Learning To Locate An Object in 3D Space From A Sequence Of Camera Images

Dimitris Margaritis
dmarg+@cs.cmu.edu
Sebastian Thrun
thrun+@cs.cmu.edu

Dept. of Computer Science
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213

Abstract

This paper addresses the problem of determining an object's 3D location from a stream of camera images recorded by mobile robot. The approach presented here allows people to "train" robots to recognize specific objects, by presenting it examples of the object to be recognized. A decision tree method is used to learn significant features of the target object from individual camera images. Individual estimates are integrated over time using Bayes rule, into a probabilistic 3D model of the robot's environment. Experimental results illustrate that the method enables a mobile robot to robustly estimate the 3D location of objects from multiple camera images.

Keywords: Mobile Robotics, Decision Trees, Probabilistic Reasoning.

Contact author:
Dimitris Margaritis

office: (412) 268–6767
lab: (412) 268–3837

1 Introduction

In recent years, there has been significant progress in the field of mobile robotics. Applications such as robots that guide blind or mentally handicapped people, robots that clean large office buildings and department stores, robots that assist people in recreational activities, etc., are slowly getting in reach. Many of these robots must integrate mobility with manipulation. They must be able to move around, and they must also be capable of manipulating their environment. For such robots, their practical success will partially depend on their ability to identify and localize objects.

This paper addresses the problem building robots that can be *trained* to recognize and locate user-specified objects. More specifically, it proposes an algorithm that enables people to train robots by simply showing a few poses of the object. Once trained, the robot can recognize these objects and determine their location in 3D space. In contrast to existing approaches to mobile manipulation, which usually assumes that objects are located in floor or table-height, our approach does not make restrictive assumptions as to where the object is located. This poses new challenges on the ability to localize objects, as a single camera image is insufficient to determine the location of an object in 3D space.

The approach proposed here uses probabilistic representations to estimate the identity and location of the target object from multiple views. It maps camera images into 2D probabilistic maps, which describe, for each pixel in the camera image, the likelihood that this pixel is part of the target object. This mapping is established by a decision tree applied to local image features, which is constructed during the training phase from labeled images. The 2D probabilistic map is then projected into the 3D work space, based on straightforward geometric considerations. Since a single camera image is insufficient to determine the location of an object in 3D, our approach integrates information from multiple images, taken from multiple viewpoints. It employs Bayes rule to generate a consistent probabilistic 3D model of the workspace. Our approach also takes into account the uncertainty introduced by robot motion, by using a probabilistic model of robot motion. As the robot moves in the environment taking images, it gradually improves the estimation of the identity and location of an object, until it finally knows what and where the object is. Experimental results using a RWI B21 robot equipped with a color camera show that multi-part objects can be located robustly and with high accuracy.

The remainder of this paper is organized as follows. In section 2, we describe decision trees along with the way our approach uses them for characterizing images. In section 3, we show how image information is integrated into a 3D model, and

provide a method for accommodating the uncertainty that is introduced by robot motion. In section 4, we present experimental results, obtained with a RWI B21 robot, followed by a survey of related research (section 5). Finally, in section 6, we comment on the assumptions and limitations of the approach and suggest directions for future research.

2 Decision Tree Learning

A decision tree is a succinct way of describing a multidimensional discrete-valued function $f : \mathcal{R}^n \times \mathcal{X}^m \rightarrow \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are finite sets of discrete elements and $\mathcal{R}$ is the set of real numbers. The $(n + m)$ inputs to this function frequently correspond to discrete and/or continuous-valued attributes of an object and the output represents an object's property that we want to predict. Each node of the tree is associated with a partition of the input space. An internal node further partitions its space into two subspaces based on the value of a single input variable, associating each of the resultant subspaces to each of the two children. The set of decision trees is complete in the space of discrete-valued functions i.e. any such function can be represented by at least one decision tree. An example of a decision tree, obtained in the context of image analysis (see below), is illustrated in Fig. 1.

