network leg for the control of biped legs. Cells 1 and 3 send signals to the left leg, cells 2 and 4 send signals to the right leg. The arrows denote the coupling strength between cells. Different arrows denote different coupling strengths.

There has been a growing interest in the study of CPG models in robotics. They have been used to model a variety of robotic tasks, such as arm motion [20], swinging [11], bipedal walking [21, 15], quadruped walking [10], one-legged hopping [19], and other rhythmic patterns. CPGs for robotic locomotion are commonly modeled by neural network oscillator circuits coupled to the joints of the robot. Each oscillator is usually modeled by two simulated coupled neurons. The output of the oscillator unit is used as the angular speed of the joint. Examples of models used in the oscillators are the Matsuoka model [13], the van der Pol equations [3], the Wilson-Cowan equations [14].

Humanoid locomotor models have been increasinly studied. Bipedal robots are best suited for many tasks in real life, such as walking in real terrain [9], work in specialized environments, such as assembly lines, and other situations where robots need to work with or replace humans [1].

In this paper, we review the mathematical approach, based on symmetry techniques, used

to classify the periodic solutions, along with their associated limb rhythms, of the four-cell network model leg and we explain why we think leg brings a new insight in the study of CPG models for humanoid robotic locomotion.

2 CPG for leg rhythms

This section is divided into three parts. We start with a review of the H/K theorem. In subsection 2.2, we explain the relationship between two computed symmetry types and the rhythms of the biped *walk* and *run*. Finally, in subsection 2.3, we describe the importance of leg in the study of CPG locomotor models in humanoid robots.

2.1 leg: differential equations and symmetries

CPG networks stand for classes of systems of differential equations. For example, the class of differential equations corresponding to leg is

$$\begin{array}{rcl} \dot{x}_1 & = & F(x_1, x_2, x_3, x_4) \\ \dot{x}_2 & = & F(x_2, x_1, x_4, x_3) \\ \dot{x}_3 & = & F(x_3, x_4, x_1, x_2) \\ \dot{x}_4 & = & F(x_4, x_3, x_2, x_1) \end{array} \tag{1}$$

where $x_i \in \mathbf{R}^k$ and $F : (\mathbf{R}^k)^4 \to \mathbf{R}^k$ where all cells are assumed to be identical.

The network leg consists of four cells, each one coupled to all other cells. The coupling allows two independent permutation symmetries: $\rho = (1\ 2)(3\ 4)$, which switches muscle groups between legs, and $\tau = (1\ 3)(2\ 4)$, which permutes muscle groups within each leg. The group of symmetries of leg is the four element group

$$\Gamma_{\mathsf{leg}} = \mathbf{Z}_2(\tau) \times \mathbf{Z}_2(\rho) \tag{2}$$

Let K be the subgroup of Γ_{leg} consisting of symmetries that fix a periodic solution pointwise (spatial symmetries), and $H \subset \Gamma_{\mathsf{leg}}$ be the subgroup consisting of symmetries that preserve the periodic trajectory setwise (spatiotemporal symmetries).

The H/K Theorem [4] establishes the algebraic conditions that a given symmetry pair (H, K) must satisfy to correspond to symmetries of a periodic solution, in a general Γ-equivariant system. In the case of the coupled system (1), these conditions are simplified to: H/K is a cyclic group [18]. The periodic solutions with (H, K) symmetry pairs are then identified with leg rhythms in bipeds.

The pairs of subgroups $K \subset H \subset \Gamma_{\mathsf{leg}}$, such that H/K is cyclic and their correspondence with known biped gait patterns is given in Table 1. Periodic solutions with $H = \Gamma_{\mathsf{leg}}$ are called *primary gaits*. Gaits that are not primary gaits are called *secondary gaits*. All the secondary gaits are obtained by symmetry-breaking of primary gaits [17]. Each bifurcation may occur if the cell dynamics and the coupling architecture are general enough.

Name	H	K
two-legged hop	Γ_{leg}	Γ_{leg}
walk	Γ_{leg}	$\mathbf{Z}_2(\rho\tau)$
two-legged jump	Γ_{leg}	$\mathbf{Z}_2(\rho)$
run	Γ_{leg}	$\mathbf{Z}_2(\tau)$
asymmetric hop	$\mathbf{Z}_2(\rho\tau)$	$\mathbf{Z}_2(\rho\tau)$
	$\mathbf{Z}_2(\rho\tau)$	$\mathbf{1}$
	$\mathbf{Z}_2(\rho)$	$\mathbf{Z}_2(\rho)$
skip	$\mathbf{Z}_2(\rho)$	$\mathbf{1}$
one-legged hop	$\mathbf{Z}_2(\tau)$	$\mathbf{Z}_2(\tau)$
gallop	$\mathbf{Z}_2(\tau)$	$\mathbf{1}$
	$\mathbf{1}$	$\mathbf{1}$

Table 1: Symmetry pairs (H, K) of standard biped leg rhythms. See text for more information.

2.2 Bipedal leg rhythms associated to leg

Walk and *run* have the property that left and and right legs are half-period phase shifted. In the *run*, the flexor and extensor muscles of the ankle joint are in phase whereas in the *walk* they are out of phase [12]. Biomechanically, the legs move like pendula (ankle joint rotates) in the *walk* and like a pogo stick (ankle joint is held rigid) in the *run*.

We use the (H, K) symmetry pairs (see Table 1) of these gaits to distinguish between them. Let $X(t) = (x_1(t), x_2(t), x_3(t), x_4(t))$ be a periodic solution produced by the network leg, with period normalized to 1. Observe that in the *walk*, $\rho\tau$ is a K symmetry, hence the periodic solution must have the form $X(t) = (x_1(t), x_2(t), x_2(t), x_1(t))$. The spatiotemporal symmetry τ implies $x_2(t) = x_1(t + \frac{1}{2})$. Thus, the periodic solution has the form

$$X(t) = (x_1(t), x_1(t + \frac{1}{2}), x_1(t + \frac{1}{2}), x_1(t))$$

Analogously for the *run*. The spatial symmetry τ and the spatiotemporal symmetry ρ force the periodic solution to have the form

$$X(t) = (x_1(t), x_1(t + \frac{1}{2}), x_1(t), x_1(t + \frac{1}{2}))$$

2.3 Leg and humanoid robot CPG locomotor models

The bipedal rhythms produced by the minimal CPG model leg are predicted solely by its symmetry. As discussed above, the H/K Theorem imposes only the restriction that H/K is cyclic in order for the symmetry pair (H, K) to correspond to a periodic solution of the system (1). This implies there is no restraint in the form of the function F, i.e., neurons can be

modeled by any function F, such that $k \geq 2$. The later also allows for the identical couplings to be of various types, namely, synaptic or diffusive, excitatory or inhibitory. Moreover, all of the secondary gaits are obtained by breaking the symmetry of the primary gaits. These properties of the model leg may give a new insight in the study and real implementation of new gaits in humanoid robotic CPG models.

3 Conclusion

Symmetry techniques show that the four-cell model leg produces the biped rhythms of *walk*, *run*, *two-legged hop*, *two-legged jump*, *skip*, *gallop*, *asymmetric hop*, and *one-legged hop*. We distinguish between primary and secondary gaits. Moreover, all of the secondary gaits are obtained by symmetry-breaking of primary gaits.

We find that leg may help in the study of new CPG models for humanoid robotics and in the real implementation of new gaits, such as the *two-legged hop*, *gallop*, or *one-legged hop*.

References

[1] B. Adams, C. Breazeal, R. A. Brooks, and B. Scassellati. Humanoid robots: A new kind of tool. *IEEE Intelligent Systems*, pages 25–31, 2000.

[2] A. H. Cohen, G. B. Ermentrout, T. Kiemel, N. Kopell, K. A. Sigvardt, and T. L. Williams. Modelling of intersegmental coordination in the lamprey central pattern generator for locomotion. *Trends in Neuroscience*, 15:434–438, 1992.

[3] M. S. Dutra, A. C. de Pina Filho, and V. F. Romano. Modeling of a bipedal locomotor using coupled nonlinear oscillators of van der pol. *Biological Cybernetics*, 88:286–292, 2003.

[4] M. Golubitsky and I. Stewart. The symmetry perspective. *Progress in Mathematics* **200**, *Birkhauser, Basel*, 2002.

[5] M. Golubitsky, I. Stewart, P. L. Buono, and J.J. Collins. Symmetry in locomotor central pattern generators and animal gaits. *Nature*, 401:693–695, 1999.

[6] M. Golubitsky, I. Stewart, and A. Török. Patterns of synchrony in coupled cell networks with multiple arrows. *SIAM J. Appl. Dynam. Sys.*, 4:78–100, 2005.

[7] S. Grillner, J. T. Buchanan, P. Walker, and L. Brodin. Neural control of locomotion in lower vertebrates. *Neural Control of Rhythmic Movements in Vertebrates, John Wiley & Sons*, pages 1–40, 1988.

[8] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[9] Q. Huang, K. Li, and Y. Nakamura. Humanoid walk control with feedforward dynamic pattern and feedback sensory reflection. *Paper presented at the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Seoul, Korea*, 2001.

[10] H. Kimura, Y. Fukuoka, and K. Konaga. Adaptive dynamic walking of a quadruped robot by using neural system model. *ADVANCED ROBOTICS*, 15:859–876, 2001.

[11] M. Lungarella and L. Berthouze. On the interplay between morphological, neural, and environmental dynamics: A robotic case study. *Adaptive Behavior*, 10:223–241, 2002.

[12] R. A. Mann. Biomechanics. *Disorders of the Foot (Jahss, M.H. ed), W.B. Saunders and Co., Philadelphia*, pages 37–67, 1982.

[13] K. Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 52:367–376, 1985.

[14] K. Nakada, T. Asai, and Y. Amemiya. Design of an artificial central pattern generator with feedback controller. *Intelligent Automation and Soft Computing*, 10:185–192, 2004.

[15] G. C. Nandi and B. Gupta. Bio-inspired control methodology of walking for intelligent prosthetic knee. *Proceedings of the 3rd International Conference of Informatics in Control, Automation and Robotics, ICINCO 2006*, 2006.

[16] K. G. Pearson. Common principles of motor control in vertebrates and invertebrates. *Annual Review of Neuroscience*, 16:265–297, 1993.

[17] C. M. A. Pinto. Numerical simulations in two cpg models for bipedal locomotion. *Journal of Vibration and Control*, 13:1487–1503, 2007.

[18] C. M. A. Pinto and M. Golubitsky. Central pattern generators for bipedal locomotion. *Journal of Mathematical Biology*, 53:474–489, 2006.

[19] M. H. Raibert, H. B. Brown Jr., and M. Chepponis. Experiments in balance with a 3d one-legged hopping machine. *The International Journal of Robotics Research*, 3:75–92, 1984.

[20] M. M. Williamson. Neural control of rhythmic arm movements. *Neural Networks*, 11:1379–1394, 1998.

[21] T. Zielinska. Coupled oscillators utilised as gait rhythm generators of a two-legged walking machine. *Biological Cybernetics*, 74:263–273, 1996.

Lifted fundamental matrices for mixtures of central projection systems

João P. Barreto *

Abstract

We study the epipolar geometry between views acquired by mixtures of central projection systems including catadioptric sensors and cameras with lens distortion. Since the projection models are in general non-linear, a new representation for the geometry of central images is proposed. This representation is the lifting through Veronese maps of the image plane to the 5D projective space. It is shown that, for most sensor combinations, there is a bilinear form relating the lifted coordinates of corresponding image points. We analyze the properties of the embedding and explicitly construct the lifted fundamental matrices in order to understand their structure. The usefulness of the framework is illustrated by estimating the epipolar geometry between images acquired by a paracatadioptric system and a camera with radial distortion.

1 Introduction

A central projection camera is an image acquisition device with a single effective viewpoint. The vision sensor measures the intensity of light traveling along rays that intersect in a single point in 3D (the projection center). Examples of broadly used central projection systems are perspective cameras, central catadioptric systems [1], and cameras with lens distortion. The projection in conventional perspective cameras is described by the pin-hole model where scene points are linearly mapped into image points. Corresponding points in two perspective views must satisfy a bilinear constraint that is usually represented by a 3×3 matrix. The fundamental matrix encodes the calibration and rigid displacement between views, and can be estimated in closed-form using image correspondences. These facts make the fundamental geometry one of the most popular and useful concepts in computer vision. Unfortunately, the projection model for central catadioptric systems and cameras with radial distortion is non-linear in homogeneous coordinates. The epipolar geometry between views acquired by mixtures of these systems does no longer have a bilinear form, which considerably limits their usefulness.

*ISR/DEEC, University of Coimbra. 3030 Coimbra, Portugal. E-mail: jpbar@deec.uc.pt.

The first papers generalizing the epipolar geometry to the non pin-hole case either assumed pre-calibrated systems [13], or completely relied on non-linear iterative minimization [15]. Closely related with the approach herein presented are the works of Geyer et al. [7], Sturm [12] and Claus et al. [5]. Geyer et al. propose an image plane lifting to a four dimensional 'circle space' and prove that there is a 4×4 fundamental matrix encoding the epipolar geometry between two paracatadioptric views. Sturm uses a slightly different lifting strategy and derives the fundamental matrices for mixtures of perspective, affine and paracatadioptric cameras. In [5], Claus et al. propose the lifting of image points to a six dimensional space to build a general purpose model for radial distortion in wide angle and catadioptric lenses. The epipolar geometry between distorted views is represented by a 6×6 matrix. As stated by the authors, their non-parametric model is algebraic in inspiration rather than geometric. Therefore, they do not provide any insight or geometric interpretation about the structure of the lifted fundamental matrix.

In this paper we introduce a representation for the image plane of central systems including catadioptric sensors and cameras with distortion. The representation is similar to the one proposed in [5], and consists in the lifting through Veronese maps of the projective plane $\wp^{\mathbf{2}}$ to the 5D projective space $\wp^{\mathbf{5}}$ [11]. Our goal is to establish a unifying framework for the geometry of general single viewpoint images. A full theory to lift points, lines, conics and conic envelopes is presented. It is also shown how to transfer a linear transformation from $\wp^{\mathbf{2}}$ to $\wp^{\mathbf{5}}$ as well as other geometric relations. We prove that, for most combinations of central projection views, there is a bilinear form relating the lifted coordinates of corresponding points. Our embedding theory is used to understand this lifted epipolar geometry and explicitly construct the different fundamental matrices. The main contributions can be summarized as follows:

- We establish a lifted representation of the image plane and develop a full embedding theory to transfer geometric entities and relations from $\wp^{\mathbf{2}}$ to $\wp^{\mathbf{5}}$.
- We present the lifted fundamental matrices for different mixtures of central projection systems. It is proved for the first time that there is a lifted bilinear constraint between images acquired by a perspective and hyperbolic camera as well as a parabolic sensor and camera with lens distortion.
- The different fundamental matrices are presented in a systematic manner and their structure is discussed. We also provide a comprehensive geometric explanation for the non existence of bilinear forms for hyperbolic views other than the combination with a perspective.

2 Epipolar geometry using Veronese maps

Conventional perspective cameras, central catadioptric systems, and cameras with radial distortion are vision systems with a single effective viewpoint. All these sensors measure the intensity of light traveling along rays that intersect in a single point in 3D (the projection center). Consider a coordinate system attached to the camera such that the $\mathbf{R}$ (rotation) and $\mathbf{t}$ (translation) describe its rigid displacement with respect to the world reference frame. Any visible 3D point $\mathbf{X}$ is mapped into a projective ray $\mathbf{x} = \mathbf{P}\mathbf{X}$ with $\mathbf{P} = \mathbf{R}[\mathbf{I}| - \mathbf{t}]$. The 3×4 matrix $\mathbf{P}$ is the conventional projection matrix, and we will say that $\mathbf{x} = (x, y, z)^T$ is a projective point in the *Canonical Perspective Plane* (CPP).

Consider the case of central catadioptric systems, where $\mathbf{x}$ is mapped into the image point $\mathbf{x}'$. The relation between these two projective points is provided in equation 1 where $\hbar$ is a non-linear function (equation 2) and $\mathbf{H_c}$ is a collineation depending on the camera intrinsics, the relative rotation between camera and mirror, and the shape of the reflective surface. Function $\hbar$ is equivalent to a projective mapping from a sphere to a plane as shown in Fig. 1 [6]. Parameter ξ in equation 2 depends on the mirror shape and takes values in the interval $]0, 1]$.

$$\mathbf{x}' = \mathbf{H_c}\hbar(\mathbf{x}) \tag{1}$$

$$\hbar(\mathbf{x}) = (x, y, z + \xi\sqrt{x^2 + y^2 + z^2})^T \tag{2}$$

Equation 3 shows the correspondence between projective rays $\mathbf{x}$ and image points $\mathbf{x}'$ for the case of perspective cameras with lens distortion. Matrix $\mathbf{K_c}$ denotes the camera intrinsic parameters and $\eth$ is a non-linear function modeling the radial distortion. In this work the lens distortion is modeled using the division model introduced in [4]. For now we will assume that the coordinates system in the image plane has origin in the distortion center that is known. The inverse function of $\eth$ is provided in equation 4 where parameter ξ quantifies the amount of radial distortion. Transformation $\eth$ has a geometric interpretation similar to the popular sphere model used for catadioptric systems. It can be proved that function $\eth$ is isomorphic to a projective mapping from a paraboloid to a plane (Fig. 1) [2].

$$\mathbf{x}' = \eth(\mathbf{K_c}\mathbf{x}) \tag{3}$$

$$\eth^{-1}(\mathbf{x}') = (x'z', y'z', z'^2 + \xi(x'^2 + y'^2))^T \tag{4}$$

The type of central projection system is defined by the value of parameter ξ. In the case of barrel distortion the ξ is negative and $\eth$ is used. If ξ is positive then the system is a catadioptric sensor. The parameter is unitary in the parabolic case and $\xi \in]0, 1[$ for hyperbolic/elliptical mirrors. For $\xi = 0$ both equations 1 and 2 become linear and represent the well known pin-hole model for perspective cameras.

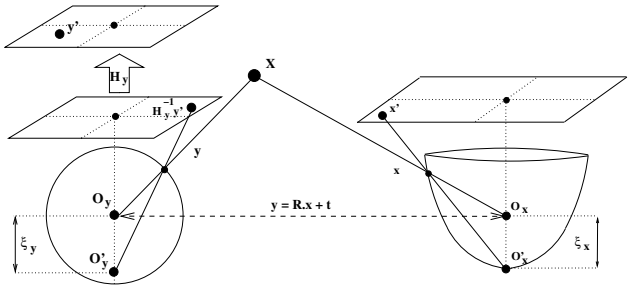


Figure 1: Geometric relation between views acquired by a catadioptric system and a dioptric camera with radial distortion. The 3D point $\mathbf{X}$ is imaged in point $\mathbf{x}'$ in the dioptric camera, and in point $\mathbf{y}'$ in the catadioptric image plane.

2.1 Epipolar Geometry

This work aims to study the multi-view relations that hold between images obtained with any mixture of central projection systems. Fig. 1 shows two views of the same 3D point $\mathbf{X}$ acquired by a camera with lens distortion and a central catadioptric sensor. If $\mathbf{x} \leftrightarrow \mathbf{y}$ are corresponding projective rays then they must satisfy $\mathbf{y}^T\mathbf{E}\mathbf{x} = 0$ where $\mathbf{E} = \hat{\mathbf{t}}\mathbf{R}$ is the essential matrix. Since both $\hbar$ and $\eth$ are invertible functions it follows from equations 1 and 2 that

$$(\underbrace{\hbar^{-1}(\mathbf{H_y^{-1}}\mathbf{y}')}_{\mathbf{y}})^T\mathbf{E}\underbrace{\mathbf{K_x^{-1}}\eth^{-1}(\mathbf{x}')}_{\mathbf{x}} = 0. \tag{5}$$

It is broadly known that corresponding points in two perspectives satisfy a bilinear constraint called the fundamental equation. The fundamental relation is linear in homogeneous coordinates and can be represented by a 3×3 matrix $\mathbf{F}$. Equation 5 is the equivalent of the fundamental relation for two views acquired by a catadioptric sensor and a camera with distortion. Due to the non-linear image mapping the relation is no longer linear which limits its usefulness when compared with the conventional fundamental matrix.

2.2 Lifting of coordinates using Veronese maps

A standard technique used in algebra to render a nonlinear problem into a linear one is to find an embedding that lifts the problem into a higher dimensional space. In a certain extent the homogeneous representation is an embedding of $\mathfrak{R}^2$ into $\mathfrak{R}^3$. Unfortunately the use of an additional coordinate does no longer suffice to cope with the non-linearity of equations 1 and 3. The present work aims to overcome this problem and derive a bilinear fundamental relation that holds for general central projection systems. We propose to embed the projective plane $\wp^2$ in the five dimensional projective space $\wp^5$ using second order Veronese mapping [11]. This polynomial embedding preserves homogeneity and is suitable to deal with quadratic

	P-H	Hyper.	Parab.	Dist.
P-H	9	1	3	3
Hyper.	1	0	0	0
Parab.	3	0	1	1
Dist.	3	0	1	1

Table 1: Dimension of the null space of matrix $\mathbf{\Psi}$ for pairs of views acquired by different combinations of sensors (*P-H* = pin-hole; *Hyper* = Hyperbolic; *Parab* = Parabolic; *Dist* = Radial Distortion).

functions because it discriminates the entire set of second order monomials. The lifting of coordinates can be performed by applying the following operator

$$\mathbf{\Gamma}(\mathbf{x}, \bar{\mathbf{x}}) = (x\bar{x}, \frac{x\bar{y} + y\bar{x}}{2}, y\bar{y}, \frac{x\bar{x} + z\bar{x}}{2}, \frac{y\bar{z} + z\bar{y}}{2}, z\bar{z})^T \tag{6}$$

Operator $\mathbf{\Gamma}$ transforms two 3×1 vectors into a 6×1 vector. Equation 6 maps the pair of projective points $\mathbf{x}$, $\bar{\mathbf{x}}$ into one, and only one, point in $\wp^{\mathbf{5}}$. This point lies on a primal of the 5D projective space called the cubic symmetroid [11]. As shown in equation 7, $\mathbf{\Gamma}$ can also be used to map a single point $\mathbf{x}$ into a point $\tilde{\mathbf{x}}$ in the lifted space, lying on a quadratic surface known as the Veronese surface [11]. The Veronese surface $\mathbf{V}$ is a subset of the cubic symmetroid $\mathbf{S}$.

$$\mathbf{x} \longrightarrow \widetilde{\mathbf{x}} = \mathbf{\Gamma}(\mathbf{x}, \mathbf{x}) = (x^2, xy, y^2, xz, yz, z^2)^T. \tag{7}$$

2.3 The lifted fundamental matrix $\digamma$

Fig. 1 shows a pair of corresponding image points $\mathbf{x}' \leftrightarrow \mathbf{y}'$. The lifted coordinates of the two points are $\widetilde{\boldsymbol{x}}'$ and $\widetilde{\boldsymbol{y}}'$ (equation 7). The idea of embedding the projective plane in $\wp^{\mathbf{5}}$ is to obtain a bilinear relation between the two views. The goal is achieved if there is a 6×6 homogeneous matrix $\digamma$ such that

$$\widetilde{\mathbf{y}}'^{\mathbf{T}} \digamma \widetilde{\mathbf{x}}' = 0. \tag{8}$$

Consider the set of lifted correspondences $\widetilde{\mathbf{x}}'_{\mathbf{i}} \leftrightarrow \widetilde{\mathbf{y}}'_{\mathbf{i}}$ with $i = 1, 2 \ldots N$ and $N > 35$. Matrix $\mathbf{\Psi}$ is obtained by stacking the N lines corresponding to the Kronecker products $\widetilde{\mathbf{x}}'^{\mathbf{T}}_{\mathbf{i}} \otimes \widetilde{\mathbf{y}}'^{\mathbf{T}}_{\mathbf{i}}$. The bilinear relation of equation 8 holds *iff* matrix $\mathbf{\Psi}$ is rank deficient. If $\mathbf{\Psi}$ has a left null space then the non-trivial solutions of equation 9 are solutions for the lifted fundamental matrix $\digamma$.

$$\underbrace{\begin{bmatrix} \widetilde{\mathbf{x}}_1'^{\mathbf{T}} \otimes \widetilde{\mathbf{y}}_1'^{\mathbf{T}} \\ \vdots \\ \widetilde{\mathbf{x}}_N'^{\mathbf{T}} \otimes \widetilde{\mathbf{y}}_N'^{\mathbf{T}} \end{bmatrix}}_{\mathbf{\Psi}} \begin{bmatrix} f_{11} \\ \vdots \\ f_{66} \end{bmatrix} = 0 \tag{9}$$

We planned a synthetic experiment to determine the combinations of central projection sensors for which equation 8 holds. For each combination the two vision sensors are placed in a virtual volume and a set of N points is generated assuming a random distribution ($N \gg 35$). Noise free correspondences are obtained by projecting the 3D points in both views. The lifted coordinates of the matching points are used to build matrix $\mathbf{\Psi}$. Tab. 1 summarizes for each case the dimension of the left null space of $\mathbf{\Psi}$. For most of the sensor combinations there is a bilinear relation between views. The only exception is whenever there is an hyperbolic sensor involved. In this situation equation 8 holds only if the other view is acquired by a pin-hole. If one of the views is a conventional perspective then there are multiple solutions for the lifted fundamental matrix.

3 The embedding in $\wp^5$

In order to interpret the results of Tab. 1 and gain insight about the structure of the lifted fundamental matrix, we need to understand the way that geometric entities and relations in the projective plane are embedded in $\wp^5$.

3.1 Lifting lines and conics

A conic curve in $\wp^2$ is usually represented by a 3×3 symmetric matrix $\mathbf{\Omega}$. Point $\mathbf{x}$ lies on $\mathbf{\Omega}$ *iff* equation $\mathbf{x}^T\mathbf{\Omega}\mathbf{x} = 0$ is verified. Since a 3×3 symmetric matrix has 6 parameters, the conic locus can also be represented by a 6×1 homogeneous vector $\widetilde{\boldsymbol{\omega}}$, that is the lifted representation of $\mathbf{\Omega}$ in $\wp^5$ (equation 10).

$$\mathbf{\Omega} = \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & f \end{bmatrix} \longrightarrow \widetilde{\boldsymbol{\omega}} = (a, 2b, c, 2d, 2e, f)^T. \tag{10}$$

Consider the rank 2 conic $\mathbf{\Omega} = \mathbf{m}\mathbf{l}^T + \mathbf{l}\mathbf{m}^T$ composed by two lines $\mathbf{m}$ and $\mathbf{l}$. From equation 10 follows that the corresponding lifted representation is

$$\mathbf{\Omega} = \mathbf{m}\mathbf{l}^T + \mathbf{l}\mathbf{m}^T \longrightarrow \widetilde{\boldsymbol{\omega}} = \widetilde{\mathbf{D}}\mathbf{\Gamma}(\mathbf{m}, \mathbf{l}) \tag{11}$$

where $\widetilde{\mathbf{D}} = diag\{1, 2, 1, 2, 2, 1\}$. A single line $\mathbf{n} = (n_x, n_y, n_z)^T$ is another example of a degenerate conic curve $\mathbf{\Omega} = \mathbf{n}\mathbf{n}^T$. Line $\mathbf{n}$ in the projective plane is mapped into $\widetilde{\mathbf{n}}$ in $\wp^5$ as shown in equation 12.

$$\mathbf{n} \rightarrow \widetilde{\mathbf{n}} = \widetilde{\mathbf{D}}\mathbf{\Gamma}(\mathbf{n}, \mathbf{n}) = (n_x^2, 2n_xn_y, n_y^2, 2n_xn_z, 2n_yn_z, n_z^2)^T \tag{12}$$

Conic $\mathbf{\Omega}$ goes through point $\mathbf{x}$ *iff* the inner product of the corresponding lifted representations is zero ($\mathbf{x}^T\mathbf{\Omega x} = 0 \rightarrow \widetilde{\boldsymbol{\omega}}^T\widetilde{\mathbf{x}} = 0$). Additionally, if points $\mathbf{x}$ and $\bar{\mathbf{x}}$ are harmonic conjugates with respect to $\mathbf{\Omega}$, then $\widetilde{\boldsymbol{\omega}}^{\boldsymbol{T}}\mathbf{\Gamma}(\mathbf{x}, \bar{\mathbf{x}}) = 0$. The embedding of the projective plane in $\wp^{\mathbf{5}}$ using Veronese mapping creates a dual relation between points and conics. Moreover, and since lines are degenerate conics of rank 1, the duality between points and lines is preserved.

3.2 Lifting conic envelopes

In general a point conic $\mathbf{\Omega}$ has a dual conic envelope $\mathbf{\Omega^*}$ associated with it [11]. The envelope is usually represented by a 3×3 symmetric matrix. A certain line $\mathbf{n}$ is on the conic envelope whenever it satisfies $\mathbf{n}^T\mathbf{\Omega^* n} = 0$. The conic envelope can also be represented by a 6×1 homogeneous vector $\widetilde{\boldsymbol{\omega}}^*$ (equation 13). In this case a line $\mathbf{n}$ lies on $\mathbf{\Omega^*}$ *iff* the corresponding lifted vectors $\widetilde{\mathbf{n}}$ and $\widetilde{\boldsymbol{\omega}}^*$ are orthogonal.

$$\mathbf{\Omega^*} = \begin{bmatrix} a^* & b^* & d^* \\ b^* & c^* & e^* \\ d^* & e^* & f^* \end{bmatrix} \longrightarrow \widetilde{\boldsymbol{\omega}}^* = (a^*, b^*, c^*, d^*, e^*, f^*)^T. \tag{13}$$

If matrix $\mathbf{\Omega^*}$ is rank deficient then the conic envelope is said to be degenerate. There are two possible cases of degeneracy: when the $\mathbf{\Omega^*}$ is composed by two pencils of lines going through points $\mathbf{x}$ and $\bar{\mathbf{x}}$ ($\mathbf{\Omega^*} = \mathbf{x}\bar{\mathbf{x}}^T + \bar{\mathbf{x}}\mathbf{x}^T$), and when the envelope is formed by a single pencil of lines ($\mathbf{\Omega^*} = \mathbf{x}\mathbf{x}^T$). The lifted representations are respectively provided in equations 14 and 15.

$$\mathbf{\Omega^*} = \mathbf{x}\bar{\mathbf{x}}^T + \bar{\mathbf{x}}\mathbf{x}^T \longrightarrow \widetilde{\boldsymbol{\omega}}^* = \mathbf{\Gamma}(\mathbf{x}, \bar{\mathbf{x}}) \tag{14}$$

$$\mathbf{\Omega^*} = \mathbf{x}\mathbf{x}^T \longrightarrow \widetilde{\boldsymbol{\omega}}^* = \mathbf{\Gamma}(\mathbf{x}, \mathbf{x}) \tag{15}$$

3.3 Lifting linear transformations

The linear transformation $\mathbf{H}$ maps points $\mathbf{x}$ and $\bar{\mathbf{x}}$ in points $\mathbf{Hx}$ and $\mathbf{H}\bar{\mathbf{x}}$. The operator $\mathbf{\Lambda}$, that lifts transformation $\mathbf{H}$ from the projective plane to the embedding space $\wp^{\mathbf{5}}$, must satisfy the following relation

$$\mathbf{\Gamma}(\mathbf{Hx}, \mathbf{H}\bar{\mathbf{x}}) = \mathbf{\Lambda(H)}.\mathbf{\Gamma}(\mathbf{x}, \bar{\mathbf{x}}) \tag{16}$$

Such operator can be derived by algebraic manipulation.

$$\mathbf{\Lambda}(\underbrace{[\,\mathbf{v_1}\ \mathbf{v_2}\ \mathbf{v_3}\,]}_{\mathbf{H}}) = \underbrace{[\,\mathbf{\Gamma_{11}\Gamma_{12}\Gamma_{22}\Gamma_{13}\Gamma_{23}\Gamma_{33}}\,]\,\widetilde{\mathbf{D}}}_{\widetilde{\mathbf{H}}} \tag{17}$$

It can be proved that the operator provided above verifies the following properties

$$\begin{array}{rcl} \mathbf{\Lambda}(\mathbf{H}^{-1}) & = & \mathbf{\Lambda}(\mathbf{H})^{-1} \\ \mathbf{\Lambda}(\mathbf{H}.\mathbf{B}) & = & \mathbf{\Lambda}(\mathbf{H}).\mathbf{\Lambda}(\mathbf{B}) \\ \mathbf{\Lambda}(\mathbf{H}^{T}) & = & \widetilde{\mathbf{D}}^{-1}.\mathbf{\Lambda}(\mathbf{H})^{T}.\widetilde{\mathbf{D}} \\ \mathbf{\Lambda}(\mathbf{I}_{3\times 3}) & = & \mathbf{I}_{6\times 6} \end{array} \tag{18}$$

Operator $\mathbf{\Lambda}$ maps any 3×3 matrix $\mathbf{H}$ into a 6×6 matrix $\widetilde{\mathbf{H}}$. A pair of points in the plane is related by $\mathbf{H}$, *iff* the pair of corresponding lifted representations in $\wp^{\mathbf{5}}$ is related by $\widetilde{\mathbf{H}}$ ($\mathbf{y} = \mathbf{H}\mathbf{x} \leftrightarrow \widetilde{\mathbf{y}} = \widetilde{\mathbf{H}}\widetilde{\mathbf{x}}$). The set of transformations $\widetilde{\mathbf{H}} = \mathbf{\Lambda}(\mathbf{H})$ is the subset of linear transformations of $\wp^{\mathbf{5}}$ that fixes both the cubic symmetroid $\mathbf{S}$ and the Veronese surface $\mathbf{V}$. However, neither $\mathbf{S}$ nor $\mathbf{V}$ are fixed point-wise. The transformation of points, conics and conic envelopes are lifted in the following manner

$$\begin{array}{lcl} \mathbf{y} = \mathbf{H}\mathbf{x} & \longrightarrow & \widetilde{\mathbf{y}} = \widetilde{\mathbf{H}}\widetilde{\mathbf{x}} \\ \mathbf{\Psi} = \mathbf{H}^{-T}\mathbf{\Omega}\mathbf{H}^{-1} & \longrightarrow & \widetilde{\psi} = \widetilde{\mathbf{H}}^{-T}\widetilde{\omega} \\ \mathbf{\Psi}^{*} = \mathbf{H}\mathbf{\Omega}^{*}\mathbf{H}^{T} & \longrightarrow & \widetilde{\psi}^{*} = \widetilde{\mathbf{H}}\widetilde{\omega}^{*} \end{array} \tag{19}$$

The operator $\mathbf{\Lambda}$ can be also be applied to lift correlations in $\wp^{\mathbf{2}}$ [11]. A correlation $\mathbf{G}$ maps points $\mathbf{x}$ into lines $\mathbf{n} = \mathbf{G}\mathbf{x}$. It can been easily proved that the corresponding lifted coordinates are related by $\widetilde{\mathbf{n}} = \widetilde{\mathbf{D}}\widetilde{\mathbf{G}}\widetilde{\mathbf{x}}$.

4 Useful relations in $\wp^{\mathbf{5}}$

The previous section shows how a linear transformation in the original space $\wp^{\mathbf{2}}$ can be transferred to a linear transformation in $\wp^{\mathbf{5}}$. This section focuses on the non-linear features of the mapping functions of equations 1 and 3. Therefore, and without loss of generality, collineations $\mathbf{H_c}$ and $\mathbf{K_c}$ will be ignored for clarity reasons.

The catadioptric projection of a line is in general a conic curve [13, 6]. This is due to the non-linear characteristics of function $\hbar$ (equation 2) that transforms points $\mathbf{x}$ in the *Canonical Perspective Plane* (CPP) into image points $\mathbf{x}'$. We show that there is a linear correspondence between the lifted representation of a line $\mathbf{n}$ and the conic curve where it is projected. This result allows us to derive a linear relation between an image point $\mathbf{x}'$ and its back-projections in the CPP. The discussion of the this section is of key importance to interpret the results of Tab. 1.

4.1 Projection of lines

Consider the central catadioptric sensor depicted in Fig. 1 and a plane $\mathbf{\Pi}$ defined by a 3D line and the system effective viewpoint $\mathbf{O}$. The normal vector of $\mathbf{\Pi}$ is $\mathbf{n} = (n_x, n_y, n_z)^T$ that can also be interpreted has a line projection in the CPP. Plane $\mathbf{\Pi}$ intersects the reference sphere in a great circle that is projected from $\mathbf{O}'$ into a conic curve $\mathbf{\Omega}'$. The non-linear formula relating

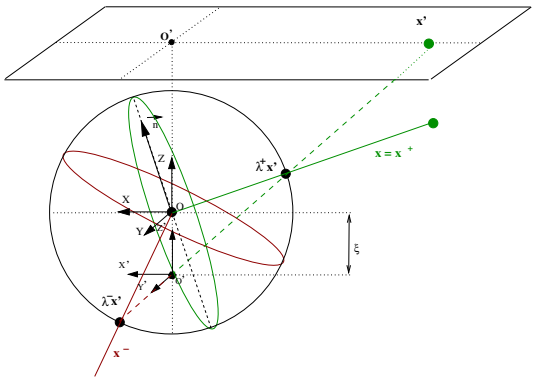


Figure 2: Back-projection of points in an hyperbolic sensor. The projective ray associated with $\mathbf{x}'$ intersects the unitary sphere in two points. These points define two distinct back-projections $\mathbf{x}^+$ (forward looking direction) and $\mathbf{x}^-$ (backward looking direction)

a line $\mathbf{n}$ in the CPP and the image conic $\boldsymbol{\Omega'}$ is provided in [6]. If $\widetilde{\mathbf{n}}$ and $\widetilde{\boldsymbol{\omega}}'$ are the lifted coordinates of $\mathbf{n}$ and $\boldsymbol{\Omega'}$ then it is straightforward to prove that equation 20 holds. The 6×6 matrix $\widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}$ of equation 21 transforms a line in the CPP into the corresponding conic curve in the catadioptric image plane. Remark that the structure of $\widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}$ does not follow equation 17, which means that there is no linear counterpart in $\wp^{\mathbf{2}}$.

$$\widetilde{\boldsymbol{\omega}}' = \widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}} \widetilde{\mathbf{n}}. \tag{20}$$

$$\widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}} = \begin{bmatrix} 1-\xi^2 & 0 & 0 & 0 & 0 & -\xi^2 \\ 0 & 1-\xi^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-\xi^2 & 0 & 0 & -\xi^2 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}; \widetilde{\boldsymbol{\Delta}}_{\boldsymbol{r}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \xi \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \xi \\ 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

The projection of a line by a camera with lens distortion is also a conic curve. Function $\eth$ of equation 4 is equivalent to projecting the scene on the surface of an unitary paraboloid and re-projecting from its vertex. As shown in Fig. 1 a projective ray $\mathbf{x}$ going through the camera projection center $\mathbf{O}$ is mapped in $\mathbf{x}'$ going through $\mathbf{O}'$. In this case plane $\boldsymbol{\Pi}$, containing the original 3D line, cuts the reference paraboloid in a great circle that is projected into a conic $\boldsymbol{\Omega'}$. The lifted representations of the line $\mathbf{n}$ in the CPP and the image conic are linearly related in $\wp^{\mathbf{5}}$ by $\widetilde{\boldsymbol{\omega}}' = \widetilde{\boldsymbol{\Delta}}_{\boldsymbol{r}} \widetilde{\mathbf{n}}$. The 6×6 matrix $\widetilde{\boldsymbol{\Delta}}_{\boldsymbol{r}}$ is provided in equation 21 where ξ quantifies the amount of radial distortion.

4.2 Back-projection of image points in central catadioptric systems

Fig. 2 shows the sphere model for a catadioptric sensor. A 3D point $\mathbf{X}$ defines a projective ray/point $\mathbf{x}$ that is mapped at $\mathbf{x}' = \hbar(\mathbf{x})$. Given an image point $\mathbf{x}' = (x', y', z')^T$, we intend to invert the mapping in order to compute its back-projection $\mathbf{x}$ in the CPP. The equation of the reference sphere in the coordinates system centered in $\mathbf{O}'$ is $x'^2 + y'^2 + (z' - \xi)^2 = 1$. Since $\mathbf{x}'$ is

a projective ray going through $\mathbf{O}'$, there is a scalar λ such that $\lambda\mathbf{x}'$ is a 3D point lying on the sphere surface. The scalar λ can be computed by solving equation $(\lambda x')^2+(\lambda y')^2+(\lambda z'-\xi)^2 = 1$. Since it is a second order equation, there are two solutions λ^+ and λ^- that are provided below.

$$\lambda^{\pm} = \frac{z'\xi \pm \sqrt{z'^2 + (1-\xi^2)(x'^2+y'^2)}}{x'^2+y'^2+z'^2} \tag{22}$$

The projective ray $\mathbf{x}'$, with origin in $\mathbf{O}'$, intersects the reference sphere in two points $\lambda^+\mathbf{x}'$ and $\lambda^-\mathbf{x}'$ (Fig. 2). By representing these points in the reference frame centered in the effective viewpoint $\mathbf{O}$ we obtain two distinct back-projections $\mathbf{x}^+$ and $\mathbf{x}^-$ (equation 23). Since the camera is forward looking the mirror, the correct solution for the back-projection is $\mathbf{x} = \mathbf{x}^+$. Point $\mathbf{x}^-$ is just a spurious algebraic solution.

$$\mathbf{x}^{\pm} = (\lambda^{\pm}x', \lambda^{\pm}y', \lambda^{\pm}z' - \xi)^T \tag{23}$$

Assume a line $\mathbf{n}$ in the CPP going through one of the back-projections of $\mathbf{x}'$. Line $\mathbf{n}$ is projected into a conic curve $\boldsymbol{\Omega'}$ that must go through the image point $\mathbf{x}'$. Therefore, and considering the embedding in $\wp^{\mathbf{5}}$, it follows that $\widetilde{\boldsymbol{\omega}}'^T\widetilde{\mathbf{x}}' = 0$. Replacing $\widetilde{\boldsymbol{\omega}}'$ by the result of equation 20 yields

$$\widetilde{\mathbf{n}}^T \underbrace{\widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}{}^T \widetilde{\mathbf{x}}'}_{\widetilde{\boldsymbol{\omega}}*} = 0. \tag{24}$$

Equation 24 is satisfied by any line $\mathbf{n}$ going through one of the back-projections of $\mathbf{x}'$ which means that $\widetilde{\boldsymbol{\omega}}^*$ is the lifted representation of a the conic envelope in the CPP. The conic envelope is degenerate (rank 2) because it is composed by two pencils of lines defined by points $\mathbf{x}^+$ and $\mathbf{x}^-$. Equation 25 is derived taking into account that $\widetilde{\boldsymbol{\omega}}^* = \boldsymbol{\Gamma}(\mathbf{x}^+, \mathbf{x}^-)$ (equation 14). The embedding in $\wp^{\mathbf{5}}$ allowed us to establish a linear relation between a point in the catadioptric image plane and the corresponding back-projections in the CPP.

$$\boldsymbol{\Gamma}(\mathbf{x}^+, \mathbf{x}^-) = \widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}{}^T \widetilde{\mathbf{x}}'. \tag{25}$$

4.3 Back-projection of image points in paracatadioptric systems and cameras with distortion

For the case of paracatadioptric systems the parameter ξ is unitary and the projective mapping of Fig. 2 becomes a stereographic projection [6]. If $\xi = 1$ then $\lambda^- = 0$ and the spurious back-projection is always $\mathbf{x}^- = (0,0,1)^T$ (equations 22 and 23). Since the re-projection center is on the sphere, the projective ray $\mathbf{x}'$ must always intersect the surface in $\mathbf{O}'$ which explains the result. Assume that the back-projection in the forward-looking direction is $\mathbf{x} = (x,y,z)$ ($\mathbf{x}^+ = \mathbf{x}$). Since $\boldsymbol{\Gamma}(\mathbf{x}^+, \mathbf{x}^-) = (0,0,0,x/2,y/2,z)^T$, it follows from equation 25 that

	Pin-Hole	Hyperbolic	Parabolic	Distortion
Pin-Hole	$\mathbf{K_y^{-T} E K_x^{-1}}$	$\widetilde{\mathbf{K}}_\mathbf{y}^{-\mathbf{T}} \widetilde{\mathbf{D}} \widetilde{\mathbf{E}} \widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}^{\boldsymbol{T}} \widetilde{\mathbf{H}}_\mathbf{x}^{-\mathbf{1}}$	$\mathbf{K_y^{-T} E \Theta}^{\boldsymbol{T}} \widetilde{\mathbf{H}}_\mathbf{x}^{-\mathbf{1}}$	$\mathbf{K_y^{-T} E K_x^{-1}} \widetilde{\boldsymbol{\Phi}}_{\boldsymbol{x}}^{\boldsymbol{T}}$
Hyperbolic	$\widetilde{\mathbf{H}}_\mathbf{y}^{-\mathbf{T}} \widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}} \widetilde{\mathbf{D}} \widetilde{\mathbf{E}} \widetilde{\mathbf{K}}_\mathbf{x}^{-\mathbf{1}}$	-	-	-
Parabolic	$\widetilde{\mathbf{H}}_\mathbf{y}^{-\mathbf{T}} \boldsymbol{\Theta} \mathbf{E} \mathbf{K}_\mathbf{x}^{-\mathbf{1}}$	-	$\widetilde{\mathbf{H}}_\mathbf{y}^{-\mathbf{T}} \boldsymbol{\Theta} \mathbf{E} \boldsymbol{\Theta}^{\boldsymbol{T}} \widetilde{\mathbf{H}}_\mathbf{x}^{-\mathbf{1}}$	$\widetilde{\mathbf{H}}_\mathbf{y}^{-\mathbf{T}} \boldsymbol{\Theta} \mathbf{E} \mathbf{K}_\mathbf{x}^{-\mathbf{1}} \boldsymbol{\Phi}_{\boldsymbol{x}}^{\boldsymbol{T}}$
Distortion	$\boldsymbol{\Phi}_{\boldsymbol{y}} \mathbf{K_y^{-T} E K_x^{-1}}$	-	$\boldsymbol{\Phi}_{\boldsymbol{y}} \mathbf{K_y^{-T} E \Theta}^{\boldsymbol{T}} \widetilde{\mathbf{H}}_\mathbf{x}^{-\mathbf{1}}$	$\boldsymbol{\Phi}_{\boldsymbol{y}} \mathbf{K_y^{-T} E K_x^{-1}} \boldsymbol{\Phi}_{\boldsymbol{x}}^{\boldsymbol{T}}$

Table 2: This table summarizes the results for the lifted fundamental matrices $\digamma$ relating pairs of views acquired by any mixture of central projection systems. The perspective image plane should not be lifted to avoid multiple solutions for the fundamental geometry (see Tab. 1). Therefore, $\digamma$ is a 3×3 matrix in the case of two pin-hole images, and a 3×6 matrix for the situation of a perspective view and a paracatadioptric/distortion view. In the remaining cases $\digamma$ is always a 6×6 square matrix.

$$\mathbf{x} = \underbrace{\begin{bmatrix} 2&0&0 \\ 0&2&0 \\ 0&0&1 \end{bmatrix} \begin{bmatrix} 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ -1&0&-1&0&0&1 \end{bmatrix}}_{\boldsymbol{\Theta}^{\boldsymbol{T}}} \widetilde{\mathbf{x}}'. \tag{26}$$

According to equation 26 there is a linear transformation that maps the lifted coordinates of a point in the paracatadioptric image into the corresponding point $\mathbf{x}$ in the CPP. Remark that the 3×6 matrix is the transpose of the three last columns of $\widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}$ when $\xi = 1$ (equation 21).

For the case of cameras with lens distortion the re-projection center is located on the vertex of the paraboloid (Fig. 1). This case is similar to the paracatadioptric system because $\mathbf{O}'$ also lies on the reference surface. The spurious back-projection $\mathbf{x}^-$ is always $(0, 0, \xi)$, and there is a 3×6 matrix $\boldsymbol{\Phi}^T$ that maps lifted image points $\widetilde{\mathbf{x}}'$ into points $\mathbf{x} = \eth^{-1}(\mathbf{x}')$.

$$\mathbf{x} = \underbrace{\begin{bmatrix} 2&0&0 \\ 0&2&0 \\ 0&0&1 \end{bmatrix} \begin{bmatrix} 0&0&0&0.5&0&0 \\ 0&0&0&0&0.5&0 \\ \xi&0&\xi&0&0&1 \end{bmatrix}}_{\boldsymbol{\Phi}^{\boldsymbol{T}}} \widetilde{\mathbf{x}}' \tag{27}$$

5 Fundamental matrices in $\wp^5$

In the experiment of section 2.3 we artificially generated a set of noise-free correspondences and investigated the dimensionality of the null space of matrix $\boldsymbol{\Psi}$ to find the mixtures of central projection systems with a fundamental matrix in lifted coordinates. The results of Tab. 1 are a good guideline, however they do not provide a geometric insight on the problem.