Decision tree growing often employs a statistical test at each node in order to determine the most discriminative input to split against. A popular choice of test is *information gain maximization*. While decision trees can perfectly fit a set of input-output tuples that are used in growing them (typically called *training set*), we are often interested in their performance in accurately predicting future, yet unseen data. As such a perfect fit of the training set is not necessarily desirable and can result in *overfitting*. Overfitting is the situation where the error over the training data is low but the error over future data is high. An technique often employed in improving generalization ability is *cross-validation*, which amounts to withholding a randomly chosen subset of the input data, called the *test set*, and testing the performance of the decision tree against this data during growth, stopping when it begins to deteriorate significantly (because of overfitting).

It is common practice to penalize complexity when inducing decision trees [Qui86, Mit97]. This is because large trees tend to overfit the data, which usually has a negative effect on the generalization accuracy. The literature has produced two different paradigms for penalizing complexity: restricting the growth of the tree, and post-pruning fully grown trees. The latter is the approach adopted in this paper. A *pruning set* of examples, disjoint from the training and test sets, is used for cross-validation in pruning. Starting from the root node, nodes are repeatedly pruned and

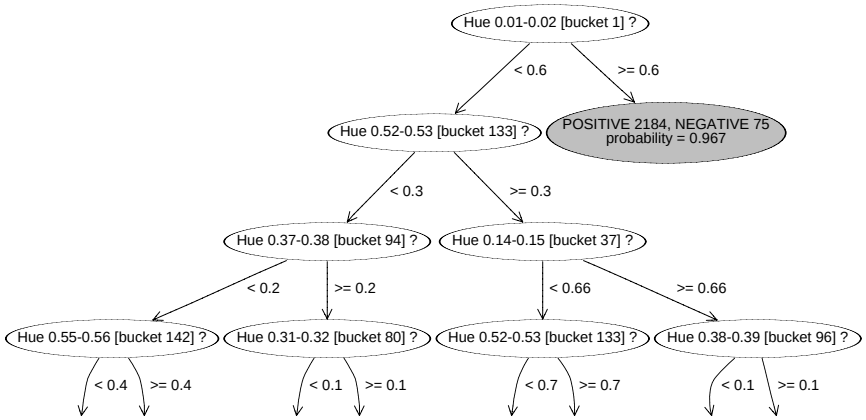

Figure 1: The top few nodes of an example decision tree. A leaf represents the probability conditioned on the values of the attributes. Internal nodes test on the fraction of positive pixels of a tile that fall in the corresponding hue range.

the training examples falling in the subtree rooted in that node are condensed into a single leaf. Performance of the resulting tree is then evaluated on the pruning set. If there is improvement, this condensation step becomes permanent. Otherwise the original node is restored and the pruning procedure recursively descends to the children of the node. Pruning using a cross-validation set usually increases the generalization accuracy while making the tree more compact.

Our approach uses trees to approximate conditional probability density functions. Decision tree are usually used to answer / queries regarding the output value of f given an input tuple of values. If, for example, f is a boolean-output function, querying is typically done by comparing the number of positive and negative training examples that were assigned during training to the leaf node that is associated with the partition that the input tuple lies in. The algorithm would then return the value (/) that is in majority in that leaf. In our use of a decision tree we differ in that we instead output the fraction of positive or negative examples found in the leaf. As such, we use the decision tree to represent an *approximation of the probability density function* on the output space conditioned on the values of attributes in the input space of f. If appropriately pruned, these probabilities are usually not zero or one because of training set noise in either the values of the inputs or the output or non-determinism due to use of a set of input variables that is insufficient to deterministically model f.

2.1 A Pdf for Characterizing an Image

Our approach uses a decision tree to map (filter) camera images into 2D probabilistic maps, which describe the probability of the presence of a target object at the various locations in the image. More specifically, the inputs to the tree are *image features* in a local region (called: *tile*) in the image, and the output is a probability value that measures the likelihood of the presence of a target object in the respective tile. In principle, our approach can be applied to arbitrary image features (e.g., pixels, edges, brightness, color, texture, etc.). In our implementation, *local color histograms* are used as input to the decision tree.