In this section we apply the embedding theory presented in sections 3 and 4 to explicitly derive the lifted fundamental matrices.

5.1 Views acquired by pin-hole cameras

Consider two views acquired by a pair of perspective cameras with intrinsic parameters $\mathbf{K_x}$ and $\mathbf{K_y}$. Corresponding image points $\mathbf{x'} \leftrightarrow \mathbf{y'}$ must satisfy ${\mathbf{y'}}^T\mathbf{F}\mathbf{x'} = 0$ with $\mathbf{F}$ the conventional 3×3 fundamental matrix. $\mathbf{F}$ is a correlation in the projective plane because it transforms points in one view into lines in the other view (the epipolar lines). According to section 3.3, a correlation $\mathbf{F}$ in $\boldsymbol{\wp^2}$ is lifted to $\widetilde{\mathbf{D}}\widetilde{\mathbf{F}}$ in the 5D projective space. Taking into account the result of equation 18, it follows that

$$\mathbf{F} = \mathbf{K_y^{-T}}\mathbf{E}\mathbf{K_x^{-1}} \quad \longrightarrow \quad \widetilde{\mathbf{D}}\widetilde{\mathbf{F}} = \widetilde{\mathbf{K}}_\mathbf{y}^{-\mathbf{T}}\widetilde{\mathbf{D}}\widetilde{\mathbf{E}}\widetilde{\mathbf{K}}_\mathbf{x}^{-\mathbf{1}} \tag{28}$$

The experiment of section 2.3 confirms that if ${\mathbf{y'}}^T\mathbf{F}\mathbf{x'} = 0$ then $\widetilde{\mathbf{y}}'^{\mathbf{T}}\widetilde{\mathbf{D}}\widetilde{\mathbf{F}}\widetilde{\mathbf{x}}' = 0$. Matrix $\digamma = \widetilde{\mathbf{D}}\widetilde{\mathbf{F}}$ is a lifted fundamental matrix relating two views acquired by perspective cameras. However there are other solutions for equation 9. It is easy to see that the following equation holds

$$\widetilde{\mathbf{y}}'^{\mathbf{T}} \left[\begin{smallmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F} \end{smallmatrix}\right] \widetilde{\mathbf{x}}' = 0.$$

Tab. 1 shows that the null space of matrix $\boldsymbol{\Psi}$ is multidimensional. For the case of image pairs acquired by pin-hole cameras the epipolar geometry is described by a bilinear form in $\boldsymbol{\wp^2}$. Since there is a fundamental matrix $\mathbf{F}$, the lifting to $\boldsymbol{\wp^5}$ is just an over parameterization that leads to multiple $\digamma$ solutions.

5.2 Views acquired by paracatadioptric sensors and cameras with radial distortion

Consider the scheme of Fig. 1 showing a camera with lens distortion and a paracatadioptric system. The projection centers are respectively $\mathbf{O_x}$ and $\mathbf{O_y}$, $\mathbf{K_x}$ represents the intrinsic parameters of the dioptric camera (equation 3), and $\mathbf{H_y}$ encodes the calibration of the paracatadioptric sensor (equation 1). The 3D point $\mathbf{X}$ defines two projective rays, $\mathbf{x}$ and $\mathbf{y}$, going through the effective viewpoints $\mathbf{O_x}$ and $\mathbf{O_y}$. The two points verify $\mathbf{y}^T\mathbf{E}\mathbf{x} = 0$ where $\mathbf{E}$ stands for the conventional essential matrix. Point $\mathbf{x}$ is mapped in the distorted image plane at $\mathbf{x'} = \eth(\mathbf{K_x}\mathbf{x})$ (equation 3). From equation 27 follows that $\mathbf{x} = \mathbf{K_x^{-1}}\boldsymbol{\Phi}_x^T\widetilde{\mathbf{x}}'$. The projection in the paracatadioptric image plane is $\mathbf{y'} = \mathbf{H_y}\hbar(\mathbf{y})$ and the inverse mapping is $\mathbf{y} = \boldsymbol{\Theta}^T\widetilde{\mathbf{H}}_\mathbf{y}^{-1}\widetilde{\mathbf{y}}'$ (equations 1, 17, 18 and 26). Replacing $\mathbf{x}$ and $\mathbf{y}$ in the essential relation yields

$$\widetilde{\mathbf{y}}'^{\mathbf{T}} \underbrace{\widetilde{\mathbf{H}}_\mathbf{y}^{-\mathbf{T}}\boldsymbol{\Theta}\mathbf{E}\mathbf{K}_\mathbf{x}^{-\mathbf{1}}\boldsymbol{\Phi}_{\boldsymbol{x}}^{\boldsymbol{T}}}_{\digamma}\widetilde{\mathbf{x}}' = 0 \tag{29}$$

The derived matrix $\mathcal{F}$ is the 6×6 fundamental matrix observed in section 2.3 for the mixture of distorted and paracatadioptric views. Remark that $\mathcal{F}$ is still a correlation in $\wp^5$, transforming the lifted coordinates of points in one view into the corresponding epipolar curves in the other view ($\widetilde{\boldsymbol{\omega}}'_{\boldsymbol{y}} = \mathcal{F}\widetilde{\mathbf{x}}'$ and $\widetilde{\boldsymbol{\omega}}'_{\boldsymbol{x}} = \mathcal{F}^{\boldsymbol{T}}\widetilde{\mathbf{y}}'$). The reasoning to derive the lifted fundamental matrices for mixtures of two paracatadioptric sensors and two cameras with radial distortion is similar. The results are presented in Tab. 2.

For views acquired by a parabolic sensor and a pin-hole camera there are multiple 6×6 fundamental matrices $\mathcal{F}$ satisfying equation 8 (Tab. 1). As discussed above, by lifting the point coordinates in a perspective image we end up with an over-parameterization that generates multiple solutions. The problem is solved by using lifted coordinates only for the paracatadioptric view. Equation 30 shows the corresponding 3×6 fundamental matrix. The case of a pin-hole and a camera with lens distortion is identical.

$$\mathbf{y}'^{\mathbf{T}} \underbrace{\mathbf{K}_{\mathbf{y}}^{-\mathbf{T}} \mathbf{E} \boldsymbol{\Theta}^{\boldsymbol{T}} \widetilde{\mathbf{H}}_{\mathbf{x}}^{-\mathbf{1}}}_{\mathcal{F}_{3\times 6}} \widetilde{\mathbf{x}}' = 0 \tag{30}$$

The division model is a simple second order model that requires the center to be known and the distortion to be isotropic [10]. There are other distortion models without such limitations [14, 8]. Remark that our framework can be used with any non-linear projection model, as far as there is a 3×6 linear transformation between lifted image coordinates and undistorted points (equation 27).

5.3 Views acquired by hyperbolic sensors

According to the synthetic experiment of section 2.3, the lifting of coordinates fails in enforcing a bilinear form for the epipolar geometry of mixtures that include hyperbolic sensors. As explained in section 4.2, the lifted representation of an image point $\widetilde{\mathbf{x}}'$ can be mapped by a linear transformation into a conic envelope that encodes the correct back-projection $\mathbf{x} = \mathbf{x}^+$, and a spurious solution $\mathbf{x}^-$. For the case of paracatadioptric systems and cameras with lens distortion the spurious solution is constant and there is a linear transformation that maps $\widetilde{\mathbf{x}}'$ in $\mathbf{x}$. The results of equations 26 and 27 proved to be crucial in deriving the lifted fundamental matrices for mixtures involving these systems. For the case of hyperbolic sensors it is not possible to decouple $\mathbf{x}^-$ and $\mathbf{x}^+$, which explains the non existence of fundamental matrices. One possible solution is to increase the dimensionality of the Veronese lifting. However, the corresponding estimation problem would probably be non tractable. An alternative is to find an embedding that simultaneously encodes orientation and preserves homogeneity. Such embedding, as far as we are aware, does not exist.

The only case that admits a lifted fundamental matrix is the combination with a perspective view. The structure of the corresponding 6×6 fundamental matrix $\mathcal{F}$ is provided in equation 31. Curiously, the epipolar curve in the perspective plane $\widetilde{\boldsymbol{\omega}}'_{\boldsymbol{y}} = \mathcal{F}\widetilde{\mathbf{x}}'$ is a rank

2 conic. This conics is composed by two lines: the forward looking epipolar line and the backward looking epipolar line.

$$\widetilde{\mathbf{y}}'^{\mathbf{T}} \underbrace{\widetilde{\mathbf{K}}_{\mathbf{y}}^{-\mathbf{T}} \widetilde{\mathbf{D}} \widetilde{\mathbf{E}} \widetilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}^{\boldsymbol{T}} \widetilde{\mathbf{H}}_{\mathbf{x}}^{-\mathbf{1}}}_{F} \widetilde{\mathbf{x}}' = 0 \tag{31}$$

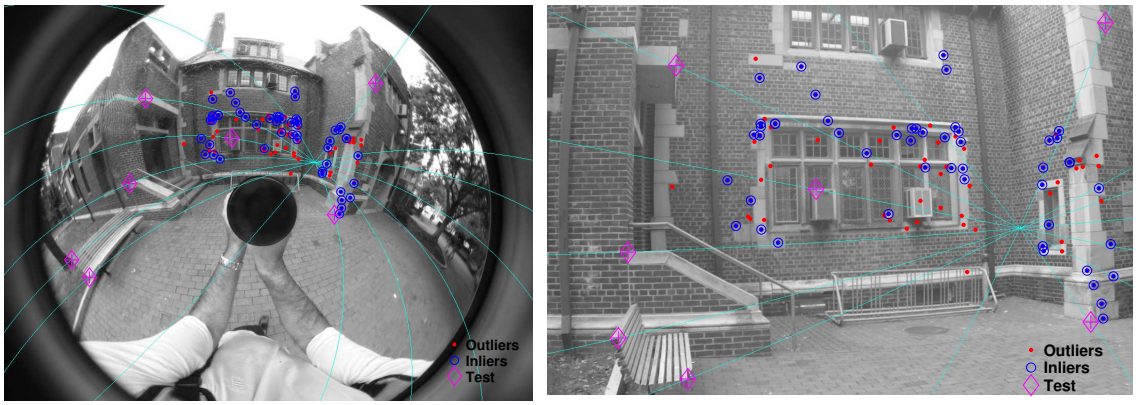


Figure 3: Estimation of the lifted fundamental matrix for views acquired by a paracatadioptric sensor and a camera with radial distortion. The Lowe's detector found 101 correspondences from which 54 were marked as inliers. The high rejection percentage is explained by the difficulty in establishing robust correspondences between images that look so different.

6 Experiments with real images

Tab. 2 shows the mixtures of central projection systems for which there are lifted bilinear constraints between views. For these cases, and under the assumption of an ideal noise-free situation, the computation of the fundamental matrix F from a set of image correspondences is straightforward. As shown in equation 9, we just need to build matrix $\boldsymbol{\Psi}$ and determine its right null space. In practice the correspondences are always affected by noise and matrix $\boldsymbol{\Psi}$ is in general full rank. In this situation we can always apply SVD decomposition to enforce a null space and estimate F. The problem is that not every 6×6 matrix is a lifted fundamental matrix. As discussed in the previous section, F must be a rank 2 matrix with a specific structure that depends on the type of central cameras used to acquire the views (Tab. 2). In [5], Claus et al. report an experiment in estimating the fundamental matrix between two views acquired with a fish-eye lens using a standard linear algorithm. According to the report, they succeeded in obtaining a bilinear form that seems to fit the data, however they were unable to extract correct distortion information. This suggests that the use of purely algebraic approaches is not enough, and that the structure of F must be taken into account.

Figure 4: Correction of the Radial Distortion in a 853×1280 image. The estimated distortion was 145.7 pixels at the image corner

To prove the applicability of our framework we ran an experiment to estimate the geometry between two uncalibrated images acquired by a paracatadioptric system and a camera with lens distortion (Fig. 3). The estimation problem is simplified by considering a skewless parabolic system with unitary aspect ratio. Under this assumption the lifting of coordinates is similar to the one used in [7, 12], and the dimension of $\mathcal{F}$ is 4×4. From the correspondences we build a matrix $\mathbf{\Psi}$ (equation 9) and estimate $\mathcal{F}$ using linear least squares. The rank constraint on $\mathcal{F}$ is enforced using SVD. From the analysis of the structure of $\mathcal{F}$ it follows that the null space on the side of the distorted view encodes the undistorted epipole and the amount of distortion. This information is extracted and used as a prior to re-fit the matrix to the correspondences. The sub-optimal two step factorization approach is entirely linear and provides a fundamental matrix with the desired structure (for further details see [3]). Since the estimation has a closed-form solution, it can be run in a RANSAC approach to discard outliers. Fig. 3 shows the results in estimating the lifted fundamental matrix. The correspondences were automatically detected using SIFT features [9]. To test the correctness of the result we manually selected 7 correspondences and drew the corresponding epipolar curves (Fig. 3). Additionally we extracted the radial distortion information from the estimated $\mathcal{F}$ and corrected the distorted view (Fig. 4).

7 Conclusions

We proposed a representation for central projection images through a lifting of the projective plane to the 5D projective space. In addition, we presented a full embedding theory to transfer geometric entities and relations between the original and lifted spaces. The theory was applied to explicitly construct the lifted fundamental matrices and understand their structure. We believe that such geometric insight is essential to develop robust estimation algorithms and extract information from the estimated matrices.

References

[1] S. Baker and S. Nayar. A theory of catadioptric image formation. In *Proc. of ICCV*, 1998.

[2] J. P. Barreto and K. Daniilidis. Unifying liftings for catadioptric and dioptric cameras. In *Proc. of OMNIVIS*, 2004.

[3] J. P. Barreto and K. Daniilidis. Fundamental matrix for cameras with radial distortion. In *Proc. of ICCV*, 2005.

[4] C. Brauer-Burchardt and K. Voss. A new algorithm to correct fish-eye- and strong wide-angle- lens-distortion from single images. In *Proc. of ICIP*, 2001.

[5] D. Claus and A. Fitzgibbon. A rational function for fish-eye lens distortion. In *Proc. of CVPR*, 2005.

[6] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems. In *Proc. of ECCV*, 2000.

[7] C. Geyer and K. Daniilidis. Structure and motion from uncalibrated catadioptric views. In *Proc. of CVPR*, 2001.

[8] R. Hartley and S. B. Kang. Parameter-free radial distortion correction with center of distortion estimation. In *Proc. of ICCV*, 2005.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[10] B. Micusik and T. Padjla. Estimation of omnidirectional camera model from epipolar geometry. In *Proc. of CVPR*, 2003.

[11] J. G. Sample and G. T. Kneebone. *Algebraic Projective Geometry*. Claredon Press, 1998.

[12] Peter Sturm. Mixing catadioptric and perspective cameras. In *Proc. of OMNIVIS*, 2002.

[13] T. Svoboda and T. Pajdla. Epipolar geometry for central catadioptric cameras. *IJCV*, August 2002.

[14] S. Thirthala and M. Pollefeys. The radial trifocal tensor: A tool for calibrating radial distortion of wide-angle cameras. In *Proc. of CVPR*, 2005.

[15] Z. Zhang. On the epipolar geometry between two images with lens distortion. In *Proc. of ICPR*, 1996.

On topology of G-configuration spaces of polyhedra

M. Rostami*

Abstract

The family $\mathcal{P}$ of all convex 3-polytopes P in Euclidean space E^3 may be partitioned into combinatorial types or *configuration spaces* by isomorphism of face lattices, and the configuration space $[P]$ of any such 3-polytope P may be subdivided further into G-Configuration space $\langle P\rangle$ by equivalence of actions of symmetry group $G(P)$ on face lattices. With respect to a natural topology induced by Hausdorff metric on $\mathcal{P}$, each $[P]$ is a contractible manifold of a certain dimension related to the number of edges of P. This is a consequence of Steinitz's fundamental theorem of convex polyhedra. In this article we prove an extension of this theorem which states that if P is a convex 3-polytope with symmetry group $G(P)$, then the G-Configuration space $\langle P\rangle$ is also a smooth manifold. The dimension , $\dim\langle P\rangle$ of this manifold is calculated with respect to the orbits of the action of symmetry group $G(P)$ on face lattice of P .

1 Introduction

A 3- polytope or **polyhedron** is a convex- hull of finitely many points in Euclidean space E^3 which are not all coplanar. Polyhedra are fundamental objects in geometry and key components in robot motion planning as well [4, 5, 9].

For any polyhedron P we denote the set of all vertices, edges and faces of P by $F_0(P)$, $F_1(P)$, $F_2(P)$, respectively. These sets, together with empty set $\emptyset$ and P itself, form a lattice $F(P)$ under set inclusion. Let us label the the polygonal faces of P by $A_1, A_2, \ldots, A_s$ and the vertices by $v_1, v_2, \ldots, v_r$. Define $M(P) = (m_{i,j})$ an $r \times s$ matrix with $m_{i,j} = 1$ if $v_i \in A_j$ and 0 otherwise. The number of non- zero elements of $M(P)$ is called multiplicity of P and denoted by $\mu(P)$ [6]. Clearly we have $\mu(P) = 2e$, where e is the number of edges of polyhedron P. Two polyhedra P and Q are said to be combinatorially equivalent, denoted by $P \approx Q$, if their face lattices are isomorphic .We denote by $[P]$ the set of all polyhedra that are equivalent to P.

Let $\Gamma = (V, E)$ be a graph with vertex set V and edge set E. A connected graph Γ is called 3-connected if the deletion of any two vertices in V together with their corresponding edges in E leaves the graph Γ connected. Γ is a planar graph if it can be embedded on the

*Departamento de Matemática, Universidade da Beira Interior. E-mail:rostami@mat.ubi.pt.

plane without edge crossing. The following classical theorem, called fundamental theorem of convex polyhedra, is due to Steinitz [7, 9].

Theorem 1.1 (Steinitz)**.** *The edge graphs of convex polyhedra are exactly the simple, planar and 3-connected graphs.*

The following should also be taken as a part of the Steinitz's Fundamental Theorem.

Theorem 1.2 (Steinitz)**.** *[7, 9] For every polyhedron P, the* configuration space $[P]$ *is a smooth manifold of dimension* $\dim[P] = e + 6$*, where e is the number of edges of P. Furthermore, $[P]$ has isotopy property in the sense that if $Q, Q' \in [P]$ then there is a continuous one-parameter family of configurations that starts with Q and ends with Q', i.e., a path in E^{3n} connecting Q and Q' such that each point of the path represented by a polyhedron combinatorially equivalent to P.*

For each polyhedron P a symmetry of P is a rigid transformation of E^3 that preserves P set-wise. Any such symmetry maps vertices to vertices, edges to edges and faces to faces and preserves inclusions (incidences). Hence any symmetry induces an automorphism on $F(P)$. The set $G(P)$ of all symmetries of P is a finite subgroup of the Euclidean group $E(3)$, acting on $F(P)$ as a group of automorphisms. We can take $G(P)$ as a finite subgroup of orthogonal group $O(3)$. Two polyhedra P and Q are said to be symmetry equivalent [6] if the action of $G(P)$ on $F(P)$ is equivalent to the action of $G(Q)$ on $F(Q)$. This means that there is an isomorphism $\lambda : F(P) \to F(Q)$ of face lattices and isometry $f : E^3 \to E^3$ such that for all $g \in G(P)$ and all $x \in F(P)$, $\lambda g(x) = (f^{-1} \circ g \circ f)(\lambda(x))$.

If we further assume that P and Q have the same (rather than conjugate) subgroups then $\lambda(g(x)) = g(\lambda(x))$. In this case we write $P \cong Q$, and say that P and Q are *G–equivalents*.

By a *configuration* we mean an indexed collection of planes and points in E^3 such that the points being indexed by vertices of P, and the planes indexed by polygonal faces contained on those planes. Hence there is an obvious correspondence between a *configuration* and a polyhedron P. We can topologize $[P]$ by Hausdorff metric and call it *Configuration space* or *C-space* of P.

Now, *C-space* $[P]$ of polyhedron P may be refined by taking the symmetry group $G(P)$ into account. By the proof of the Steinitz's theorem, we know that $[P]$ is a contractible manifold of dimension $\dim[P] = e + 6$. Let $\langle P \rangle$ denote the set of all polyhedra G–equivalent to P. Then, clearly $[P]$ is a union of these symmetry types, one of which is $\langle P \rangle$ itself. We call $\langle P \rangle$ the *configuration space* or *G–space* of P.

2 Transformation groups and slice theorem

A Lie group is a manifold and a group such that the operations (multiplication and inversion) of the group are continuous. The actions of Lie groups on manifolds result orbit spaces. The

structure of these obits is usually quite complicated. But sometimes it can be shown that they are stratified into smooth manifolds.

The stratification mainly is done by the help of a theorem, called **Slice Theorem**, which is fundamental in studying the structure of **Transformation Groups**. Hence we give a brief resumé of the relevant facts about this theorem and referring the reader to [2] and [6] for details.

Let M be a smooth manifold and G a compact subgroup of the orthogonal group $O(3)$ acting smoothly on M via:

$$\varphi : G \times M \to M.$$

Then, $(M, G) = (M, G, \varphi)$ is called a transformation group and M is called a G–manifold. If the action is transitive (having precisely one orbit) then M is called homogeneous space.

For each $x \in M$, $G(x) = \{g(x) : \ x \in G\}$ is an orbit and $G_x = \{g \in G : \ g(x) = x\}$ is the stability subgroup of G at x. If H is closed subgroup in G then the orbit type of H is the subset of M of those points $x \in M$ such that G_x is conjugate to H. So each orbit type is a union of G–orbits. Hence it is possible to write the orbit space M/G as disjoint union of orbit types. Thus a homogeneous space is a transformation group of the form $(G/H, G)$, $x \in M$. For any compact transformation group (M, G) the orbit $G(x)$ through x is a compact homogeneous space embedded in M. The statements in the following theorem are among the consequences of the Slice Theorem.

Theorem 2.1. *[2] Let G be a compact Lie group and M a (smooth) G–manifold. Then we have:*

1. *The orbit $G(x)$ of x is a G–invariant submanifold of M, $x \in M$;*
2. *If every orbit in M has type G/G_x, $x \in M$, then the orbital space M/G is a smooth manifold;*
3. *If H is a closed subgroup of G, then the union $M(H)$ of the orbits of type G/H is a G–invariant submanifold of M. Furthermore, the orbit space $M(H)/G$ again is a smooth manifold of M. For the proof see [2, 4.10, 4.18 and 4.19]*

3 The topology of GC–spaces

Now consider P, with centroid O and symmetry group $G \subseteq O(3)$ and let $F_x = \operatorname{fix} G_x = \left\{y \in E^3 : \text{ for all } g \in G_x, \ g(y) = y\right\}$ be the fixed point set of G_x. Thus F_x is linear subspace of E^3. Define an equivalence relation $\sim_G$ in E^3 by $x \sim_G y \Leftrightarrow F_x = F_y$. Now $y \in F_x$ implies that $G_x \subseteq G_y$, $F_x \subseteq F_y$ and the equivalence class $[x] \subseteq F_x$. Then the equivalence classes $[x]$, $x \in E^3$ **stratify** E^3 by finitely many such strata (orbit types) [2]. Hence $[x]$ is a union of

G–orbits. In particular $G(x)$ partitions in E^3 into finitely many orbital types of G [6]. Now, consider $\langle P\rangle$ and let $Q \in \langle P\rangle$. Each vertex of Q may be moved along a line within a plan or in any direction in E^3 in a small neighbourhood of its original position in P itself, likewise each face of polyhedron close to the corresponding face A of P , according as A intersects a one-stratum in an exterior point of A (necessarily at right angle) or intersects 2-stratum in interior of A, or neither of these. Let v be a vertex of P. Then, $\delta(v) = k$, $k = 1, 2, 3$ if v lies on a k-stratum of G, and $\delta(A) = k$, $k = 1, 2, 3$ if the face A has k degrees of freedom within the above restrictions.

Let $\overline{v} = \{g(v) : \ g \in G\}$, $\overline{A} = \{g(A) : \ g \in G\}$, $F_0(P)/G = \{\overline{v} : \ v \in F_0(P)\}$ and $F_2(P)/G = \left\{\overline{A} : \ A \in F_2(P)\right\}$. We denote by $M(P)$ the set of all pairs $(v, A) \in F_0(P) \times F_2(P)$ for which $v \in A$, and $M(P)/G = \{(g(v), g(A)) : \ g \in G \text{ and } (v, A) \in M(P)\}$. Denote by $\mu_*(P)$ the cardinality of $M(P)/G$. Recall that $\mu(P)$ is the number of incidences between the vertices and faces of P. Hence $\mu_*(P)$ is the number of orbits of such incidences under the action of $G(P)$ on $F(P)$. For example, if P is square right pyramid (Figure 1), then $\mu(P) = 2e = 16$, but $\mu_*(P) = 1 + 2 = 3$.

Clearly, if $v, u \in F_0(P)$ and $\overline{v} = \overline{u}$ then the dimension of F_v is equal to the dimension of F_u, i.e., $\delta(v) = \delta(u)$. The same holds for the faces of P. The following part of the theorem will be sketchy, adopted from Robertson [6], we refer the reader there for further details.

Theorem 3.1. *Let P be polyhedron in E^3 then $\langle P\rangle$, the GC–space of P, is a smooth manifold.*

Proof: Let N_P be the set of all polyhedra in $[P]$ with centroid O and say radius one [6]. Then N_P is a submanifold of codimension 4, by simply factoring out the effects of dilations and translations. This is called the submanifold of normal polyhedra. Hence, $(N_P, O(3))$ is a compact transformation group. For each Q in N_P, the isotropy group $O(3)_Q$ is just the symmetry group $G(Q)$ of Q [6]. But $(N_P, O(3))$ is a differentiable transformation group and according to Theorem 2.1 the orbit space $N_P/O(3)$ is smooth manifold .Consequently, $\langle P\rangle$ is smooth manifold as well [6]. □

As an example, let P be a polyhedron combinatorially equivalent to cube, "cuboid" [6]. Then $[P]$ is an 18-manifold and the principal orbit type corresponds to the configuration of polyhedra $Q \approx P$ with $G(Q) = 1$ is an open submanifold of $[P]$ of dimension 18. All other configuration spaces have dimensions lower than 18.

Now, according to Theorem 1.2 the configuration space $[P]$ of polyhedron P is a smooth manifold of dimension $e + 6$ where e is the number of edges of P.

An intuitive derivation of the dimension of the GC–space $\langle P\rangle$ can be given as follows: Suppose that the polyhedron P has n vertices, e edges and f faces. If the vertices of P were allowed to move independently in E^3, they would have $3n$ *degrees of freedom.* However, it

requires k vertices to specify a k-gonal face $k \geq 3$, and consequently the d losses $k-3$ degrees of freedom. Thus the whole *configuration space* $[P]$ has $d = 3n - \sum_{k\geq 3}(k-3)$ *degrees of freedom*; the sum being taken over all faces of P.

Now, using x_k to denote the number of k-gonal faces of P, and taking into account that each edge is contained in two faces, we have $\sum kx_k = 2e$ and the dimension of the GC–space $\dim[P] = 3n - \sum_{k\geq 3}(k-3) = 3n - kx_k + 3f = 3(n+f) - 2e = 3(n+f) - \mu(P)$. By Euler formula in convex polyhedra , we have $n - e + f = 2$. Hence $\dim[P] = e + 6$.

Consider a square right pyramid (Figure 1):

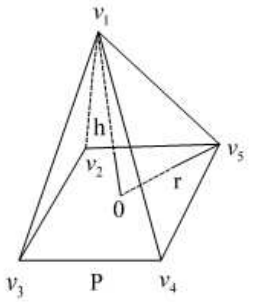


Figure 1.

Now, $G(P)$ the symmetry group of P is dihedral group. Suppose we fix the group $G = G(P)$. Then vertex v_1 can be chosen only on the axis of G. Therefore it has *one degree of freedom*. Likewise v_2 must lie on reflection plane, hence has only two degrees of freedom. Having chosen v_2, the vertices v_3 and v_4 which are on the same orbit of v_2, have no degree of freedom at all, since they are determined by our choice of G and v_2. Hence the vertices have a total of $1 + 2 = 3$ degrees of freedom. Similarly for the faces, each triangular face or equivalently the plane that contains it has only two degrees of freedom in the space of affine plane in E^3, since each plane is invariant under a reflection element of G. But the square face has only one degree freedom, because it is orthogonal to the axis of rotation of G. Therefore the faces have just $2 + 1 = 3$ degrees of freedom. Of course the faces and vertices can not be chosen independently of one another. The incidence of v_1 with respect to any of the four triangular faces adjacent to it, determines the incidence of that vertex to the other three faces under the action of G. Hence v_1 has only one "independent" incidence. The vertices v_2, v_3 and v_4 are in the same G-orbit and each one is incident with two triangles

and one square faces. Take one of them say v_2. There is a reflection which fixes v_2 and sends adjacent triangular faces each one to the other. Thus the number of independent incidences $\mu_*(P)$ of P is $1+2=3$. Each such incidence relation in the form of the condition that a vertex lies in a particular face, reduces the dimension of the *GC-space* by one. Hence we get : $\dim\langle P\rangle = (1+2)+(1+2)-(1+2) = 3$.

The idea of this example can be applied in general to find the dimension of the symmetry type of any polyhedron P. Indeed, we can factor out the relation $\dim[P] = 3(n+f) - \mu(P)$ by the action of $G(P)$ on $F(P)$ to find the dimension of the *GC-space*, $\dim\langle P\rangle = \sum \lfloor \delta(\overline{v}) + \delta(\overline{A}) \rfloor - \mu_*(P)$, where the summation $\sum$ is extended over all vertex orbits $\overline{v} \in F_0(P)/G$ and face orbits $\overline{A} \in F_2(P)/G$.

References

[1] M. Farber and M. Grant. Symmetric motion planning. math.AT] 27 Oct.2006.

[2] K. Kawakubo. *The theory of transformation groups*. Oxford University Press, 1991.

[3] Y. Liu and Popplestne. Symmetry constraint inference in assembly planning automatic assembly configuration specification. *Proc.of AAAI-90*, pages 1038–1044, 1990.

[4] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Tran. Comput.*, C–32:108–120, 1983.

[5] T. Lozano-Pérez and M. Wesley. An algorithm for planning collision free paths among polyhedral obstacles. *Comm. ACM*, 22(10):560–570, 1979.

[6] S. A. Robertson. *Polytopes and symmetry*. Number 90 in London Math. Soc. Lecture Notes. Cambridge University Press, 1984.

[7] E. Steinitz and H. Rademacher. *Vorlesugen uber die theorie der polyhder*. Springer, 1934. reprint 1976.

[8] Z. H. Xiong and Z. X. Li. On the discrete symmetric localization problem. In *Proceedings of the 2002 IEEE International Conference on Robotics*, Washington, DC, 2002.

[9] G. M. Ziegler. *Lectures on polytopes*. Number 152 in Graduate Texts in Mathematics. Springer, New York, 1995.

Generalized least squares problems on Riemannian manifolds

L. Machado* F. Silva Leite† K. Hüper ‡

Abstract

Our objective is to present a generalization of classical least squares method to Riemannian manifolds M. The main drawback to adapt the classical approach relies on the non availability of explicit forms for polynomial curves on non-Euclidean spaces. This difficulty is overcome with a convenient formulation of a variational problem whose solutions turn out to give rise to curves that best fit a given set of data on M and are called "smoothing geometric splines". A detailed analysis of this approach when M is the Euclidean space $\mathbb{R}^n$ supports our strong believe that this is the natural generalization.

1 Introduction

A typical task in robotics is to plan movements from a given end-effector configuration (position and orientation) to a desired new configuration. Since the configuration space of most mechanical systems fails to be a vector space and typically has components which are smooth manifolds, such as the rotation group or the Euclidean group of rigid motions, problems of path planning and tracking in robotics have been an important source of motivation for the study of interpolation problems on non-Euclidean spaces. References where the classical interpolation problems have been replaced by generalizations to Riemannian manifolds are already quite vast and vary from deep mathematical foundations [17], [6, 7], [5], [10], [13], [2, 3], to more applied results, where low dimensional manifolds are at play [19], [11], [8], [18], [4].

Most of the cited work deals with geometric cubic splines and the main feature of these interpolating curves relies on the fact that they minimize that component of acceleration that lies tangent to the manifold.

*Institute of Systems and Robotics, University of Coimbra, and Department of Mathematics, University of Trás-os-Montes e Alto Douro, 5000-911 Vila Real, Portugal. E-mail: lmiguel@utad.pt.

†Institute of Systems and Robotics, and Department of Mathematics, University of Coimbra, 3001-454 Coimbra, Portugal. E-mail: fleite@mat.uc.pt.

‡National ICT Australia, Canberra Research Laboratory, Locked Bag 8001,Canberra ACT 2601, Australia, and Department of Information Engineering, Research School of Information Sciences and Engineering, The Australian National University, Canberra ACT 0200, Australia. E-mail: knut.hueper@nicta.com.au.

The study of higher order geometric splines has also been required in the context of robotic motion planning. Such is the case when the robot mechanisms, instead of having just rigid components, have a complex deformable attachment like fluid or elastic components [1]. These higher order geometric splines have also been defined using a variational approach, but contrary to what happens in Euclidean spaces, no explicit solutions are know for generalizations of splines to Riemannian manifolds.

There are many circumstances, however, when interpolation is not a crucial requirement. Such is the case when the data can't be measured exactly and is corrupted by noise. A more realistic alternative is to require that the curve passes close to the data, and such small deviation usually results in a significant decrease in the computational cost, making them more appropriate for many problems arising in physics and engineering.

Our objective is to address the problem of finding smoothing splines on non-Euclidean spaces that best fit a given set of points at given instants of time, generalizing the classical least squares problems to Riemannian manifolds. These problems are well understood in Euclidean spaces, but a straightforward generalization would require that we have at hand explicit formulas for the analogues of polynomials. This was the main drawback in generalizing least squares problems on manifolds. However, since polynomials on Euclidean spaces can be seen as solutions of certain variational problems, our approach to least squares problems is also variational.

2 Revisiting the classical least squares problem

The classical least squares method can be seen as a typical method for curve fitting on Euclidean spaces. In this classical method, we are given a set of $N+1$ points in $\mathbb{R}^n$, $p_0, p_1, \ldots, p_N$, and a monotone increasing sequence of instants of time $t_0 < t_1 < \cdots < t_N$, and the objective is to find among the class of polynomial functions of degree not exceeding m ($m \leq N$), that we denote by $\mathcal{P}_m$, a polynomial parameterized by $t \mapsto \gamma(t) = a_0 + a_1 t + \ldots + a_m t^m \in \mathbb{R}^n$, that best fits the given data set of points, in the sense that the functional

$$E(\gamma) = \sum_{i=0}^{N} \left\| p_i - \gamma(t_i) \right\|^2,$$

should be as small as possible, where $\left\| \cdot \right\|$ denotes the Euclidean norm in $\mathbb{R}^n$.

It is well known [12], that for each $m \in \mathbb{N}$, there is a unique polynomial in $\mathcal{P}_m$ minimizing the functional E above.

3 Generalization to Riemannian manifolds

The non availability of explicit forms to the analogues of polynomial curves on general Riemannian manifolds is the main drawback for presenting the straightforward generalization of

the classical least squares approach given in section 2. However, when M is a connected and complete manifold, equipped with a Riemannian metric $\langle \cdot, \cdot \rangle$ and corresponding Riemannian distance d, we can formulate a variational problem which gives rise to the natural generalization of the classical least squares problem. In what follows, we first formulate this variational problem, then find the corresponding necessary optimality conditions and finally present the case when the manifold is the Euclidean space. This detailed analysis supports our conviction that the presented variational approach is the natural generalization of the classical least squares problems to non-Euclidean spaces.

In what follows, $\frac{D}{dt}$ denotes the covariant derivative along a curve and R plays the role of the curvature tensor on M.

3.1 A variational approach

We consider a collection of N points, $p_0, \ldots, p_N$, on M and a partition of the time interval $[0,1]$: $0 = t_0 < t_1 < \cdots < t_N = 1$, and look for an appropriate admissible curve $\gamma : [0,1] \to M$, that minimizes the functional

$$J(\gamma) = \frac{1}{2} \sum_{i=0}^{N} d^2 \left(p_i, \gamma(t_i)\right) + \frac{\lambda}{2} \int_0^1 \left\langle \frac{D^m \gamma}{dt^m}, \frac{D^m \gamma}{dt^m} \right\rangle dt, \tag{1}$$

over the class Ω, of all C^{m-1} paths $\gamma : [0,1] \longrightarrow M$ such that $\gamma\Big|_{[t_i, t_{i+1}]}$ is smooth, for $i = 0, \ldots, N-1$, and, in addition, has bounded limits $\lim\limits_{t \to t_i^-} \frac{D^j \gamma}{dt^j}(t)$ and $\lim\limits_{t \to t_i^+} \frac{D^j \gamma}{dt^j}(t)$, for every integer $j \geq m$. d is defined for points p and q sufficiently close [9], as $d^2(p,q) = \left\langle \exp_p^{-1} q, \exp_p^{-1} q \right\rangle$. The parameter $\lambda \in \mathbb{R}^+$ controls the trade-off between the smoothness of the fitted curve and its closeness to the given points and can therefore be considered as a smoothing parameter. For the above reasons, the authors have introduced the term "smoothing geometric splines" to characterize the solutions of the above variational problem.

Remark 3.1. *We note that the integral term in* (1) *is the functional which gives rise to analogues of higher order polynomials on Riemannian manifolds, as explained in* [5].

3.2 Necessary optimality conditions

We start the analysis of the generalized least squares problem by presenting the necessary optimality conditions for the functional J given by (1).

Theorem 3.1. [14] *A necessary condition for γ to be an extremal for the functional J, given by* (1)*, over the class Ω, is that γ is C^{2m-2} in the whole interval $[0,1]$ and for $t \in [t_i, t_{i+1}]$ and $i = 0, \ldots, N-1$, γ satisfies*

$$\frac{D^{2m}\gamma}{dt^{2m}} + \sum_{j=2}^{m} (-1)^j R\left(\frac{D^{2m-j}\gamma}{dt^{2m-j}}, \frac{D^{j-1}\gamma}{dt^{j-1}}\right) \frac{d\gamma}{dt} = 0. \tag{2}$$

Moreover, at the knot points t_i, γ satisfies the following regularity conditions

$$\frac{D^j\gamma}{dt^j}(t_i^+) - \frac{D^j\gamma}{dt^j}(t_i^-) = \begin{cases} 0; & j = 0, \ldots, m-1, \quad (i = 1, \ldots, N-1) \\ 0; & j = m, \ldots, 2m-2, \quad (i = 0, \ldots, N) \\ \frac{(-1)^m}{\lambda} \exp^{-1}_{\gamma(t_i)}(p_i); & j = 2m-1, \quad (i = 0, \ldots, N) \end{cases}, \tag{3}$$

where we assume for the sake of simplicity that

$$\frac{D^j\gamma}{dt^j}(0^-) = \frac{D^j\gamma}{dt^j}(1^+) = 0, \quad j = m, \ldots, 2m-1.$$

By the analysis of the necessary optimality conditions for the previous minimization problem, we can see that smoothing geometric splines are thus obtained by piecing smoothly together segments of geometric polynomials of degree $2m-1$ and satisfy some regularity conditions at the knot time points t_i's.

The complexity of the nonlinear differential equation (2) encountered by using the Lagrangian formalism unable us to write down the explicit solution for the minimization problem stated in subsection 3.1, unless we particularize M to be the Euclidean space $\mathbb{R}^n$. This is the goal of the next subsection.

3.3 Particular case when $M = \mathbb{R}^n$

When M is the Euclidean space $\mathbb{R}^n$, the covariant derivative coincides with the usual derivative and the curvature tensor vanishes. Therefore, the differential equation (2) reduces simply to

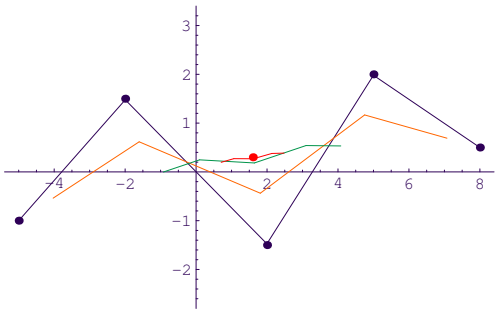
$$\frac{d^{2m}\gamma}{dt^{2m}}(t) = 0, \quad \forall t \in \big[t_i, t_{i+1}\big], \tag{4}$$

and the regularity conditions (3) become

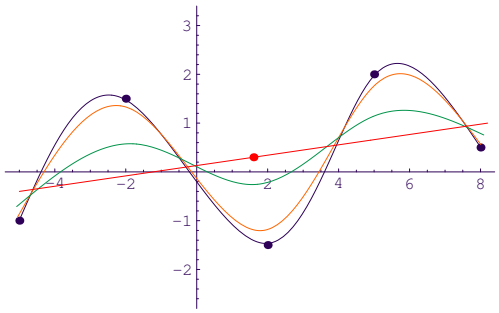
$$\frac{d^j\gamma}{dt^j}(t_i^+) - \frac{d^j\gamma}{dt^j}(t_i^-) = \begin{cases} 0; & j = 0, \ldots, m-1, \quad (i = 1, \ldots, N-1) \\ 0; & j = m, \ldots, 2m-2, \quad (i = 0, \ldots, N) \\ \frac{(-1)^m}{\lambda}\big(p_i - \gamma(t_i)\big); & j = 2m-1, \quad (i = 0, \ldots, N) \end{cases}. \tag{5}$$

The above variational problem in the Euclidean space has a unique solution (see [14], for more details) that can be achieved by integrating the differential equation (4) in each subinterval $\big[t_i, t_{i+1}\big]$ and the spline coefficients can then be found using the regularity conditions (5). In [16] and [15], the authors presented the explicit solutions for the cases when $m = 1$ and $m = 2$, respectively. It has also been proved that when the smoothing parameter λ goes to $+\infty$, the smoothing splines approach, respectively, the center of mass of the given points and the linear regression line.

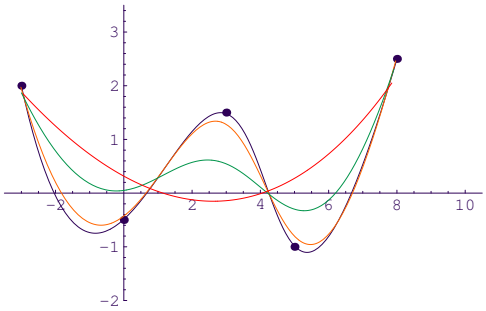
To find explicit solutions for $m \geq 3$ is not an easy task. However, using the software *Mathematica* 5.0, it is possible to illustrate our method. The cases $m = 3$ and $m = 4$ are presented in figure 1.



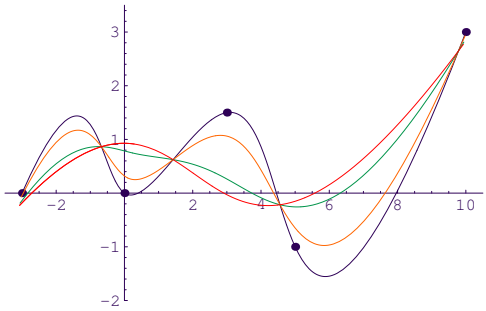
Smoothing splines for $m = 1$, $p_0 = (-5,-1)$, $p_1 = (-2,\frac{3}{2})$, $p_2 = (2,-\frac{3}{2})$, $p_3 = (5,2)$ and $p_4 = (8,\frac{1}{2})$, $t_0 = 0$, $t_1 = \frac{1}{4}$, $t_2 = \frac{1}{2}$, $t_3 = \frac{3}{4}$ and $t_4 = 1$. The smoothing splines were obtained for the following values of λ: $\lambda_1 = 10^{-3}$, $\lambda_2 = 10^{-1}$, $\lambda_3 = 1$ and $\lambda_4 = 4$.



Smoothing splines for $m = 2$, $p_0 = (-5,-1)$, $p_1 = (-2,\frac{3}{2})$, $p_2 = (2,-\frac{3}{2})$, $p_3 = (5,2)$ and $p_4 = (8,\frac{1}{2})$, $t_0 = 0$, $t_1 = \frac{1}{4}$, $t_2 = \frac{1}{2}$, $t_3 = \frac{3}{4}$ and $t_4 = 1$. The smoothing splines were obtained for the following values of λ: $\lambda_1 = 10^{-5}$, $\lambda_2 = 10^{-3}$, $\lambda_3 = 10^{-2}$ and $\lambda_4 = 10$.



Smoothing splines for $m = 3$, $p_0 = (-3,2)$, $p_1 = (0,-\frac{1}{2})$, $p_2 = (3,\frac{3}{2})$, $p_3 = (5,-1)$ and $p_4 = (8,\frac{5}{2})$, $t_0 = 0$, $t_1 = \frac{1}{4}$, $t_2 = \frac{1}{2}$, $t_3 = \frac{3}{4}$ and $t_4 = 1$. The smoothing splines were obtained for the following values of λ: $\lambda_1 = 10^{-7}$, $\lambda_2 = 10^{-6}$, $\lambda_3 = 10^{-5}$ and $\lambda_4 = 10^{3}$.



Smoothing splines for $m = 4$, $p_0 = (-3,0)$, $p_1 = (0,0)$, $p_2 = (3,\frac{3}{2})$, $p_3 = (5,-1)$ and $p_4 = (10,3)$, $t_0 = 0$, $t_1 = \frac{1}{4}$, $t_2 = \frac{1}{2}$, $t_3 = \frac{3}{4}$ and $t_4 = 1$. The smoothing splines were obtained for the following values of λ: $\lambda_1 = 10^{-10}$, $\lambda_2 = 10^{-7}$, $\lambda_3 = 10^{-6}$ and $\lambda_4 = 10^{7}$.

Figure 1: Solutions of the variational problem in $\mathbb{R}^2$ for the cases when $m \leq 4$.

What can be noticed in the above figures is that, for all of the proposed data, the solutions of the variational problem approach the corresponding solution of the classical least squares problem as long as the smoothing parameter λ goes to $+\infty$. This is the main reason why we believe that our approach is the natural way to generalize the classical least squares method to Riemannian manifolds.

References

[1] J. Baillieul. Kinematic Redundancy and the Control of Robots with Flexible Components. *IEEE International Conference on Robotics and Automation, Nice, France*, 1992.

[2] A. Bloch and P. Crouch. Nonholonomic and Vakonomic Control Systems on Riemannian Manifolds. *SIAM J. Control and Optimization*, 33(1):126–148, 1995.

[3] A. M. Bloch and P. Crouch. Nonholonomic Control Systems on Riemannian manifolds. *SIAM Journal on Control and Optimization*, 33(1):126–148, 1995.

[4] F. Bullo and M. Zefran. On Mechanical Systems with Nonholonomic Constraints and Symmetries. *Systems and Control Letters*, 42(1/2):135–164, 1998.

[5] M. Camarinha, F. Silva Leite, and P. Crouch. Splines of class C^k on Non-Euclidean Spaces. *IMA Journal of Mathematical Control and Information*, 12:399–410, 1995.

[6] P. Crouch and F. Silva Leite. Geometry and the Dynamic Interpolation Problem. *Proc. American Control Conference, Boston*, pages 1131–1137, 1991.

[7] P. Crouch and F. Silva Leite. The Dynamic Interpolation Problem: on Riemannian Manifolds, Lie Groups and Symmetric Spaces. *Journal of Dynamical and Control Systems*, 1(2):177–202, 1995.

[8] Q. J. Ge and B. Ravani. Geometric Construction of Bezier Motions. *ASME Journal of Mechanical Design*, 116:749–755, 1994.

[9] H. Karcher. Riemannian Center of Mass and Mollifier Smoothing. *Communications on Pure and Applied Mathematics*, 30:509–541, 1977.

[10] A. K. Krakowski. *Geometrical Methods of Inference*. PhD Thesis, Department of Mathematics and Statistics, The University of Western Australia, Australia, 2002.

[11] V. Kumar, M. Zefran, and J. Ostrowski. Motion Planning in Humans and Robots. *The Eighth International Symposium of Robotics Research, Hayama, Japan*, 1997.

[12] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting*. Academic Press, 1990.

[13] A. Lewis. Controllability of Simple Mechanical Control systems. *SIAM J. Control and Optimization*, 35(3):766–790, 1997.

[14] L. Machado. *Least Squares Problems on Riemannian Manifolds*. PhD Thesis, Department of Mathematics, University of Coimbra, Portugal, 2006.

[15] L. Machado and F. Silva Leite. Fitting Smooth Paths on Riemannian Manifolds. *International Journal of Applied Mathematics & Statistics*, 4(J06):25–53, 2006.