The tree is learned using labeled training examples. More specifically, construction of the training, test and pruning sets is typically done using the following procedure:

1. An input picture is obtained.
2. A rectangle R is drawn around the object by the user. This might include parts of the background.
3. The image is divided in a matrix of non-overlapping rectangular tiles, completely covering its surface. The size of each tile is small relative to the projection of the object on the image. 8×8 is used in this paper.
4. Each tile is used to construct a single positive or negative example. The features that occur in the tile, which can be continuous or discrete, are extracted and used as input values for the example associated with that tile.
5. Depending on whether each tile is fully contained within R or not, the example is assigned to be positive or negative, respectively.

This set of examples is equally divided into training, test and pruning sets, and these are used in growing a decision tree for that combination of object and environment that it was seen in. The resulting tree, when applied to new images within that environment, provides probability densities for the presence of a target object.

Figure 2 illustrates our method. Shown there, in the top row, is a series of tree training images. The target object is labeled by hand. The bottom row shows a testing image, along with the probability field generated by the tree. As can be seen there, the algorithm assigns high likelihood to the correct location, but also misclassified a small number of regions in the image background. From this single camera image, it is impossible to determine the location of the target object in 3D coordinates. The remainder of this paper describes our approach to integrating these probabilistic estimates in 3D space.

Figure 2: Detection of a bottle from previous examples. The top row contains images where the outlined part contains the tiles used as positive examples. The rest of the image's tiles are negative examples. Probabilities above 0.8 are marked in the previously unseen picture in the bottom row. Not shown are another set of 18 "background" pictures consisting of negative examples only.

3 Integrating Multiple Camera Imaged in 3D

Our approach integrates the probabilistic information, extracted from individual images, into a spatial 3D model of the world. Our approach represents information about the location of the object in a 3D occupancy grid. Each grid cell is associated with an approximation of the probability that part of the object occupies that particular cell. Each such probability is initialized with a number that corresponds to a prior belief that the object occupies a cell given no information about the world. This number can be learned from data, typically though counting according as per the frequentists' approach to probability. The exact value of the prior is not significant in the long term, since the value will converge towards the actual probability after a sufficient number of observations. However, if there is evidence that the object in question occurs more frequently in certain areas (for example, a shoe may be expected to lie on the floor most of the time), this information can be used to appropriately initialize prior probabilities and assign higher values to these locations. During detection, each information-gathering step is followed by an updating of the

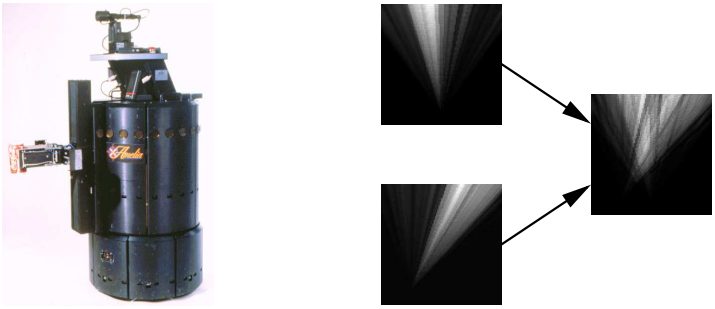

Figure 3: On the left, the robot used in our experiments is shown. It is equipped with a parallel two-fingered gripper for object manipulation. On the right, an illustration is shown of how information from images taken from two different viewpoint is integrated in the occupancy grid. Shown to the left are two single projections applied to an "empty" grid. The picture on the right shows how they are combined together. The images depict the average values of grid cell probabilities when viewed from above (i.e. averaging probability values along the z-axis).

probability of each cell according to Bayes law, as described below. Robot motion also affects the grid due to uncertainty of the robot's translational and rotational velocities.