[16] L. Machado, F. Silva Leite, and K. Hüper. Riemannian Means as Solutions of Variational Problems. *LMS J. Comput. Math.*, (8):86–103, 2006.

[17] L. Noakes, G. Heinzinger, and B. Paden. Cubic Splines on Curved Spaces. *IMA Journal of Mathematics Control and Information*, 6:465–473, 1989.

[18] F. Park and B. Ravani. Bézier Curves on Riemannian Manifolds and Lie Groups with Kinematic Applications. *ASME Journal of Mechanical Design*, 117:36–40, 1995.

[19] M. Zefran, V. Kumar, and C. Croke. On the Generation of Smooth Three-dimensional Rigid Body Motions. *IEEE Trans. on Robotics and Automation*, 14(4):579–589, 1995.

Model predictive control of under-actuated mechanical systems

Amélia C. D. Caldeira* Fernando A. C. C. Fontes†

Abstract

Model Predictive Control (MPC) is an optimization-based control technique that has received an increasing research interest and has been widely applied in industry. Similarly to optimal control, MPC has an inherent ability to deal naturally with constraints on the inputs and on the state. Moreover, since the controls generated are closed-loop strategies obtained by optimizing some criterion, the method possesses some desirable performance properties [8] and also intrinsic robustness properties [7]. Thousands of applications have been reported [9], making MPC being classified as the only advanced control technique with a substantial impact on industrial control [6]. Until recently, a technical difficulty prevented the continuous-time MPC approaches to be used to stabilize important classes of nonlinear systems, such as under-actuated mechanical systems which frequently appear in robotics. Such class of nonlinear systems cannot be stabilized by a continuous time-invariant feedback. As a consequence, the trajectories cannot be described by classical (Caratheodory) solutions to differential equations.This difficulty is overcame in the framework proposed in [1] that describes how a continuous-time MPC framework using a positive inter-sampling time, combined with the use of an appropriate concept of solution to a differential equation, can address nonholonomic systems. An implementation of an MPC strategy to control a wheeled mobile robot using the results above is reported in [4, 5].

An MPC framework with guaranteed robustness properties addressing a general class of nonlinear systems is discussed in [2, 3]. Preliminary results on the implementation of a robust MPC strategy, devised in the above references, to a wheeled robot are reported. Conditions under which steering to a set is guaranteed are established.

Keywords: Model predictive control; stability analysis; under-actuated mechanical systems

*Departamento de Matemática, Instituto Superior de Engenharia do Porto, R. Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal. E-mail:acd@isep.ipp.pt

†Departamento de Matemática para a Ciência e Tecnologia, Universidade do Minho. Campo de Azurém, 4800-058 Guimarães, Portugal. E-mail:ffontes@mct.uminho.pt

1 Introduction

Under-actuated mechanical systems can model many interesting applications arising in robotics such as under-actuated manipulators, wheeled vehicles, etc.. Also, these systems typically exhibit a nonlinear and nonholonomic behavior and have fewer control inputs than degrees of freedom. So, the design stabilizing feedback controllers for these systems offers some interesting challenges. Namely, these systems are inherently nonlinear: they cannot be handled by any linear control method and are not transformable into linear systems (even locally) in any meaningful way. Furthermore, a (time-invariant) feedback law capable of stabilizing this class of systems must be allowed to be discontinuous. As a consequence, nonclassical definitions of a solution to a differential equation are needed to analyse the resulting trajectories. Despite these difficulties, a continuous-time Model Predictive Control (MPC) framework has been shown to be an appropriate methodology to generate stabilizing feedbacks for such systems. Model Predictive Control (MPC) is an increasingly popular control technique that has been developed both by the systems theory community where it is also known as Receding Horizon Control, and by the process engineering community where it is often referred to by commercial names such as Dynamic Matrix Control. This technique constructs a feedback law by solving on-line a sequence of open-loop optimal control problems, each of them using the currently measured state of the plant as its initial state. Similarly to optimal control, MPC has an inherent ability to deal naturally with constraints on the inputs and on the state. Since the controls are obtained by optimizing some criterion, the method possesses some desirable performance properties, and also intrinsic robustness properties [7]. These facts can partially explain the substantial impact it has made on industry: surveys carried out a decade ago with five MPC software vendors, identified more than 2200 industrial applications [9].

The framework proposed in [1] that describes how a continuous-time MPC framework using a positive inter-sampling time, combined with the use of an appropriate concept of solution to a differential equation, can address nonholonomic systems. An implementation of an MPC strategy to control a wheeled mobile robot using the results above is reported in [4, 5].

An MPC framework with guaranteed stability properties addressing a general class of nonlinear systems is discussed in [2, 3]. Preliminary results on the implementation of a stabilizing MPC strategy, devised in the above references, to a wheeled robot are reported. Conditions under which steering to a set is guaranteed are established.

2 Kinematic Model of the Robot

Consider the mobile robot of a unicycle type which is represented by the model:

$\left[\dot{x}, \dot{y}, \dot{\theta}\right] = \left[\frac{u_1+u_2}{2}\cos\theta, \frac{u_1+u_2}{2}\sin\theta, \frac{u_1-u_2}{L}\right]$, $u_1(t), u_2(t) \in [-u_{\max}, u_{\max}]$.

The Cartesian coordinates (x, y) are the position in the plane of the midpoint of the

axle connecting the rear wheels (L is the distance between the rear wheels) and θ denotes the heading angle measured anticlockwise from the x-axis. The controls u_1 and u_2 are the angular velocity of the right wheel and of the left wheel, respectively.

The velocity control of the two rear wheels determines the translation velocity of the robot: $v = \frac{u_1+u_2}{2}$ and the angular velocity: $w = \frac{u_1-u_2}{L}$. Suppose $L = 1$ without lost of generality. So, the model of the unicycle can be represented by the model: $\dot{\mathbf{x}} = [v\cos\theta, v\sin\theta, w]^T$ or, $\dot{x}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$, where $\mathbf{x} = [x \;\; y \;\; \theta]^T$ and $\mathbf{u} = [v \;\; w]^T$. Considering $u_{\max} = 1$, we have that $v \in [-1, 1]$ and $w \in [-1, 1]$

The system is subject to one nonholonomic constraint, the velocity vector is always orthogonal to the wheel axis, so the nonholonomic constraint $\left(\dot{x}, \dot{y}\right)^T (\sin\theta, -\cos\theta) = 0$. When trying to obtain a linearization of this system around any operating point $(\mathbf{x}_0; u_1 = 0; u_2 = 0)$ the resulting linear system is not controllable. Therefore, linear control methods, or any auxiliary procedure based on linearization, cannot be used to handle this system.

3 The MPC Formulation

Consider a nonlinear plant with input constraints, where the evolution of the state after time t is predicted by the model $\dot{x}(s) = f(\mathbf{x}(s), \mathbf{u}(s))$, for $s \in [T, +\infty)$ and $x(t) = x_t \in X_0$.

The data of this model comprise a set $X_0 \subset \mathbb{R}^n$ containing all possible initial states, a vector $\mathbf{x}_t \in X_0$ that is the state of the plant measured at time t, a given function $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and a multifunction $U : \mathbb{R} \to \mathbb{R}^m$ of possible sets of control values. These data combined with a particular measurable control function $\mathbf{u} : [t, +\infty[\to \mathbb{R}^m$ define an absolutely continuous trajectory $\mathbf{x} : [t, +\infty[\to \mathbb{R}^n$.

Suppose this system is uniformly asymptotically controllable on X_0.

Consider a sequence of sampling instants $\pi := \{t_i\}_{i\geq 0}$, with a constant inter-sampling time $\delta > 0$ such that $t_{i+1} = t_i + \delta$, $\forall i \geq 0$. Let the control horizon T_c and prediction horizon T_p, with $T_c \leq T_p$, be multiples of δ. Consider also a terminal set $S\,(\subset \mathbb{R}^n)$, a terminal cost function $W : \mathbb{R}^n \to \mathbb{R}$, and a running cost function $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$.

The objective is to drive this system to the origin. In practice we consider target set
$\Theta = \left\{(x, y, \theta) \in \mathbb{R}^2 \times [-\pi, \pi] : \|(x, y)\| \leq \varepsilon_1, |\theta| \leq \varepsilon_2\right\}$ where $\varepsilon_1, \varepsilon_2 > 0$.

The feedback control is obtained by repeatedly solving online open-loop optimal control problems (OCP) at each sampling instant t_i, every time using the current measure of the state of the plant $\mathbf{x}_{t_i}$.

The MPC control law is obtained with the algorithm described now:

1. Measure state of the plant x_{t_i};

2. Get $\overline{u} : [t_i, t_i + T] \rightarrow \mathbb{R}^n$ solution of the OCP:

$$\begin{array}{ll} \text{Minimize} & \int_{t_i}^{t_i+T} L\left(\mathbf{x}(t), \mathbf{u}(t)\right) dt + W\left(x\left(t_i + T\right)\right) \\ \text{subject to:} & \dot{x} = f\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right), \text{ a.e. } t \in [t_i, t_i + T] \\ & x\left(t_i\right) = x_{t_i} \\ & u\left(t\right) \in U\left(t\right), \text{ a.e. } t \in [t_i, t_i + T] \\ & x\left(t_i + T\right) \in S \end{array} \tag{1}$$

3. Apply to the plant the control $u^*\left(t\right) := \overline{u}\left(t\right)$ in the interval $[t_i, t_i + \delta[$. The remaining control is $\left(\overline{u}\left(t\right), t > t_i + \delta\right)$ discarded;

4. Repeat the procedure from (1.) for the next sampling instant: $t_i = t_i + \delta$, using the new measure of the state of the plant x_{t_i+1}.

The pair $(\overline{\mathbf{x}}, \overline{\mathbf{u}})$ denotes an optimal solution to an open-loop OCP. The process $(\mathbf{x}^*, \mathbf{u}^*)$ is the closed-loop trajectory and control resulting from the MPC strategy. The *design parameters* (the variables present in the open-loop OCP that are not from the system model) are: the time horizon T, the functions L and W, and the set S. They must satisfies the stability conditions (SC) and therefore guarantee that the resulting MPC strategy is stabilizing, and are constructed based on a simple strategy that drives the system to the origin.

Consider the following stability condition SC[1]:

Theorem: Choose the design parameters satisfying:

SC1 The set S is closed and contains the origin.

SC2 The function L is continuous, $L(., 0, 0) = 0$, and there is a continuous positive definite and radially unbounded function $M : \mathbb{R}^n \rightarrow \mathbb{R}_+$ such that $L\left(t, \mathbf{x}, \mathbf{u}\right) \geq M\left(\mathbf{x}\right)$ for all $(t, \mathbf{u}) \in \mathbb{R}_+ \times \mathbb{R}^m$. Moreover, the "extended velocity set" $\{(v, l) \in \mathbb{R}^n \times \mathbb{R}_+ : v = f\left(t, \mathbf{x}, \mathbf{u}\right), l \geq L\left(t, \mathbf{x}, \mathbf{u}\right), \mathbf{u} \in U\left(t\right)\}$ is convex for all $(t, \mathbf{x})$.

SC3 The function W is positive semi-definite and continuously differentiable.

SC4 The time horizon T is such that, the set S is reachable in time T from any initial state and from any point in the generated trajectories: that is, there exists a set $X \supset X_0$ such that for each pair $(t_0, \mathbf{x}_0) \in \mathbb{R}_+ \times X$ there exists a control $u : [t_0, t_0 + T] \rightarrow \mathbb{R}^m$, with $u\left(s\right) \in U\left(s\right)$ for all $s \in [t_0, t_0 + T]$, satisfying $x\left(t_0 + T; t_0, \mathbf{x}_0, \mathbf{u}\right) \in S$. Also, for all control functions u in the conditions above $x\left(t; t_0, \mathbf{x}_0, \mathbf{u}\right) \in X$ for all $t \in [t_0, t_0 + T]$.

SC5 For every time $t \in [T, \infty[$ and each $x_t \in S$, we can choose a control function $\widetilde{\mathbf{u}}$ continuous from the right at t, satisfying:

$W_t\left(t, \mathbf{x}_t\right) + W_x\left(t, \mathbf{x}_t\right) \cdot f\left(t, \mathbf{x}_t, \widetilde{\mathbf{u}}\left(\left(t\right)\right)\right) \leq -L\left(t, \mathbf{x}_t, \widetilde{\mathbf{u}}\left(\left(t\right)\right)\right)$ **(SC5a)**

$x\left(t + \delta; t, \mathbf{x}_t, \widetilde{\mathbf{u}}\right) \in S$ **(SC5b)**.

Then, for a sufficiently small inter-sample time δ, the closed-loop system resulting from the application of the MPC strategy is asymptotically stable.

Consider: $\Theta = \left\{\mathbf{x} = (x, y, \theta) \in \mathbb{R}^2 \times [-\pi, \pi] : \|(x, y)\| \leq 0.05, |\theta| \leq 0.1\right\}$. It is convenient to define $\phi(x, y)$ to be the angle that points to the origin from position (x, y) away from the

origin, more precisely: $\phi(x,y)=0$ if $\|(x,y)\|\leq 0.05$. Otherwise, $\phi(x,y)=-(\pi/2)\text{sign}(y)$ if $x=0, y\neq 0$, $\phi(x,y)=\tan^{-1}(y/x)+\pi$ if $x>0$, or $\phi(x,y)=\tan^{-1}(y/x)$ if $x<0$.

A possible auxiliary stabilizing strategy $\widetilde{u}$ is: if $\|(x,y)\|<0.05$ then $v=0$ and $w=-\frac{\pi}{4}\times\theta$. Otherwise $v=0.1$ and $w=-\frac{\pi}{4}\times[\theta-\phi(x,y)]$.

Define the terminal set S to be the set of states heading towards the origin of the plane together with the origin of the plane, that is:

$S=\left\{\mathbf{x}=(x,y,\theta)\in\mathbb{R}^2\times[-\pi,\pi]:(x,y,\theta)\in\Theta\vee(x,y)=(0,0)\right\}$.

This set can be reached for any state if we define the horizon to be the time to complete an 180 degrees turn, that is $T=\pi/w_{\max}=\pi$.

Define $L(\mathbf{x})=x^2+y^2+\theta^2$ and $W(\mathbf{x})=2\int_0^{\overline{t}}L(x(t),y(t),\theta(t))\,dt$, using the auxiliary strategy $\widetilde{u}$ defined in [2] and where $\overline{t}$ is the time to reach the origin. An explicit formula is: $W(\mathbf{x})=\frac{1}{3}\left(r^3+|\theta|^3\right)+r\theta^3$, where $r=\sqrt{x^2+y^2}$ [2].

It is an easy task to check that these design parameters satisfy conditions **SC1** to **SC4** and **SC5b**. We verify **SC5a** below.

$\nabla W(\mathbf{x})=\left(xr+2x\theta^2/r, yr+2y\theta^2/r, \theta|\theta|+2\theta r\right)$ when $\theta\neq 0$ and $r\neq 0$. Suppose $\|(x,y)\|\leq 0,05$ and if $|\theta|\leq 0.1$ then we are already in Θ. Consider now the case when $|\theta|>0.1$. Since $r\leq 0.05\leq 0.1$ then $|\theta|>r$. The controls are chosen as $v=0$ and $w=-\frac{\pi}{4}\times\theta$, suppose $r\neq 0$ therefore:

$\nabla W(\mathbf{x})\cdot f(\mathbf{x},\mathbf{u}) \quad =-\frac{\pi}{4}\times\theta(\theta|\theta|+2\theta r) \quad \leq -L(x,y,\theta)$

Suppose $\|(x,y)\|>0.05$. The controls are chosen as $v=0.1$ and $w=-\frac{\pi}{4}\times[\theta-\phi(x,y)]$. Let $a=\theta-\phi(x,y)$ (note that $a\leq\frac{\pi}{4}$). Then

$f(\mathbf{x},\mathbf{u})=\left(\frac{\cos\phi\cos a-\sin\phi\sin a}{10}, \frac{\sin\phi\cos a+\cos\phi\sin a}{10}, -\frac{\pi}{4}\left[\theta+\pi-\tan^{-1}\left(\frac{y}{x}\right)\right]\right)$.

It is easy to prove, in the case $x>0$, that $\cos\phi=\cos\left[\tan^{-1}\left(\frac{y}{x}\right)+\pi\right]=-\frac{x}{\sqrt{x^2+y^2}}$, and $\sin\phi=\sin\left[\tan^{-1}\left(\frac{y}{x}\right)+\pi\right]=-\frac{y}{\sqrt{x^2+y^2}}$.

If $\theta\neq 0$ the gradient of W is well defined and (note that $2r\theta\leq r^2+\theta^2$):

$\nabla W(\mathbf{x})\cdot f(\mathbf{x},\mathbf{u})=-\frac{r^2+2\theta^2}{10}\cos a-\frac{\pi}{4}(a+2\pi)(\theta|\theta|+2\theta r)\leq -L(x,y,\theta)$

The cases when $x<0$ and $x=0$ can be verified in a similar way.

It follows from our main stability result that this choice of design parameters guarantees the stability of the closed-loop trajectory.

4 Implementation and simulation

The optimal control 1 (in Bolza form) is reformulated in an equivalent Mayer form and discretized. The resulting nonlinear programmes are then solved in MATLAB using as initial guess the auxiliary stabilizing strategy. Simulation results are shown both using the auxiliary stabilizing strategy as well as the MPC strategy for the initial position $\mathbf{x}_0=\left(0,1,\frac{\pi}{2}\right)^T$.

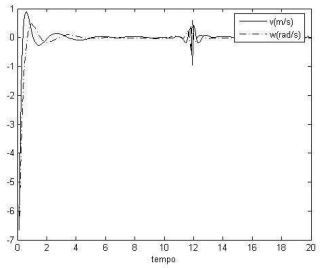


Fig.1: MPC Trajectory

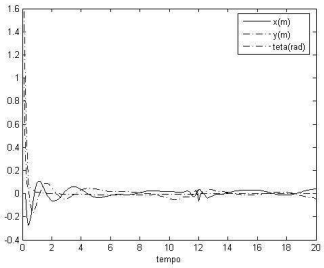


Fig.2: MPC Control

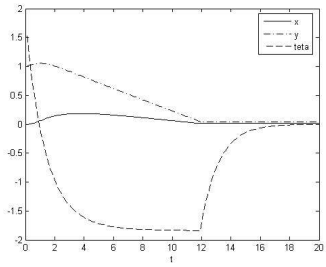


Fig.3: Trajectory using $\tilde{u}$

5 Conclusions

In this work we report preliminary results on the implementation of a stabilizing MPC strategy to a wheeled robot. Conditions under which steering to a set is guaranteed are established. A set of design parameters satisfying all these conditions for the control of a unicycle mobile robot are derived. How to improve the computation efficiency is currently under investigation.

References

[1] F. A. C. C. Fontes. Discontinuous feedbacks, discontinuous optimal controls, and continuous-time model predictive control. International Journal of Robust and Nonlinear Control, 13(3-4):191-209, 2003.

[2] F. A. C. C. Fontes and L. Magni. Min-max model predictive control of nonlinear systems using discontinuous feedbacks. IEEE Transactions on Automatic Control, 48:1750-1755, 2003.

[3] F. A. C. C. Fontes, L. Magni and E. Gyurkovics. Sampled-data model predictive control for nonlinear time-varying systems: Stability and robustness. In Allgower, Findeisen, and Biegler, editors, Assessment and Future Directions of NMPC, volume 358 of Lecture Notes in Control and Information Systems, pages 115-129. Springer Verlag, 2007.

[4] D. Gu, H. Hu. A stabilizing receding horizon regulator for non-holonomic mobile robots. IEEE Transactions on Robotics, 21(5):1022-1028, 2005.

[5] D. Gu, H. Hu. Receding horizon tracking control of wheeled mobile robots. IEEE Transations on Control Systems Techonology, 14(4):743-749, 2006.

[6] J. M. Maciejowski. Predictive Control with Constrains. Prentice Hall, Harlow, England, 2001.

[7] L. Magni and R. Sepulchre. Stability margins of nonlinear receding horizon control via inverse optimality. Systems and Control Letters, 32:241-245, 1997.

[8] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. Automatica, 36:789-814, 2000.

[9] S. J. Qin and T. A. Badgwell. An overview of nonlinear model predictive control aplications. In NMPCWorkshop, Ascona, Switzerland, 1998.

Perceptual grouping of edges and corners: grammatical inference for implicit object reconstruction from 2-manifold 3D-data

Peter Michael Goebel* Markus Vincze†

Abstract

The work of this paper shows how perceptual grouping can be applied within a recently presented framework to support learning of object reconstruction recipes by grammatical inference. Furthermore, the ability to collect information of views from different viewpoints enables us to deal with implicit object topology. We use the perceptual grouping of 3D point-cloud image points into planes by probabilistic robust fitting and the segmentation of edges and corners by intersecting the planes. The edge and corner primitives found by processing of Monte-Carlo simulations and real 3D point-cloud images, are used to train a N-gram model that build object prototypes by Bayesian belief networks. We use perplexity to find out the best performing belief network under candidates. The modeling approach can be utilized for the representation and recognition of object types with occlusions in cluttered, and noisy environments.

1 Introduction

Psychological findings of mammalian visual perception generate supports for cognitive vision models in machine-vision in that they assume the most regular and simplest organization that is equivalent with an image of a given setup [21]. Hence, Gestalt theory appears to be the root of perceptual grouping (PG) and aims at the detection of structure and regularities over several stimuli by extraction of groups of image features that occurred non-accidental, where the grouping principles are based on similarity, proximity, common fate, collinearity, good continuation and past experience [13].

Thus, a one-to-one mapping is found, which maps geometric primitives to sets of numerical parameters into a subset of $\mathbb{R}^n$, where n is the dimension of the representation [4].

*Vision for Robotics Lab., Automation and Control Institute ACIN, Vienna University of Technology. E-mail: gp@acin.tuwien.ac.at. Supported by project S9101 "Cognitive Vision" of the Austrian Science Foundation.

†Vision for Robotics Lab., ACIN, Vienna University of Technology.E-mail:vincze@acin.tuwien.ac.at.

Since identification of objects within a familiar scene is influenced by the observer's location, reconstruction can be supported by contextual information about where we are in space and in which direction we are looking [3]. Although projective views from objects possess some invariance to viewpoint change [2], they appear very different when the change gets above a certain limit. A general mathematical definition is needed in order to generate appropriate and complete object representations, since it appears rather impossible to collect all necessary information for a reconstruction of object-topology of all possible object-views in advance. In our approach, we find this definition in the notion of manifolds, where the key idea is that all appearing object-views are projections from the same original object and therefore smoothly interrelated.

Definition 1.1. *When, as shown as in Figure 1 – X is a set of points in $\mathbb{R}^n$, $(U_i \subset X)$ an open set, φ_i maps $\{\varphi_i := U_i \cap X \mapsto \mathbb{R}^d \mid d < n\}$ of projective representations of the set X – and Iff $\{U_i \cap U_{j \neq i} \neq \emptyset\}$ and $\left(\varphi_i \cdot \varphi_j^{-1}\right), \left(\varphi_j \cdot \varphi_i^{-1}\right)$ are two mappings $\mathbb{R}^d \mapsto \mathbb{R}^d$ infinitely differentiable – then the set X together with the open sets U_i and the maps φ_i is called a C^∞ manifold of dimension d (adapted from [4]).*

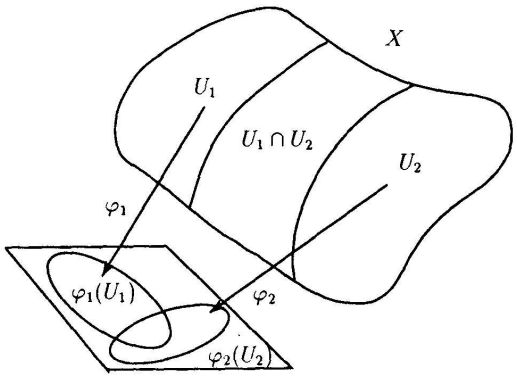


Figure 1: A 2-Manifold: element $x \in X$ is represented by a pair $(\mathbf{p}, i)$, where $\mathbf{p} \in \mathbb{R}^d$ and i is the index, φ_i such that $\mathbf{p} = \varphi_i(x)$, then the set of pairs (U_i, φ_i) is called an *atlas* of set X[4].

However, object reconstruction methods that are state-of-the-art commonly utilize a priori knowledge for each object that should be recognized. To be of practical interest in a real world sense a method must also be able to deal with objects for which it has no explicit prior model ready. In recent work we proposed a cognitive framework [8] with the implementation of an vision module based on mammalian psychophysical findings by using predefined object recipes for object reconstruction [6].

The work of this paper shows how perceptual grouping can be applied within the framework to support learning of object reconstruction recipes by grammatical inference. We use the PG of 3D point-cloud image points into planes by probabilistic robust fitting with RANSAC [5] and the segmentation of edges and corners by intersecting the planes. The edge

and corner primitives found by processing of Monte-Carlo simulations and real 3D point-cloud images, are used to train a N-gram model that build object prototypes by Bayesian belief networks. We use perplexity to find out the best performing recipe under candidates.

The paper is organized as follow: Section 2 shows related work; Section 3 shows how object detection is related to representation; followed by Section 4 with presenting a N-gram model as the learning corpus; Section 5 concludes with an outlook to future work.

2 Related work

There exist a vast literature on PG in vision, see e.g. [19] for a survey. Early work in PG dates back to Marr [16], who was first who suggested to incorporate grouping based on curvilinearity into larger structures by his primal sketch approach; Witkin and Tenenbaum [22] postulated non-accidentalness for spatiotemporal coherence; Lowe [15] derived an expectation estimate for accidental occurrences by assuming an uniform distribution to line segments; Sarkar and Boyer [18] developed a Bayesian network method for geometric knowledge-based representation; Zucker [24] introduced closure as more global feature to better deal with occlusions; and Ackermann et al [1] introduced a Markov random field grouping approach with learning from hand-labeled trainings sets; however, this list is still far from completeness.

More recent work of Procter [17] investigated grouping of edge-triple features, to recognize polyhedrons from 2D image projections, however, the method turned out to be too sensitive to noise and thus failed practical demonstration.

Levinshtein et al [14] proposed recovering a Marr [16] like abstraction hierarchy from a set of examples by applying a multi-scale blob and ridge detector for feature extraction, here draw backs arise from the fact that positional information of the blobs is lost during a graph embedding and that a vast number of parameters are to be defined.

Zillich [23] aimed at issues concerning complexity and robustness by proposing an incremental processing scheme for the PG of edges in indoor scenes. He proposed using Gestalt principles to support PG and implemented successfully a Markov random field approach in order to deal with real-world objects. Although, in abstract sense, the approach relates very close to our approach, his self-criticism is, he unfortunately highly assumed the getting of clean edges by local edge detectors, which degraded system performance in complexer setups. The main difference to our approach is that we focus on robustness by applying multiresolution methods [8] and to use more global approaches rather than localized ones. We intend to get first only a coarse representation of an object at hand and refine the representation afterwards when more information is required.

3 From detection to representations

Objects, seen on a very abstract level, can be represented by graphs. These graphs are representations of the connections between structural elements, such as *(i) corners*; *(ii) edges*; and *(iii) boundaries*, where two areas meet: in a fold[1], or in a blade[2], or a face[3]. Structural elements and their connections are defined by relations between image primitives. Image primitives are composed by groupings of image features that are extracted from image points. Thus, the more abstract process is that the structural elements, which constitute an object, are searched in a reduced search space by defining some relations between them. To get ready for corner classification, one has first to segment edges within the image.

3.1 Structural segmentation in 3D

Image segmentation is commonly the partitioning of an image into regions that separate different areas from each other and also from its background. Because of lack of space, we refer [9] for a comprehensive review of low-level segmentation methods[4].

In our present approach we use structural properties such as cutting-edges between intersecting planes that are detected in the 3D-point-cloud images by using a robust plane grouping method applied in [20]. The method uses the Random Sample Consensus Algorithm (RANSAC) [5], which is a probabilistic algorithm for robust fitting of models in the presence of many data outliers and noise. The plane fitting algorithm randomly selects points from the 3D-point-cloud image and groups triples together in order to define triangles. Only triples that form triangles that lie within a cube's face are grouped into planes, where a plane is then defined by its vector to the origin $\underline{\mathbf{c}}$ (center) and its normal vector $\underline{\mathbf{n}}$ that is perpendicular to the plane[5].

3.2 Symbolic corner labeling

Depending on the viewpoint, corners may appear very differently. In our approach, we classify corners into four junction types, such as: *(L)-type* which means an occluding line and denotes a blade that is an object region in front of the background; *(Y)-type* means a 3-junction, where three surfaces intersect with the angles between each pair are < 180 degree; *(T)-type* means a 3-junction with one of the angles has exactly 180 degrees; and *(W)-type* means a 3-junction with one angle > 180 degrees.

[1] when both areas are from the same object

[2] when one area is from the object and one from the background

[3] when one area appears closed

[4] Their performance compared to human observers is low, especially when applied in noisy environments.

[5] Note the method appears akin to a 3D Hough transform.

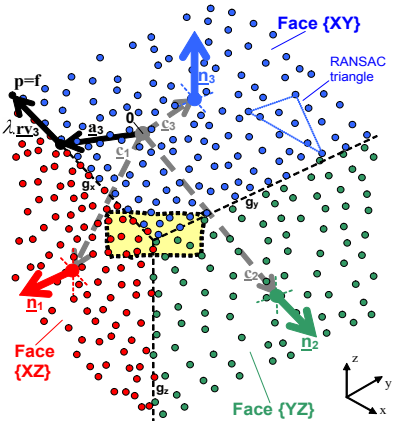


Figure 2: shows the grouping of points from the point-cloud into planes by RANSAC triangle fitting method. Hence, every face of the cube gets defined its own normal vector $\underline{\mathbf{n}}$, sitting at the end of the center vector $\underline{\mathbf{c}}$ from the origin O. The intersection of two planes results in a crossing edge g with the space point $\underline{\mathbf{a}}$. Herein a point $\mathbf{p}$, already on line g, (i.e. $\mathbf{p}$ equals the base point $\mathbf{f}$ and therefore $d = 0$) is connected by $\lambda \cdot \underline{\mathbf{rv}}$ with the endpoint of space point vector $\underline{\mathbf{a}}$.

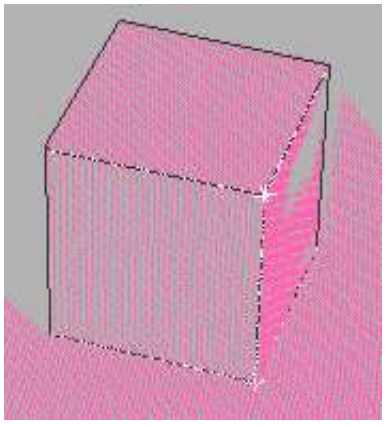


Figure 3: a real object's 3D point-cloud with the segmentation result, where white marks at the folds represent the final proximity points at the crossing edges. The resulting edge lines are shown in black; the black blade lines are segmented at the border between the point-cloud face and the object shadow by local segmentation methods. The white crosses depict the location of the corners of the cube, where the 2 faces and the ground plane cross.

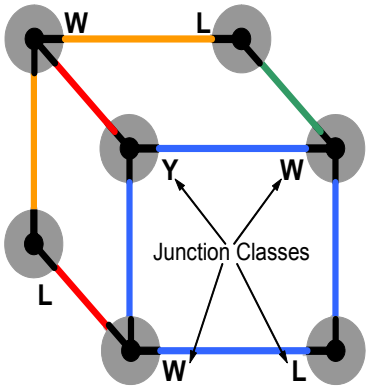


Figure 4: the junction-types for corner labeling of an example cube, as used in our approach: *(L)* means an occluding boarder and denotes a *blade* where two surfaces meet and just one of them is visible; *(Y)* means a 3-junction, where three surfaces intersect with the angles between each pair are less than 180 degree; and *(W)* means a 3-junction with one angle greater than 180 degree. Note: no *(T)*-junction type is present here.

4 The graphical model and grammatical inference

There exist two principal methodologies in modeling: *(i) discriminative*, versus *(ii) generative* approaches. *Graphical models* are generative approaches, since they may generate synthetic data by sampling from a distribution, define prior probabilities or complete probability distributions. Such models appear commonly too complex for discriminative[6] direct estimation of posterior probabilities. Thus, the distributions must be factorized into manageable parts that can be realized by: *(i) naive Bayes* - assuming strongly naive independence between random variables; *(ii) N-grams* - which model symbol sequences, using statistical properties; *(iii) hidden Markov models* - that are Markov processes with unobservable parameters, which

[6]e.g. by Support Vector Machines, traditional neural networks, or conditional random fields

makes its challenge by determining hidden parameters from the observable ones; and *(iv) probabilistic context free grammars* - that are context-free grammars, in which each production is augmented with a probability. Hence, a formal grammar is defined by the quadruple $G := (N, \Sigma, P, S)$; where (N) is a finite set of nonterminal (syntactical) symbols; (Σ) a finite set of terminal symbols; (P) a finite set of production rules of the form $A \rightarrow BC$, or $A \rightarrow \omega$, where $A, B, C \in N$ and $\omega \in \Sigma$; and (S) the start symbol. A grammar is context-free if $(N \cap \Sigma) = \{\}$ i.e. N, Σ are disjoint sets, and if $(\exists\ string \in (N \cup \Sigma)^* : (S, string) \in P)$; a regular grammar is a finite state automaton which is context-free; regular grammars produce regular languages. *Grammatical inference* (GI) aims in learning of regular language from examples, i.e. by learning the structure of a finite state automaton and by estimating its transition probabilities [10].

4.1 The statistical language N-gram model

Statistical language models are used for the modeling of sequences of symbols under the assumption that the underlying generation process is an approximate Markov-chain process, where the *approximate Markovian property* of an order n process is that the conditional probability of future states depends only upon the past $n-1$ states, what means that the process is conditionally independent of the $> n-1$ past states. Thus, a statistical language model, in general, defines a probability distribution over the set of symbols sampled from a finite alphabet. Its representation by a Markov chain appears as a simple, but high performing concept [11]. The maximal context symbol length is also called *the symbols history*, where lengths with $n = 1$ are Uni-gram, $n = 2$ are Bi-gram, and $n = 3$ are Tri-gram models.

Definition 4.1. *A N-gram-Model is a Markov-chain model, giving the probability definition for non-terminating symbols with a maximal context length of $n-1$ predecessors by Bayes' chain rule*

$$p(\omega) \approx \prod_{t=1}^{T} p(\omega_t | \underbrace{\omega_{t-n+1}, \ldots, \omega_{t-1}}_{n \text{ symbols history}}) \tag{1}$$

A certain n-tuple of symbols is called a N-gram and is denoted $n-G := \boldsymbol{y}z$ where z is the predicted symbol and $\boldsymbol{y} = [y_1, y_2, \ldots, y_{n-1}]$ the context length [11].

When one is able to make sure that the training data contain all objects' data that should be learned, N-gram models have the advantage over hidden Markov language models to allow the calculation of optimum parameters directly from training data. In our present approach, the naive straight ahead solution to the learning problem is to count the absolute frequency $c(\omega_1, \ldots, \omega_N)$ of all symbol tuples and all possible contexts $\omega_1, \ldots, \omega_N$ to define all conditional probabilities by relative frequencies.

In order to get a satisfying set of sample data, we have tested trigram, bigram and unigram modeling by data from real cube-point-clouds with different viewpoints and also MC 2-

manifold projection simulations of polyhedron objects with 4/4, 8/6, and 20/12 vertices/faces that are tetrahedron, hexahedron, and dodecahedron views, respectively. Figure 5-left, shows a sequence of projections a)...d) of a polyhedron with $n = 8$ vertices, a cube, where the data was generated by MC simulations; at the middle, the respective planar graphs are given, as described in [6]; and at the right, derived Bayes' networks are shown that will be discussed in Section 4.2. In Table 1, four unigram transition matrices of MC-samples of Figure 5 are given, showing the observed counts of junction-type to junction-type transitions that are cumulated into training counts for $N_{i,j} \mid i,j : \{L,W,Y,T\}$, as shown at the left of Table 2.

Table 1: Unigram counts of junction to junction dependencies observed from different viewpoints of the Monte-Carlo simulation according to , rows a)...d).

Junction to junction transition counts, observed from																
	Viewpoint a)				Viewpoint b)				Viewpoint c)				Viewpoint d)			
	L	W	Y	T	L	W	Y	T	L	W	Y	T	L	W	Y	T
L	0	6	0	0	2	0	0	4	0	6	0	0	2	0	0	4
W	6	0	3	0	0	0	0	0	6	0	3	0	0	0	0	0
Y	0	3	0	0	0	0	0	0	0	3	0	0	0	0	0	0
T	0	0	0	0	4	0	0	1	0	0	0	0	4	0	0	1

A problem is that the probability for *unseen events* is per definition zero as one can see in the matrices of Table 1. Hence, in the case of presenting unseen events to the model, the N-gram model runs into *empirical holes* or singularities of its distribution. Therefore, a post-processing step for *smoothing* [11] in order to overcome the problem is indicated. The simplest one would be to apply the *Adding-one*[7] method to all elements in the matrices. Hence, this would overestimate the probability of the unseen events, since we are gaining only few counts per sample. Therefore, we apply smoothing that better deals with low counts by a modified[8] *Good-Turing discounting* (1953) method before normalizing into probabilities, we estimate the probability for N-grams with zero counts and occurrence $N_{C=0}$ by looking on the number of N-grams that occurred with a global minimum count $N_{C_{min}>0}$ and calculate counts $N_{unseen} = N_{C_{min}>0}/N_{C=0}$. The smoothed counts yield $\omega_{i,j} = (\forall N_{i,j} = 0 : N_{unseen}) \cup N_{i,j}$, and therefore, the transition probabilities τ for the full unigram transition matrix yields

$$\tau_{(1)} = \hat{p}(\omega_{i,j}) = \omega_{i,j} / \sum_{k=1}^{4} \omega_{i,k} \mid i,j = \{1 \ldots 4\} \tag{2}$$

Similarly to the calculation of the (4 by 4) unigram transition matrix $\tau_{(1)}$, both, a (4 by 4 by 4) bigram transition matrix $\tau_{(2)}$, defining twofold[9] junction/junction-type conditional probabilities, and a (4 by 4 by 4 by 4) trigram transition matrix $\tau_{(3)}$, defining threefold[10]

[7] also referred to as *Laplace's Law*

[8] The modification is to use the minimum count not equal zero rather than a count equal 1 [11]

[9] i.e. $\{p(L|LL), p(L|LW), \ldots, p(L|LT), p(L|WL), \ldots, p(L|TT), p(W|LL), \ldots, \ldots, \ldots, p(T|TT)\}$

[10] i.e. $\{p(L|LLL), p(L|LLW), \ldots, p(L|LLT), p(L|LWL), \ldots, \ldots, \ldots, p(T|TTT)\}$

Table 2: Unigram Left: Training counts $N_{i,j}$ according to Monte-Carlo samples of Figure 5; Middle: modified Good-Turing smoothed counts $\omega_{i,j}$; Right: transition probabilities $\tau_{(1)}$.

Training Counts $N_{i,j}$					Smoothed Counts $\omega_{i,j}$					Transition Probabilities $\tau_{(1)} = \hat{p}(\omega_{i,j})$				
	L	W	Y	T		L	W	Y	T		L	W	Y	T
L	4	12	0	8	L	4	12	0.125	8	L	0.166	0.497	0.005	0.332
W	12	0	6	0	W	12	0.125	6	0.125	W	0.658	0.007	0.329	0.007
Y	0	6	0	0	Y	0.125	6	0.125	0.125	Y	0.020	0.941	0.020	0.020
T	8	0	0	2	T	8	0.125	0.125	2	T	0.780	0.012	0.012	0.195

junction/junction-type conditional probabilities, are calculated.

However, in order to model object prototypes by data of the transition matrices, we defined to have recipes of object prototypes realized by a probabilistic finite state automaton, defined according to [12, 6] $\{Q, \Sigma, \delta, \tau, S_0, F, \varphi\}$, with Q as a finite set of states, Σ the Alphabet, $\delta : Q \times \Sigma \mapsto Q$ the transition function, $\tau : Q \times \Sigma \mapsto]0, 1]$ the transition probabilities, S_0 the initial state, $F \subset Q$ is a subset of final states from the set Q and $\varphi : Q \times \Sigma \mapsto]0, 1]$ the probability for a state to be final. In Section 4.2, we define the states of the automaton of an object at hand, by Bayes' belief networks with the probabilities calculated so far.

4.2 Belief networks

Belief networks model firstly the independence relationships between groups of random variables and secondly reflect their topology graphically in a directed acyclic graph (DAG). The edges of the DAG show the conditioning variables in their expansions and represent the recipes for object construction. As in our approach the network starts in S_0 at an arbitrary junction, and the DAG gets assigned directions only in order to satisfy Bayes chain rule, it may end in an also arbitrary final state $F_{Terminate}$. Thus, it gets possible to remodel the belief network for optimization purposes.

In Figure 5, four results of MC simulations of a cube are given: in the first and second column, the views and their plane graphs are shown; the belief nets with coloring the nodes by its costs[11] are shown in column three. When it turns out that there is a trigram used by the network, we use perplexity to test if we can minimize the order of the N-grams by recreating the network with changing link directions and preserving joint probability (see Figure 5-column four). Perplexity is a related measure of the uncertainty of a language event.

Definition 4.2. *The perplexity of a language model is the reciprocal of the geometric average of the symbol probabilities of a test set* $\Omega = \omega_1, \omega_2, \ldots, \omega_N)$ *of the predictions [11]:*

$$PP(\Omega) = \left[\prod_{i=1}^{|\Omega|} p\left(\omega_i | \omega_1 \ldots \omega_{i-1}\right) \right]^{-\frac{1}{|\Omega|}}. \tag{3}$$

[11]The computationally costs are increasing from using unigrams, to bigrams and trigrams for conditioning.

Thus, the higher the conditional probability of the symbol sequence, the lower the perplexity, and therefore, minimizing the perplexity is the optimization criteria used.

4.3 Training of the model

For training the model, we split given data in three disjoint sets: *(i) the training set* $\mathbb{T}$, used for stepwise learning; *(ii) the validation set* $\mathbb{V}$, used to verify an order change of the model; and *(iii) the test set* $\mathbb{A}$, used to assess the performance of the model.

Hence, in every MC *training step*, we firstly select a training object t randomly from the training set $\mathbb{T}$. Secondly, we repeat for $i = 1 \ldots N$ times a random selection of viewpoint positions $P_{P(x,y,z)}(t)$ around each test object t, and calculate transition counts for unigrams, bigrams, and trigrams of the junction to junction connectivity from the set of junctions $\mathbb{J} = \langle W, L, Y, T \rangle$ as defined as in Section 4.1. Thirdly, we apply smoothing and calculate the transition probability matrices $\tau_{(1)}$, $\tau_{(2)}$, and $\tau_{(3)}$ that are used to define the belief network of the new recipe candidate r, representing the conditional probabilities of the dependencies between all junctions of the object given. Finally, it is checked if a variant recipe can be found that provides the same or lower perplexity with using lower ordered N-grams, which is then selected for replacement of the recipe at hand. This new order N-gram recipe candidate is verified with the validation set $\mathbb{V}$ by the *verification step* in order to preserve performance, and the selection result is stored as a new recipe r to the set R of known recipes.

The *inference step* is designated to find the best recipe $r \in R$ that fits to a given observation O. We find a solution by calculating the likelihood $p(O, r)$ and classifying the observation O into a class that maximizes the posterior probability

$$p(r^*, O) = max_i \ \frac{p(O, r_i)p(r_i)}{p(O)} \tag{4}$$

Since $p(O)$ is independent from r, we only have to consider the nominator of Equ.4 to find the optimum

$$r^* = argmax_r \ p(r|O) = argmax_r \ p(O|r)p(r) \tag{5}$$

Despite to common use of dropping $p(r)$, we though use $p(r)$ to ensure high selectivity in cases where the observation O is only partly given. The inference step is validated by the test set $\mathbb{A}$ within an confidence interval of 95%.

5 Conclusions and future work

In this work, we have shown segmentation of structural information by perceptual grouping; we defined a N-gram graphical model and used belief networks to model object reconstruction recipes with Monte-Carlo simulation training and real data. With further developing the proposed approach of this work and together with results form previous work [8, 7, 6], learning

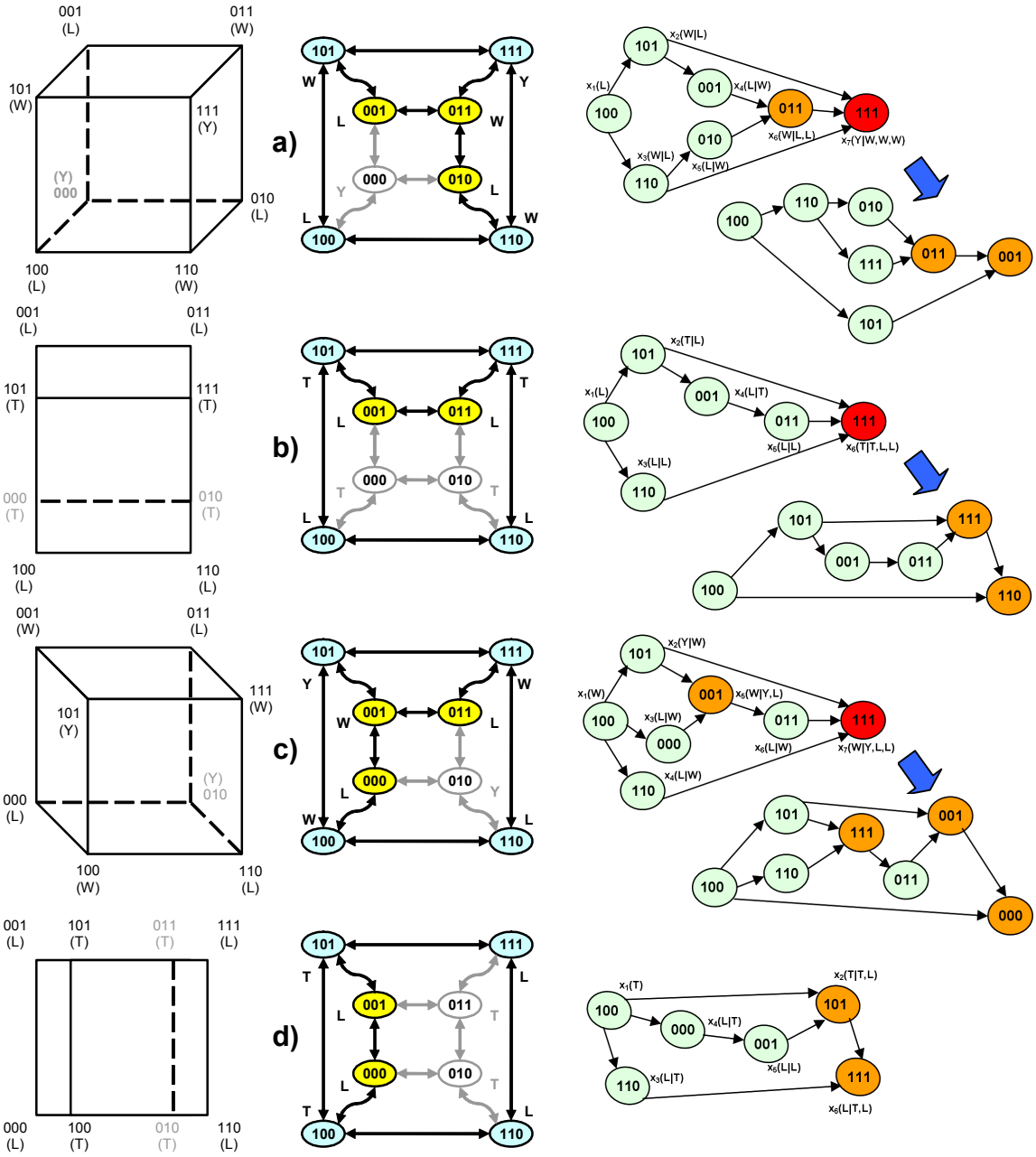


Figure 5: A viewpoint sequence a)..d), taken out of the Monte-Carlo training simulation. From left to right: firstly, the projective views are given; secondly, the planar graphs; thirdly, the Bayes' belief networks, showing trigram-realizations and their bigram replacements (i.e. up to three conditional variables are possible, when using trigrams, shown in red colored nodes); hence, if it is possible to get a 'cheaper' junction conditioning with bigrams, the belief networks are restructured to appear only in two conditional variables at maximum.

and especially understanding of unknown complex objects will become feasible. From this work, it follows that the combination of both, the planar graph representation proposed in [6], and the statistical approach of grammatical inference by a N-gram model of this work, performs better than the subgraph matching approach proposed in [7], if they are compared in terms of runtime complexity and learning efficiency. However, as simple as humans may investigate an unknown object in order to understand its topology, the approach supports such a investigation by combining a sequence of images from different viewpoints for learning of implicit object topology.