3.1 Information Integration

The key idea for mapping 2D image information into a 3D spatial representation is to map image tiles into *pyramids* in space. Each image obtained from the environment provides us with information about the location of parts of the object. Since we assume a single camera input, we have no information about the depth of features contained in one of the tiles of the image. We therefore make no assumption on the distance of the part from the eyepoint. However, we do obtain information about the Euler angles (azimuth θ and altitude ϕ) of the feature with respect to the robot's current location. In particular we know that it is contained within the pyramid emanating from the eyepoint whose four converging sides intersect the four corners of the tile on the image plane that is perpendicular to the direction the camera is facing. Grid cells intersecting this pyramid are therefore updated using Bayes law.

An example of the updating is shown in Fig. 3. Here two different pyramids are shown (projected into the x-y plane), which have been generated from camera images taken at different locations. Bayes rule is applied to integrate these pyramids, in order to generate a single, consistent belief.

The integration works as follows. The probability that a part of the object occupying a cell at grid location (x, y, z) at time t is denoted by $\Pr[\xi(x, y, z, t)]$. Coordi-

nates x, y and z are with respect to a fixed, world-centered coordinate system (they are not local robot-centered coordinates). $\xi(x, y, z, t)$ is a boolean random variable denoting the existence of a part of the object at a location somewhere inside the corresponding grid cell. In the following we will use ξ instead of $\xi(x, y, z, t)$ for the sake of brevity. If $i(t)$ denotes the image obtained at time t and $D(t-1)$ the set of previous images/motion commands in all previous steps, the probability value $p(\xi)$ at grid cell location ξ is computed as follows:

$$\begin{aligned} p(\xi) &\equiv \Pr[\xi|i(t), D(t-1)] \\ &= \Pr[i(t)|\xi, D(t-1)] \, \frac{\Pr[\xi|D(t-1)]}{\Pr[i(t)|D(t-1)]}. \end{aligned}$$

$\Pr[\xi|D(t-1)]$ is the prior probability accumulated in the cell from previous iterations of the procedure, which takes into account all previous data. $\Pr[i(t)|\xi, D(t-1)] = \Pr[i(t)|\xi]$ by making a Markov conditional independence assumption that implies that, given the *fact* of the existence or not of part of the object in the cell, the image obtained does not depend on previous images. Under this assumption, by using

$$\Pr[i(t)|\xi] = \frac{\Pr[\xi|i(t)] \Pr[i(t)]}{\Pr[\xi]}$$

we obtain

$$p(\xi) = \frac{\Pr[\xi|i(t)] \Pr[i(t)]}{\Pr[\xi]} \, \frac{\Pr[\xi|D(t-1)]}{\Pr[i(t)|D(t-1)]}$$

and

$$p(\overline{\xi}) = \frac{(1 - \Pr[\xi|i(t)]) \ \Pr[i(t)]}{1 - \Pr[\xi]} \, \frac{1 - \Pr[\xi|D(t-1)]}{\Pr[i(t)|D(t-1)]}$$

where $\overline{\xi}$ is the complement of event ξ. $\Pr[\xi|i(t)]$ is the probability estimate returned by the decision tree for the tile corresponding to the cell at (x, y, z) by only taking the current image into account. In estimation problems of this type, it is common practice to compute the *odds-ratio*, for which $\Pr[i(t)]$ and $\Pr[i(t)|D(t-1)]$ cancel out:

$$\begin{aligned} \textit{odds-ratio}(\xi) &\equiv \frac{p(\xi)}{p(\overline{\xi})} \\ &= \frac{\Pr[\xi|i(t)]}{1 - \Pr[\xi|i(t)]} \, \frac{\Pr[\xi|D(t-1)]}{1 - \Pr[\xi|D(t-1)]} \, \frac{1 - \Pr[\xi]}{\Pr[\xi]} \Rightarrow \\ p(\xi) &= \frac{\textit{odds-ratio}(\xi)}{1 + \textit{odds-ratio}(\xi)}. \end{aligned}$$