Acknowledgment

This work was supported by project S9101 "Cognitive Vision" of the Austrian Science Foundation. The authors would like to thank Mrs. Fariba Dehghani-Schobesberger for the fruitful discussions on the topic of this work.

References

[1] F. Ackermann, A. Maßmann, S. Posch, G. Sageder, and D. Schlüter. Perceptual grouping of contour segments using markov random fields. *Patt. Rec. and I. A.*, 7(1):11–17, 1997.

[2] I. Biederman. Recognition by components. *Psych. Rev.*, 94, 1987.

[3] C. G. Christou, B. S. Tjan, and H. H. Bülthoff. Viewpoint information provided by a familiar environment facilitates object identification. *TR Max-Planck institute f. biol. cybernetics.*, 68, 1999.

[4] O. Faugeras. *Three-Dimensional Computer Vision.* MIT Press., 2001.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting. *Comm. of the ACM.*, 24(9):381–395, 1981.

[6] P. M. Goebel and M. Vincze. A cognitive modeling approach for the semantic aggregation of object prototypes from geometric primitives. In *Proc. ACIVS, Delft Univ. (NL)*, 2007.

[7] P. M. Goebel and M. Vincze. Implicit modeling of object topology with guidance from temporal view attention. In *Proc. of ICVW 2007, Bielefeld University (D)*, 2007.

[8] P. M. Goebel and M. Vincze. Vision for cognitive systems: A new compound concept connecting natural scenes with cognitive models. In *LNCS, INDIN 07, Austria.*, 2007.

[9] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics and Image Processing.*, 29(1):100–132, January 1985.

[10] C. Higuera. *Data Complexity in Pattern Recognition, Eds. M. Basu and Tin Kam Ho.*, chapter Data Complexity Issues in Grammatical Inference . Prentice Hall., 2006.

[11] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Intro. to Natural Language Processing, Comp. Linguistics and Speech Recognition.* Prentice Hall., 2003.

[12] C. Kermorvant and P. Dupont. Stochastic grammatical inference with multinomial tests. In *ICGI '02: Proc. on Grammatical Inf.* Springer, 2002.

[13] K. Koffka. *Principles of Gestalt Psychology.* Hartcourt, NY, 1935.

[14] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Learning hierarchical shape models from examples. In *Proc. EMMCVPR 2005*, 2005.

[15] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence.*, 31(3):335–395, 1987.

[16] D. Marr. *Vision: A Computational Approach.* Freeman & Co., 1982.

[17] S. Procter. *Model-Based Polyhedral Object Recognition Using Edge-Triple Features.* PhD thesis, Centre for Vision, Speech and Signal Processing, University of Surrey, UK., 1998.

[18] S. Sarkar. and K. L. Boyer. Integration, inference, and management of spatial information using bayesian networks. *IEEE Trans. Patt. Anal. and M. Intell.*, 15(3):256–274, 1993.

[19] S. Sarkar. and K. L. Boyer. Perceptual organization in computer vision: A review and a proposal for a classificatory structure. *IEEE Trans. on Systems, Man, and Cybernetics.*, 23(2):382–399, 1993.

[20] W. Stöcher and G. Biegelbauer. Automated simultaneous calibration of a multi-view laser stripe profiler. In *ICRA '05: Proc. of IEEE Int. Conf. on Rob. and Aut.*, 2005.

[21] R. F. Wang and E. S. Spelke. Human spatial representation: insights from animals. *Trends in Cognitive Sciences.*, 6(9), 2002.

[22] A. Witkin and J. Tenenbaum. On the role of structure in vision, in human and machine vision. In *Proc. Conf. on Human Vision.*, pages 481–543. Acad. Press Inc., 1983.

[23] M. Zillich. *Making sense of images: Parameter-free perceptual grouping.* PhD thesis, ACIN, Automation & Control Institute, Vienna University of Technology., 2007.

[24] S. W. Zucker. *Vision, Brain, and Cooperative Computation (M.A. Arbib and A.R. Hanson, Eds.)*, pages 231–262. Cambridge, MA: MIT Press., 1987.

Local descriptors for visual SLAM

Mónica Ballesta* Arturo Gil* Óscar Martínez Mozos† Óscar Reinoso*

Abstract

We present a comparison of several local image descriptors in the context of visual Simultaneous Localization and Mapping (SLAM). In visual SLAM a set of points in the environment are extracted from images and used as landmarks. The points are represented by local descriptors used to resolve the association between landmarks. In this paper, we study the class separability of several descriptors under changes in viewpoint and scale. Several experiments were carried out using sequences of images in 2D and 3D scenes.

1 Introduction

Building a map of the environment is a fundamental skill for a mobile robot, since maps are required for a series of high level tasks. Typical approaches use range sensors to build maps in two or three dimensions (e.g. [5, 6] [3, 14]).

Recently, the interest on using cameras as the main sensors to build the map has increased significantly. Such approach is denoted as visual SLAM. Typically, approaches using vision apply a feature-based SLAM (e.g, [2, 4, 8]), in which significant points in the environment are used as landmarks. Two steps can be distinguished in the utilization of visual landmarks: The detection of interest points and the description of the selected points. The first step involves the selection of suitable points in the images that can be used as landmarks. The points should be detected at different distances and viewing angles, since they will be observed by the robot from different poses. In a second step the landmarks are described by a feature vector which is computed using local image information. The descriptor is used to solve the data association problem: when the robot observes a landmark in the environment, it must decide whether the observation corresponds to a previously seen landmark or to a new one. The data association is a fundamental part of the SLAM process, since wrong associations will produce incorrect maps.

*Miguel Hernández University. E-mail:{m.ballesta|arturo.gil|o.reinoso}@umh.es. Supported by the Spanish Government under projects DPI2004-07433-C02-01 and PCT-G54016977-2005.

†University of Freiburg. E-mail:omartine@informatik.uni-freiburg.de. Supported by the EU under project CoSy FP6-004250-IP.

In practice, however, the interest points detected in the images are not very stable, and the matching between different views becomes difficult. In consequence, the problem of selecting a suitable interest point detector and descriptor for visual SLAM is still open.

In a previous work [11], we evaluated some interest point detectors to be used as landmarks in visual SLAM. The Harris corner detector was found to be the most suitable for visual SLAM applications. In this paper we present a comparison of different interest point descriptors using Harris corner detector as point detector.

In [10], Mikolajczyk and Schmid evaluated a set of local descriptors using a criterion based on the number of correct and false matches between pairs of images. Instead, in this work we concentrate on the variation of the descriptor when viewed from different angles and distances. We apply a pattern recognition approach using validity clustering measurements [13] to estimate how well the descriptors representing the same landmark along a sequence are grouped in the different descriptor spaces. This measurements will indicate which descriptor has better separability properties, facilitating the data association. Several experiments have been carried out using sequences of real indoor environment images. We believe that these results would help the selection of visual landmarks for SLAM applications.

2 Visual descriptors

Next, we list the set of different descriptors that have been evaluated in this study. For all of them we compute the descriptors at the local neighborhood of the points detected by Harris.

SIFT: The Scale-Invariant Feature Transform (SIFT) detects distinctive key points in images and computes a descriptor for them. The algorithm, developed by Lowe, was initially used for object recognition tasks [9]. SIFT features are located at maxima and minima of a difference of Gaussian functions applied in scale space. Next, the descriptors are computed based on orientation histograms at a 4x4 subregion around the interest point, resulting in a 128 dimensional vector.

SURF: Speeded Up Robust Features (SURF) is a scale and rotation invariant descriptor presented by Bay *et al.* [1]. The detection process is based on the Hessian matrix. SURF descriptors are based on sums of 2D Haar wavelet responses, calculated in a 4x4 subregion around each interest point. The standard SURF descriptor has a dimension of 64 and the Extended version (e-SURF) of 128. The u-SURF version is not invariant to rotation and has a dimension of 64.

Gray level patch: This method describes each landmark using the gray level values at a subregion around the interest point. This method has been used in [2] as descriptor of Harris points in a visual SLAM framework.

Orientation Histograms: The orientation histograms are computed from the gradient image, which represents the gray value variations in the x and y direction. In [7] orientation histograms are applied for navigation tasks.

Zernike Moments: The moment formulation of the Zernike polynomials [15] appears to be one of the most popular in terms of noise resilience, information redundancy and reconstruction capability. They are are constructed using a set of complex polynomials which form a complete orthogonal basis set defined on the unit disc.

3 Descriptor evaluation

To evaluate the stability of the different interest point descriptors under changes in viewpoint and scale we track each interest point along different images in a sequence. Examples of sequences are shown in Fig. 1. The interest points are extracted using Harris corner detector as shown in [11]. To track the points along the different images we have implemented two different algorithms for 2D and 3D images respectively. In the first case, we used the homography matrix as in [12]. In the case of 3D images, we have implemented a method that is based on the fundamental matrix. This method is divided in two steps. First, seven correspondences between each pair of images are selected, which allows to compute a fundamental matrix F. Using the fundamental matrix F we find a set of preliminary correspondences that are used as input for the computation of a second fundamental matrix F'. In this second step, the fundamental matrix is computed using a RANSAC approach, which results in a more accurate matrix F', that permits to find the correspondences with more precision.

For each tracked interest point p in a sequence of images $S = \{i_1, \ldots, i_N\}$, we obtain a set D_p of descriptor vectors $D_p = \{d_{p_1}, \ldots, d_{p_N}\}$. Each descriptor d_{p_n} represents the interest point p in the image i_n. The set D_p forms a cluster in the vector space representing the interest point p in the images along the sequence.

In this work, we use the J_3 separability criterion [13] to measure the separability of the clusters representing the interest points. This measure is based on two scatter matrices: S_w and S_b. S_w is called *within-class scatter matrix*, and measures the compactness of the clusters. The *between-class scatter matrix* S_b measures the separability between vectors belonging to different clusters. In our case, S_w measures the invariance of the descriptor to viewpoint and scale changes, whereas S_b measures the distinctiveness of the points described. The J_3 criterion is defined as:

$$J_3 = trace(S_w^{-1} S_m), \tag{1}$$

where S_m is the *mixture scatter matrix* and is computed as $S_m = S_w + S_b$. A good descriptor has a low value of S_w, since the variability of the vectors describing the same class is

Figure 1: The upper sequence shows a planar object (a poster) under different viewpoints. The bottom sequence depicts a 3D scene under different scale changes.

small. Furthermore, it is desirable that vectors describing different points are as distinctive as possible, resulting in a high value of S_b. In consequence, a suitable descriptor would have a high value of J_3. This descriptor would have good results in terms of the data association problem, despite of changes in the imaging conditions, such as viewpoint and scale changes. To compare descriptors with different length we use a normalized version $J_3' = \frac{J_3}{N}$, where N is the descriptor length.

4 Experiments

Tables 1 and 2 show the results of applying the J_3' criterion to different sequences of 2D and 3D scenes. The u-SURF descriptor achieves the highest value of separability in 96% of the sequences. However, u-SURF is not rotational invariant. When comparing only rotational invariant descriptors, SURF and e-SURF present similar results. In this case, the computational cost of computing the extended version of SURF is not worthy, since the results are not improved substantially. E-SURF always outperforms SIFT in changes in viewpoint. However, in scale changes it is only better in 43% of the cases (2D sequences).

Taking into account the results of Tables 1 and 2 together with the results of our previous work [11], we believe that the u-SURF descriptor in combination with the Harris corner detector is suitable for the common situation in which a robot explores the environment with a camera that only rotates around the vertical axis.

Table 1: J_3' values computed in the viewpoint changing sequences

Sequence	SIFT	SURF	e-SURF	u-SURF	Patch	Histogram	Zernike
2D sequences							
1	22.90	36.87	34.18	**126.63**	15.53	2.48	6.39
2	15.89	39.45	34.00	**119.58**	9.03	1.83	2.93
3	10.18	30.49	25.81	**118.64**	6.06	1.85	2.90
4	27.24	68.32	57.81	**184.06**	15.78	2.13	6.54
5	23.75	27.60	28.32	**55.94**	13.59	2.02	5.87
6	13.38	29.45	23.47	**67.36**	6.83	1.77	3.68
3D sequences							
7	5.71	10.70	10.70	**35.93**	2.59	1.46	2.13
8	17.62	16.45	18.96	**73.23**	5.99	1.51	4.33
9	7.11	7.83	7.65	**25.17**	3.33	1.72	2.35
10	16.44	14.47	16.60	**50.58**	7.37	1.54	5.54
11	6.22	9.60	9.41	**30.33**	2.76	1.78	2.25
12	10.26	9.63	11.13	**41.09**	4.00	1.43	3.43

Table 2: J_3' values computed in the scale changing sequences

Sequence	SIFT	SURF	e-SURF	u-SURF	Patch	Histogram	Zernike
2D sequences							
1	7.10	3.29	2.87	**8.82**	2.32	1.78	2.15
2	7.97	6.27	5.89	**13.67**	2.59	1.51	2.45
3	9.42	4.47	4.50	**13.03**	3.45	1.92	2.81
4	14.09	7.00	9.05	**26.89**	4.22	1.94	2.70
5	103.36	17.58	38.58	**131.54**	27.73	0.87	14.20
6	4.24	3.51	3.22	**8.56**	2.81	1.12	2.32
7	7.34	4.03	4.90	**12.71**	4.87	1.77	2.73
8	**26.49**	5.99	10.62	22.65	12.34	2.89	9.05
3D sequences							
9	7.06	10.12	10.24	**28.01**	4.47	1.70	3.10
10	14.48	10.39	14.97	**47.48**	5.98	1.67	4.54
11	8.76	9.18	10.02	**24.72**	3.47	2.48	3.95
12	22.22	15.53	23.09	**67.38**	8.50	2.15	5.61
13	6.28	8.84	10.00	**25.56**	3.56	1.94	3.06
14	17.45	11.10	16.86	**42.37**	7.37	2.10	5.88

5 Conclusions

We have performed an evaluation of visual local descriptors to be applied for SLAM tasks. For this purpose, we analyzed each descriptor according to its separability. The results of the experiments showed the behavior of seven different descriptors under changes in viewpoint and scale. We believe that this information will be useful when selecting an interest point descriptor as visual landmark for SLAM.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, 2006.

[2] Andrew J. Davison and David W. Murray. Simultaneous localisation and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[3] R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *IEEE Int. Conf. on Robotics & Automation*, 2005.

[4] A. Gil, O. Reinoso, W. Burgard, C. Stachniss, and O. Martínez Mozos. Improving data association in rao-blackwellized visual SLAM. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2006.

[5] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1), 2007.

[6] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Las Vegas, NV, USA, 2003.

[7] J. Kosecka, L. Zhou, P. Barber, and Z. Duric. Qualitative image based localization in indoor environments. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[8] J. Little, S. Se, and D.G. Lowe. Global localization using distinctive visual features. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2002.

[9] D.G. Lowe. Object recognition from local scale-invariant features. In *Int. Conf. on Computer Vision*, 1999.

[10] K. Mikolajczyk and C Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10), 2005.

[11] O. Martínez Mozos, A. Gil, M.Ballesta, and O. Reinoso. Interest point detectors for visual slam. In *Proc. of the Conference of the Spanish Association for Artificial Intelligence (CAEPIA)*, 2007.

[12] C. Schmid, R. Mohr, and C. Bauckhage. Evaluaton of interest point detectors. *International Journal of computer Vision*, 37(2), 2000.

[13] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press, third edition, 2006.

[14] R. Triebel and W. Burgard. Improving simultaneous mapping and localization in 3d using global constraints. In *National Conference on Artificial Intelligence (AAAI)*, 2005.

[15] F. Zernike. Diffraction theory of the cut procedure and its improved form, the phase contrast method. *Physica*, 1:689–704, 1934.

Computational Laban movement analysis using probability calculus

Joerg Rett* Jorge Dias†

Abstract

This work presents a system which implements the concept of Laban Movement Analysis (LMA) using probability calculus and Bayesian theory. Our Human-Interaction-Database (HID) provides sequences of position data from several persons performing distinct movements in 3-D (magnetic tracker) and 2-D (vision). From the position data a set of low-level features is calculated. Probability calculus is used to relate these low-level features and the frame of reference associated to the variables of LMA. The Bayesian theory provides the concept for calculation, learning and classification.

1 Introduction

When a person observes an actor performing a body movement he tries to anticipate the information the actor wants to convey. Analyzing the expressiveness of human movements has been under investigation for many centuries leading to several formalizations and concepts. Still, these concepts rely on humans analyzing other humans. Proving nowadays human-machine interfaces with a computational version of those concepts would be a big step toward *socially interactive robots*. To accomplish this, the gap of missing higher level cognitive systems that analyze the observations need to be closed. One can think of the problem as a scenario where a robot observes the movement of a human, analyzes the movement pattern and acts according to the extracted information (see fig. 1).

2 Laban movement analysis

Our approach is based on Laban Movement Analysis (LMA), which is a concept that provides descriptors for the static, as well as for the dynamic content of human body movements [1].

*Institute of Systems and Robotics E-mail:jrett@isr.uc.pt. Supported by FCT-Fundação para a Ciência e a Tecnologia Grant #12956/2003 and by the BACS-project-6th Framework Programme of the European Commission contract number: FP6-IST-027140, Action line: Cognitive Systems.

†Institute of Systems and Robotics E-mail:jorge@isr.uc.pt.

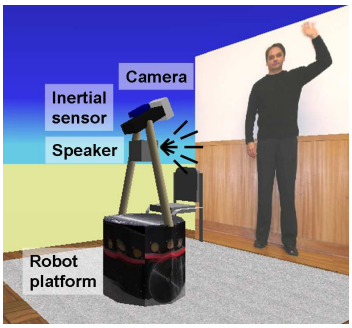


Figure 1: Nicole in position to interact.

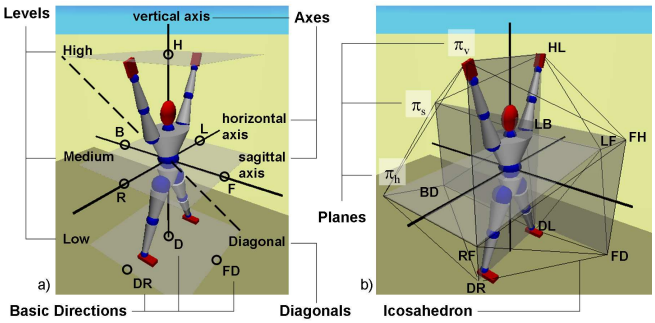
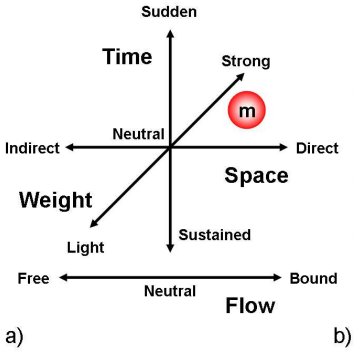


Figure 2: The concepts of a) Levels of Space, Basic Directions, Three Axes, and b) Three Planes and Icosahedron



Effort	Movement
Space Direct	Pointing gesture
- Indirect	Waving away bugs
Weight Strong	Punching,
- Light	Dabbing paint on a canvas
Time Sudden	Swatting a fly
- Sustained	Stretching to yawn
Flow Bound	Moving in slow motion
- Free	Waving wildly

b)

Figure 3: The bipolar *Effort* qualities: a) represented as a 4-D space containing a movement m, b) with their movement prototypes.

The theory of LMA consists of several major components. *Space* treats the spatial extent of the mover's *Kinesphere* (often interpreted as reach-space) and what form is being revealed by the spatial pathways of the movement. Different entities are specified to express movements in a frame of reference determined by the body of the actor as shown in fig. 2. In [3] the concept of *Vector Symbols* was presented, which is based on lines of motion rather than points in space. One part of these *Vectors symbols* are movements parallel to one of the *Axes* and movements along lines that are equally stressed in all three dimensions (*Diagonals*) (see fig. 2.a). The *Space* component also defines three planes, i.e. the *Door Plane* π_v, the *Table plane* π_h, and the *Wheel Plane* π_s as shown in fig. 2.b). *Effort* deals with the dynamic qualities of the movement and the inner attitude towards using energy. It consists of four motion factors: *Space*, *Weight*, *Time*, and *Flow*. As each factor is bipolar and can have values between two extremities, one can think of the *Effort* component as a 4-D space as shown in fig. 3.a. Prototypical movements where a certain *Effort*-value is predominant are presented in fig. 3.b.

3 Bayesian framework for LMA

The Bayesian framework enables us to model the dependencies between the low-level features and the descriptors of LMA. Bayesian-nets show the dependencies in a graphical way using nodes and links. Probability calculus enables us to compute the probability distribution for the values of a variable given the sets of known and unknown variables. The definitions for propositions, variables, probabilities and their conjunctions can be found in annex A.

Figure 4 shows the Bayesian-net of our system, representing the dependencies and thus, gives the possibility to simplify the joint distribution. The nodes represent variables (e.g. movement M) while the links describe the dependencies between the nodes. We have chosen to start with the variables that exhibit the highest level of abstraction (i.e. M), calling it the *Concept Space*. Their child nodes are found in the *Laban Space* which holds the sets of variables concerned with *Effort* and *Space*. We have called the lowest level of abstraction *Physical Space* which holds the set of low level variables (i.e. K, Vel and Acc). The direction symbols (i.e. A, B and C) are both, a Laban concept and a low-level feature. Our system uses movement segmentation to get distinct *Effort* parameters for each segment (phase Ph) of the movement [6]. The *Space* component of LMA is modeled using the concept of vector symbols. Our direction symbols A, B and C are calculated from the direction of the displacement vector, one for each plane π_v, π_h and π_s, respectively. The *Effort* component of LMA models the dependencies between e.g. the *Time* $E.Ti$ and the velocity Vel variable. Table 1 in the annex presents all variables used in the model with their name, symbol and a short description. The variable values and their cardinality are shown in (6) of the annex. We can describe the joint distribution while omitting the conjunction symbol $\wedge$ as:

$$\begin{aligned} & P(I\ M\ Ph\ E.Sp\ E.We\ E.Ti\ E.Fl\ A\ B\ C\ K\ Vel\ Acc) \\ = \ & P(I)\ P(M)\ P(Ph)\ P(E.Sp \mid M\ Ph)\ P(E.We \mid M\ Ph)\ P(E.Ti \mid M\ Ph) \\ & \quad P(E.Fl \mid M\ Ph)\ P(A \mid M\ I)\ P(B \mid M\ I); P(C \mid M\ I) \\ & \quad P(K \mid E.Sp\ E.We)\ P(Vel \mid E.Sp\ E.We\ E.Ti\ E.Fl) \end{aligned} \tag{1}$$

Knowing that the variables are all discrete we can express conditional probability distributions (e.g. $P(A \mid M\ I)$ as tables. Learning is represented in our model through the process of filling all the conditional probability tables with values. The tables are filled by collecting all evidences for a given assumption and creating a probability distribution. In the case of $P(A \mid M\ I)$ a number of direction symbols $a_1, a_2, \ldots a_n$ is collected form n trials for a known movement $M = m$ and frame $I = i$. A distribution can be obtained by building a histogram. As this scheme requires a set of labeled data we may call it supervised learning.

Classification is the final step after the model has been established and the tables have been learned. Given our joint distribution $P(I\ M\ Ph\ E.Sp\ E.We\ E.Ti\ E.Fl\ A\ B\ C\ K\ Vel\ Acc)$ we need to formulate a question, i.e. what we want to classify and what we can observe. In

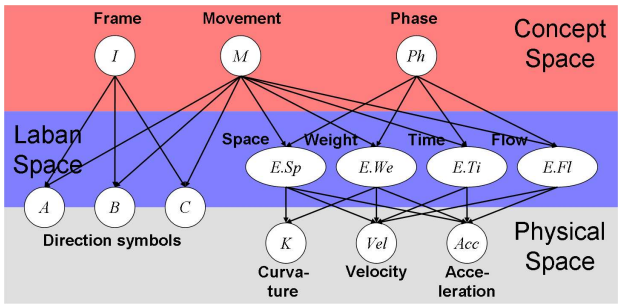


Figure 4: Bayes-Net of the LMA model.

our case we are interested to classify an unknown movement from the evidences observed in the *Physical Space*. In the following we continue with a simplified question, i.e. classifying a movement M taking into account only the direction symbols A and the frame I (2).

$$P(M \mid I\ A) = P(M)P(A \mid M\ I) \tag{2}$$

We can compute the *likelihood* of a sequence of n direction symbols by assuming that the observed direction symbols are independently and identically distributed (i.i.d.). The joint probability will be the product of the probabilities for each frame as shown in (3).

$$P(a_{1:n} \mid m\ i_{1:n}) = \prod_{j=1}^{n} P(a_j \mid m\ i_j) \tag{3}$$

The probability distribution of the movements M after observing $n+1$ direction symbols A can be formulate in a recursive way. Assuming that each frame I a new observed direction symbols arrives, we can state and express the online behavior by (4).

$$P(M_{n+1} \mid i_{1:n+1}\ a_{1:n+1}) = P(M_n)\ P(a_{n+1} \mid M\ i_{n+1}) \tag{4}$$

The probability distribution of m for $n = 0$ is the *prior*. As a rule for classification we use the maximum a posteriori (MAP) method.

4 Implementation of LMA

We have created a database of human movements, called Human Interaction Database (HID) which is accessible through *WWW* [5]. The database consists of image sequences, high precision 3-D position data and results from our visual tracker and classifier. The HID holds the movements suggested in fig. 3. The high precision 3-D position data is obtained from a 6-DoF magnetic tracker (Polhemus Liberty) with sensors attached to several body parts and objects. To collect the visual tracking data we use the gesture perception system (GP-System)

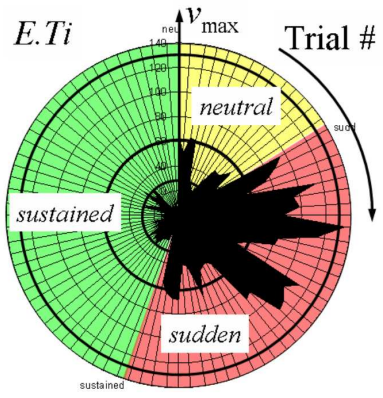


Figure 5: Evaluating the maximum velocity v_{max} on several movement trials with known Effort-Time.

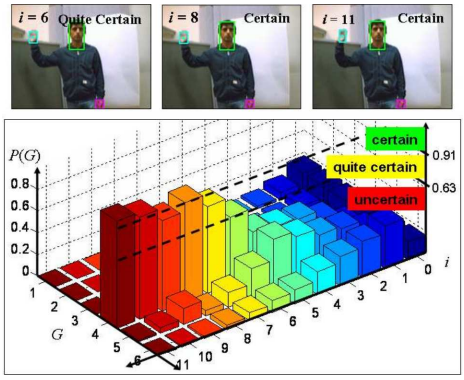


Figure 6: Evolution of the movement probabilities $P(M)$ along the time (i).

[4] of our social robot Nicole. The tracking data consists of: i) the 2-D or 3-D position $\mathbf{X}_{bp}$ of a point belonging to a body part bp and ii) the timestamp t_i given by some timer function of the system. From the tracking data, the values of velocity, acceleration and curvature are calculated for each of the three planes π_v, π_h and π_s. To implement learning and classification we use the *Bayesian programming* methodology [2]. A *Bayesian Program* (BP) is a generic formalism to build probabilistic models and to solve decision and inference problems on these models. An example for a generic *Bayesian Program* is shown in fig. 7 in the annex.

5 Results

To evaluate the importance of low-level features as evidences for the Laban parameters, 75 trials were performed, each represented by a sector of the circle shown in fig. 5. It can be seen that most of the high (130 - 60) velocities can be found in the *sudden* sector. Similar is true for the medium velocities (60 - 30) and the *neutral* sector and also for the low velocities (30 - 0) and the *sustained* sector.

The evolution of the anticipated movements and the certainty of the belief is shown in fig. 6. The performance of a *Bye-Bye* gesture, shows which gesture the agent anticipates due to the highest probability and how certain he is about his guess.

6 Conclusion

This article gave an introduction to the framework of Laban Movement Analysis (LMA) emphasizing those entities that were expected to be useful for a computational representation.

We presented the Bayesian models for computational LMA in form of Bayesian-nets and joint distributions. A simplified model using the *Space* component of LMA to classify movements was chosen to present the concept of learning and classification. We showed our technical approach to track human movements (i.e. visual and magnetic) and store the output in our Human-Interaction Database (HID). We presented the set of low-level features calculated from the tracked data and the methodology of *Bayesian Programming*. The results indicated that there is a set of low-level features that can be used as evidences for the Laban parameters and that the classifier is able to make online-predictions, thus giving the system a sense of anticipation. In the application of socially assistive robots we are developing and evaluating the feasibility of using our social robot Nicole for rehabilitation. We have ongoing work in the area of smart houses and environments to apply computational LMA to *people tracking*.

References

[1] I. Bartenieff and D. Lewis. *Body Movement: Coping with the Environment*. Gordon and Breach Science, New York, 1980.

[2] Julien Diard, Pierre Bessière, and Emmanuel Mazer. A survey of probabilistic models, using the bayesian programming methodology as a unifying framework. In *Proc. of the Int. Conf. on Computational Intelligence, Robotics and Autonomous Systems*, Singapore (SG), December 2003.

[3] J. S. Longstaff. Translating vector symbols from laban's (1926) choreographie. In *26. Biennial Conference of the International Council of Kinetography Laban, ICKL, Ohio, USA*, pages 70–86, 2001.

[4] J. Rett and J. Dias. Visual based human motion analysis: Mapping gestures using a puppet model. In *Proceedings of EPIA 05, Lecture Notes in AI, Springer Verlag, Berlin*, 2005.

[5] J. Rett, A. Neves, and J. Dias. Hid-human interaction database: http://paloma.isr.uc.pt/hid/, 2007.

[6] N. Rossini. The analysis of gesture: Establishing a set of parameters. In *Gesture-based Communication in Human-Computer Interaction, LNAI 2915, Springer Verlag*, pages 124–131, 2003.

Variable	Symb.	Description
Frame	I	index of the data frame
Movement	M	Type of movement, e.g. $M = pointing$
Phase	Ph	Temporal segment, e.g. $Ph = Rest$
Effort Space	$E.Sp$	e.g. $E.Sp = direct$
Effort Weight	$E.We$	e.g. $E.We = strong$
Effort Time	$E.Ti$	e.g. $E.Ti = sudden$
Effort Flow	$E.Fl$	e.g. $E.Fl = free$
Direct. Symb.	A, B, C	Vector Symbols (Atoms) in π_v, π_h, π_s
Curvature	K	Change of displacement angles
Speed	Vel	Velocity level, e.g. $Vel = low$
Speed Gain	Acc	Acceleration level, e.g. $Acc = high$

Table 1: Global variables

A Annex

A logical proposition can be either true or false. The conjunction of propositions a and b is denoted $a \wedge b$ and the negation of proposition a by $\neg a$. Variables are denoted by names starting with one uppercase letter. A discrete variable X is a set of logical propositions x_i which means that the variable X takes its ith value. $\langle X \rangle$ denotes the cardinal of the set X. To be able to deal with uncertainty, we attach probabilities to propositions. To each proposition a a unique real value P(a) in the interval [0, 1] is assigned. The probability of conjunctions of propositions is denoted by $P(a \wedge b)$. The probability of a proposition a conditioned by some other proposition b is denoted by $P(a|b)$. The conjunction rule (5) gives the probability of a conjunction of propositions.

$$P(a|b) = P(a) \times P(b|a) = P(b) \times P(a|b) \tag{5}$$

All variables used in the model are presented in table 1.

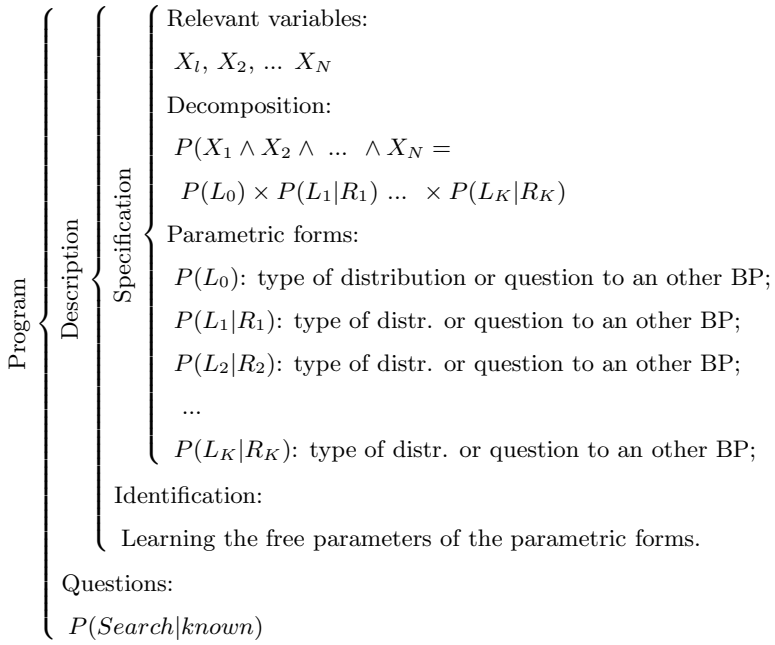


Figure 7: Generic Bayesian Program.

Variables of the *Laban Space* and *Physical Space* with their values and cardinality.

$$\begin{aligned}
E.Sp &\in \{direct, neutral, indirect\}\ \langle 3\rangle \\
E.Ti &\in \{sudden, neutral, sustained\}\ \langle 3\rangle \\
E.We &\in \{strong, neutral, light\}\ \langle 3\rangle \\
E.Fl &\in \{bound, neutral, free\}\ \langle 3\rangle \\
A &\in \{O, U, UR, R, DR, D, DL, L, UL\}\ \langle 9\rangle \\
B &\in \{O, F, FR, R, BR, B, BL, L, LF\}\ \langle 9\rangle \\
C &\in \{O, U, UF, F, DF, D, DB, B, UB\}\ \langle 9\rangle \\
Vel &\in \{rest, slow, medium, fast\}\ \langle 4\rangle \\
Acc &\in \{zero, low, medium, high\}\ \langle 4\rangle \\
K &\in \{180, 135, 90, 45, 0, -45, -90, -135\}\ \langle 8\rangle
\end{aligned} \tag{6}$$

An example for a generic *Bayesian Program*.

Visual servoing for floppy robots using LWPR

Fredrik Larsson* Erik Jonsson * Michael Felsberg*

Abstract

We have combined inverse kinematics learned by LWPR with visual servoing to correct for inaccuracies in a low cost robotic arm. By low cost we mean weak inaccurate servos and no available joint-feedback. We show that from the trained LWPR model the Jacobian can be estimated. The Jacobian maps wanted changes in position to corresponding changes in control signals. Estimating the Jacobian for the first iteration of visual servoing is straightforward and we propose an approximative updating scheme for the following iterations when the Jacobian can not be estimated exactly. This results in a sufficient accuracy to be used in a shape sorting puzzle.

1 Introduction

Initially an analytical closed-form inverse kinematics solution for a 5 DOF robotic arm was developed and implemented. This analytical solution proved not to meet the accuracy required for a general assembly setup, e.g. a shape sorting puzzle like the one used in the COSPAL (*COgnitive Systems using Perception-Action Learning*) project [1, 4]. The correctness of the analytical model could be confirmed through a simulated ideal robot and the source of the problem was deemed to be nonlinearities introduced by weak servos unable to compensate for the effect of gravity. Instead of developing a new analytical model, which took the effect of gravity into account, a learning approach was selected.

As learning method we chose Locally Weighted Projection Regression (LWPR) [11]. This is an incremental supervised learning method and is considered a state-of-the-art method for function approximation in high dimensional spaces.

LWPR by itself was not able to give us the accuracy needed and we combined the trainined LWPR model with the well known concept of *visual servoing* [8]. We show how to overcome

*Computer Vision Laboratory. Dep. of EE, Linköping University, Sweden. E-mail: larsson@isy.liu.se.
This work has been supported by EC Grant IST-2003-004176 COSPAL. This paper does not represent the opinion of the European Community, and the European Community is not responsible for any use which may be made of its contents.

the difficulties that arise from the merge of the two methods and present results showing a high level of accuracy.

2 Locally weighted projection regression

LWPR is an incremental local learning algorithm for nonlinear function approximation in high dimensional spaces and has successfully been used in learning robot control [10, 9]. The key concept in LWPR is to approximate the underlying function by local linear models. The LWPR-model automatically updates the number of receptive fields (RFs), i.e. local models, as well as the location (which is decided by the RF center $\mathbf{c}$) of each RF. The size and shape of the region of validity (decided by the distance metric $\mathbf{D}$) of each RF is updated continuously based on the performance of each model. Within each local model an incremental version of weighted partial least-squares (PLS) regression is used.

LWPR uses a Gaussian weighting kernel to calculate the *activation* or *weight* of RF k (the subscript k will be used to denote that the particular variable or parameter belongs to RF k) given query $\mathbf{x}$ according to

$$w_k = \exp(-\frac{(\mathbf{c}_k - \mathbf{x})^T \mathbf{D}_k (\mathbf{c}_k - \mathbf{x})}{2}). \tag{1}$$

Note that (1) can be seen as a non-regular channel representation of Gaussian type if the distance metric $\mathbf{D}_k$ is equal for all k [5].

The predicted output $\hat{\mathbf{y}}$ is given as the weighted output of all RFs according to

$$\hat{\mathbf{y}} = \frac{\sum_{k=1}^{K} w_k \hat{\mathbf{y}}_k}{\sum_{k=1}^{K} w_k} \tag{2}$$

with K being the total number of RFs.

The output of each RF can be written as a linear mapping

$$\hat{\mathbf{y}}_k = \mathbf{A}_k \mathbf{x} + \boldsymbol{\beta}_{k,0} \tag{3}$$

where $\mathbf{A}_k$ and $\boldsymbol{\beta}_{k,0}$ are known parameters acquired through the incremental PLS. The incremental PLS bears a resemblance to incremental associative networks [7], one difference being the use of subspace projections in PLS.

We have been using LWPR to learn the mapping between the configuration $\mathbf{x}$ of the end-effector and the control signals $\mathbf{y}$. All training data was acquired through image processing since no joint-feedback was available from the robotic arm used. To reach a high level of accuracy we combined the moderately trained LWPR model with visual servoing.

3 Visual servoing based on LWPR

We have been using position based visual servoing (categorized according to [8]) to minimize the norm of the deviation vector $\Delta\mathbf{x} = \mathbf{x}_w - \mathbf{x}$, where $\mathbf{x}$ denotes the reached configuration and $\mathbf{x}_w$ denotes the desired configuration of the end-effector.

If the current position with deviation $\Delta\mathbf{x}$ originates from the control signal $\mathbf{y}$, the new control signal is given as $\mathbf{y}_{\text{new}} = \mathbf{y} - \mathbf{J}\Delta\mathbf{x}$, where the Jacobian $\mathbf{J}$ is the linear mapping that maps changes $\Delta\mathbf{x}$ in configuration space to changes $\Delta\mathbf{y}$ in control signal space. When the Jacobian has been estimated the task of correcting for an erroneous control signal is in theory rather simple.

Using LWPR as a basis for visual servoing is a straightforward procedure for the first iteration. LWPR gives a number of local linear models from which the Jacobian can be estimated. However, problems arise when we need to update the Jacobian to use it for the following iterations.

Equation (1),(2) and (3) give $\mathbf{J}$ as

$$\mathbf{J} = \frac{d\hat{\mathbf{y}}}{d\mathbf{x}} = \frac{\sum_{k=1}^{K} w_k(\mathbf{A}_k + (\hat{\mathbf{y}} - \hat{\mathbf{y}}_k)(\mathbf{x} - \mathbf{c}_k)^T\mathbf{D}_k)}{\sum_{k=1}^{K} w_k}. \tag{4}$$

The problem of updating $\mathbf{J}$ after the first iteration is due to the fact that the current utput was obtained by use of the old $\mathbf{J}$ and not by the LWPR model. This means that we do not know which query $\mathbf{x}$ that would give us the current $\widehat{\mathbf{y}}$ and as can be seen in (4) this is needed. The solution to this non-trivial problem is the main contribution of this paper. We propose a static approach and an approximative updating approach.

Static approach: The simplest solution is the static approach. The Jacobian is simply not updated and the Jacobian used in the first step is (still) used in the following steps.

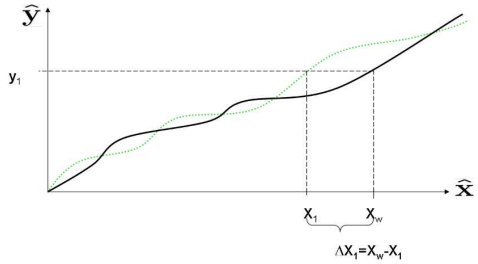
Approximative updating approach: The somewhat more complex solution treats the LWPR model as if it was exact. This means that we use the reached position as query and estimate the Jacobian for this configuration. The procedure is explained in Figure 1.

4 Results

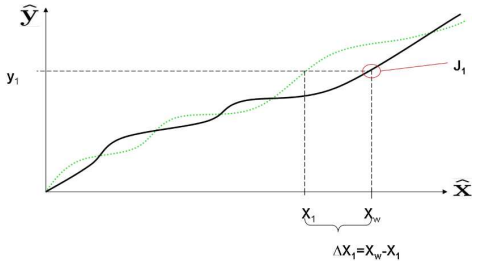
The real world experimental setup consisted of a low cost robotic arm of Lynx-6 type [2] (see figure 2) and a calibrated stereo rig. The end-effector of the robotic arm was equipped with three spherical markers in distinct colors. By stereo triangulation, the 3D position of the spherical markers were obtained relative one of the cameras. The positions were then

transformed into the robot frame. For the results presented below, we only deal with the 3D-position of the end-effector, neglecting the rotation and approach angle.

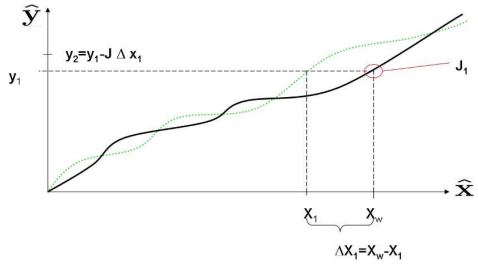
The test scenario used is a reduced 3D scenario. The end-effector can be positioned in two different planes (the grip- or the movement-plane) and the approach vector is to be perpendicular to the ground plane. The task space of the robotic arm is restricted (by physical and practical constraints) to a half circle with radius of 240 mm. Training points were acquired by using the inaccurate analytical model.



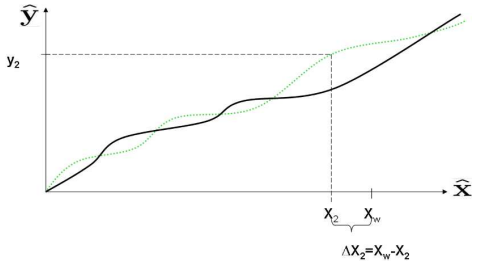
A : Given the wanted configuration $\mathbf{x}_w$ we obtain the first precidition $\mathbf{y}_1$. Which results in deviation $\Delta\mathbf{x}_1$.



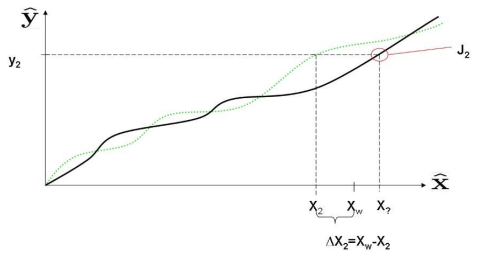
B : The true Jacobian $\mathbf{J}_1$ is estimated.



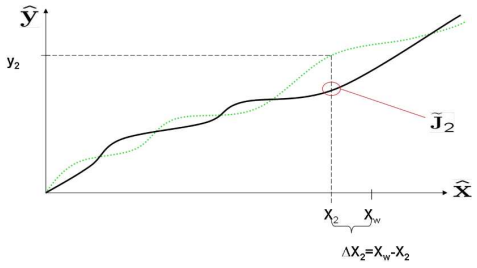
C : The prediction is updated, giving $\mathbf{y}_2$.



D : $\mathbf{y}_2$ results in $\mathbf{x}_2$ with deviation $\Delta\mathbf{x}_2$.



E : The true Jacobian $\mathbf{J}_2$ can not be estimated due to the unknown $\mathbf{x}_?$.



F : The approximative Jacobian $\widetilde{\mathbf{J}}_2$ is estimated.

Figure 1: The approximative updating approach explained. The dotted line represents the true function and the solid line the LWPR approximation.

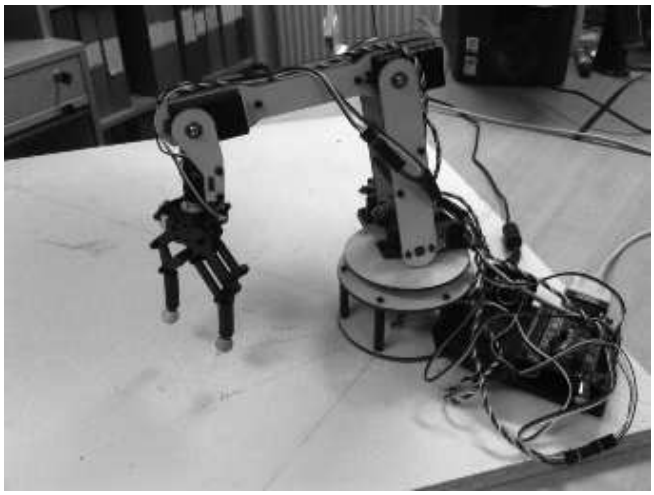


Figure 2: Our Lynx-6 robotic arm positioned in the movement-plane. The spherical markers can be seen at the end of the end-effector

Table 1 contain the results from real world experiments. LWPR denotes the mean deviation with just the trained LWPR model. J Static/Update denotes whether the static or the updating approach has been used for visual servoing. It is worth noticing that perfect positioning with just the estimated noise added would correspond to a mean error of 2.05 mm.

Real World Evaluation

Training points:	100	500	1000	5000
LWPR	16.89	12.83	7.53	8.78
J Static	9.83	5.41	1.79	1.64
J Update	9.07	4.32	1.65	1.65
Analytical solution:		15.87		

Table 1: Mean deviation in mm from desired position. 50 test points were used for evaluation except from in the analytical case were 100 test positions were used. Stopping criteria for the visual servoing was 10 iterations or a deviation less than 1 mm.

5 Discussion

By combining LWPR with visual servoing we have reached an accuracy sufficient for a shape sorting puzzle. The main novelty of this paper is to present two methods to overcome the difficulties that arise from the merge of LWPR and visual servoing. The results show that the approximative updating approach is favorable. To further improve the accuracy we would need to reduce the noise in the estimated positions since it is currently the limiting factor.

The restrictions imposed by the test scenarios mean that we are avoiding the problems with the solution to the inverse kinematic problem not being one to one. However, if the training data shown to the LWPR model would have some ambiguities this may cause problems. In fact, if all positions would be reachable with servo 1 set to e.g. $+\pi$ or $-\pi$, the linear averaging of the LWPR model will predict the output for servo 1 to 0. Of course, this can be avoided with preprocessing of the signals, e.g. using the *channel representation* [6] which allows for robust estimation of multiple modes [3].

References

[1] COSPAL project. http://www.cospal.org/, January 2007.

[2] Lynxmotion robot kits. http://www.lynxmotion.com/, January 2007.

[3] M. Felsberg, P.-E. Forssén, and H. Scharr. *IEEE Transactions on Pattern Analysis and Machine*, 28(2):209–222, February 2006.

[4] M. Felsberg, J. Wiklund, E. Jonsson, A. Moe, and G. Granlund. Exploratory learning structure in artificial cognitive systems. In *ICVW*, 2007.

[5] P-E. Forssén. *Low and Medium Level Vision using Channel Representations*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, March 2004. Dissertation No. 858, ISBN 91-7373-876-X.

[6] G. H. Granlund. An associative perception-action structure using a localized space variant information representation. In *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Kiel, Germany, September 2000.

[7] E. Jonsson, M. Felsberg, and G. Granlund. Incremental associative learning. Technical Report LiTH-ISY-R-2691, Dept. EE, Linköping University, Sept 2005.

[8] D. Kragic and H. I. Christensen. Technical report, ISRN KTH/NA/P–02/01–SE, Jan. 2002., CVAP259.

[9] S. Schaal, C.G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

[10] S. Vijayakumar, A. D'souza, T. Shibata, J. Conradt, and S. Schaal. Statistical learning for humanoid robots. *Auton. Robots*, 12(1):55–69, 2002.

[11] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional spaces. In *Proceedings ICML*, pages 288–293, 2000.