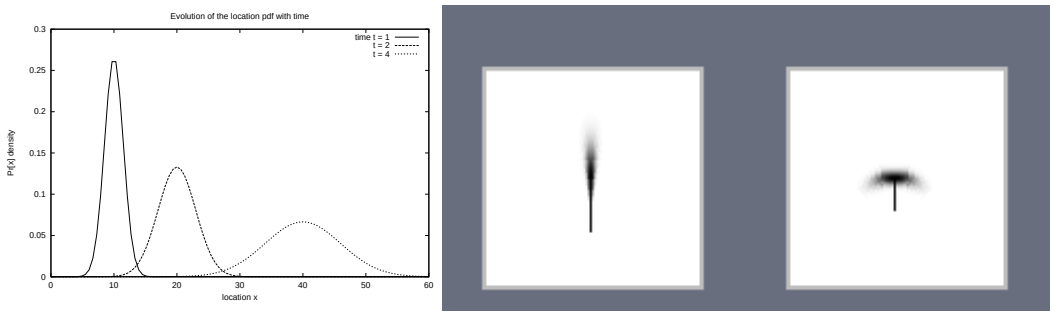

Figure 4: Probabilistic model of robot motion. Left image: Belief of the location of the object deteriorates in time under uncertainty of the magnitude of the velocity. Here $v \sim N(10, 1^2)$. Right image: This graph illustrates the outcome of specific motion commands projected along the z axis.

Similar formulas for belief integration can be found in [Pea88, Thr98b].

3.2 Robot Motion

Each robot motion introduces uncertainty into the robot's estimate of the object's location because of imperfect actuators and measuring devices. We model the translational as well as rotational magnitude of the velocity of the robot as a Gaussian random variable with mean equal to the nominal velocity given to the robotic motion controller—we make the assumption that there are no systematic errors. The standard deviations used are pessimistic estimates of the deviation around the nominal corresponding velocity magnitude. The accurate determination of the standard deviations does not significantly influence our location estimates given frequent enough observations. Under this assumption, their actual value is not critical and can be overestimated.

If the magnitude of the velocity is normally distributed with mean v_0 and standard deviation σ_v, $v \sim N(v_0, \sigma_v^2)$ (assume one-dimensional for the purpose of this example), the location of a object with that velocity after time t is a random variable $x \sim N(v_0 t, \sigma_v^2 t^2)$, also normally distributed, with mean $v_0 t$ and standard deviation $\sigma_v t$. This suggests that uncertainty of an objects location increases with time as time goes by, as shown in Fig. 4. Once a measurement is taken, the probabilities of the affected cells are updated in the manner described in section 3.1.

4 Experimental Results

We conducted our experiments on a B21 mobile robot equipped with a single Sony XC-999 color camera with a 6mm focal length lens, mounted on a pan-tilt unit. Images of size 240×256 are acquired through a Matrox Meteor framegrabber con-

nected to the camera and are used to train a decision tree in the manner described in section 2.1.

We chose a simple histogram representation of down-sampled versions of the training images as the input features to our decision tree algorithm. In particular, we use color histograms for each tile, at resolution of 256 color bins. Therefore each tile represents an example of 256 input features, namely the pixel percentages at each color bin, and one binary-valued output, corresponding to the event that the tile is part of the object being trained on. Even though this choice of input features does not take into account all information present in the picture, this is simply an artifact of the current implementation and by no means imposes any restriction on the choice of input features of the approach in general. More complex features may be employed in future implementations. However, as we demonstrate below, this simple representation performs adequately well in certain frequently occurring situations where the object is sufficiently distinct from the background, containing enough information for recovering the approximate location of simple objects in 3D. The "distinctiveness" is determined by the resolution of our color histogram, coupled with the amount of hue variation that changes in light intensity on the object result in.

An example application of a decision tree trained on three examples with an object (in this case, a bottle) and 18 background images (containing negative examples only) is shown in Fig. 2. The top few nodes of the tree are shown in Fig. 1. In a similar fashion we constructed a decision tree to recognize a larger simple object, a red chair, by using the same negative all-example images and three additional images containing the chair in different poses. We then manually maneuvered the mobile robot around the chair taking 7 new pictures from different angles. These pictures are shown in Fig. 5. The second column in that figure depicts the probability map that is output from the decision tree for each image. At certain locations we acquired images and projected the probability map in 3D, with each probability map element corresponding to a pyramid, as described in 3.1. Every cell covered by a pyramid is affected by the corresponding probability in the probability map. The results of projection when viewed along the x, y and z axes are shown in the three rightmost columns in Fig. 5. Each pixel in these projections has intensity proportional to the average probability along the axis of projection passing from that pixel. The z-axis projections make the locations around the chair that the pictures were taken particularly easy to see.

In reality, the robot does not keep a grid for each image but rather incorporates information incrementally in the single grid it maintains, which is justified under the Markov assumption. This is done by applying Bayes law for each cell individually.

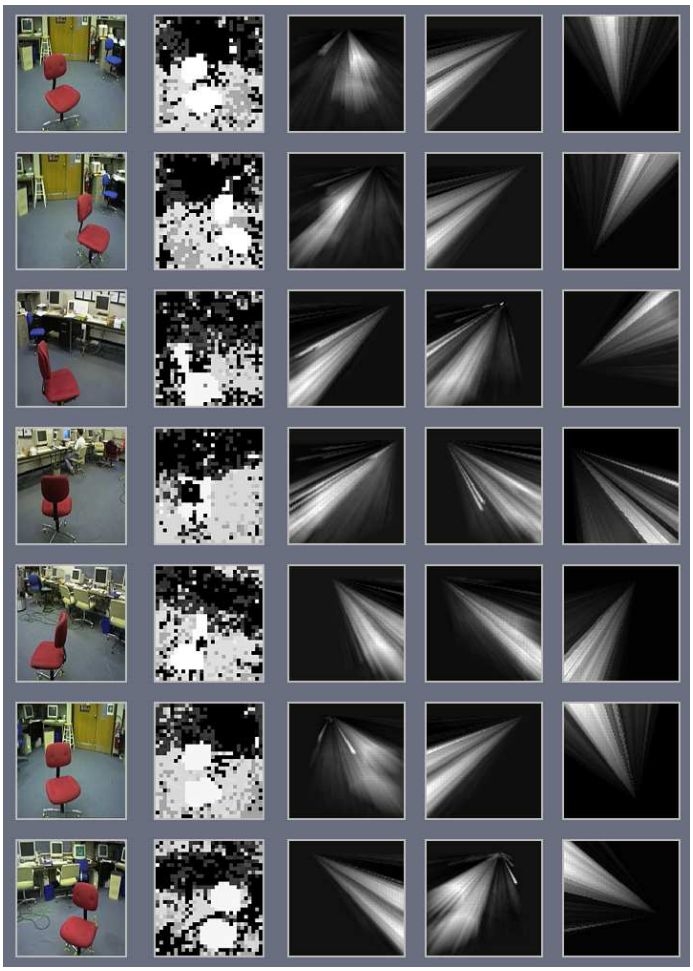

Figure 5: Probability map that is the output of the decision tree trained to recognize the red chair. The brightest tiles in the probability map (second column) correspond to probability greater than 0.9. Projection of the map in 3D are shown in the last three columns, as averages along the x, y and z (rightmost column) axis respectively.

There is no normalization done over the whole grid, which corresponds to the semantics we assign to the probability stored at each cell: it represent the probability that a part of the object occupies that cell. As such, we make no assumptions about the size of the object with respect to the cell size.

Between images, the robot is maneuvered manually to the spot where the next image will be taken. These motions increase our uncertainty in the manner described in section 3.2. The robot used in the experiments is a semi-holonomic one, its

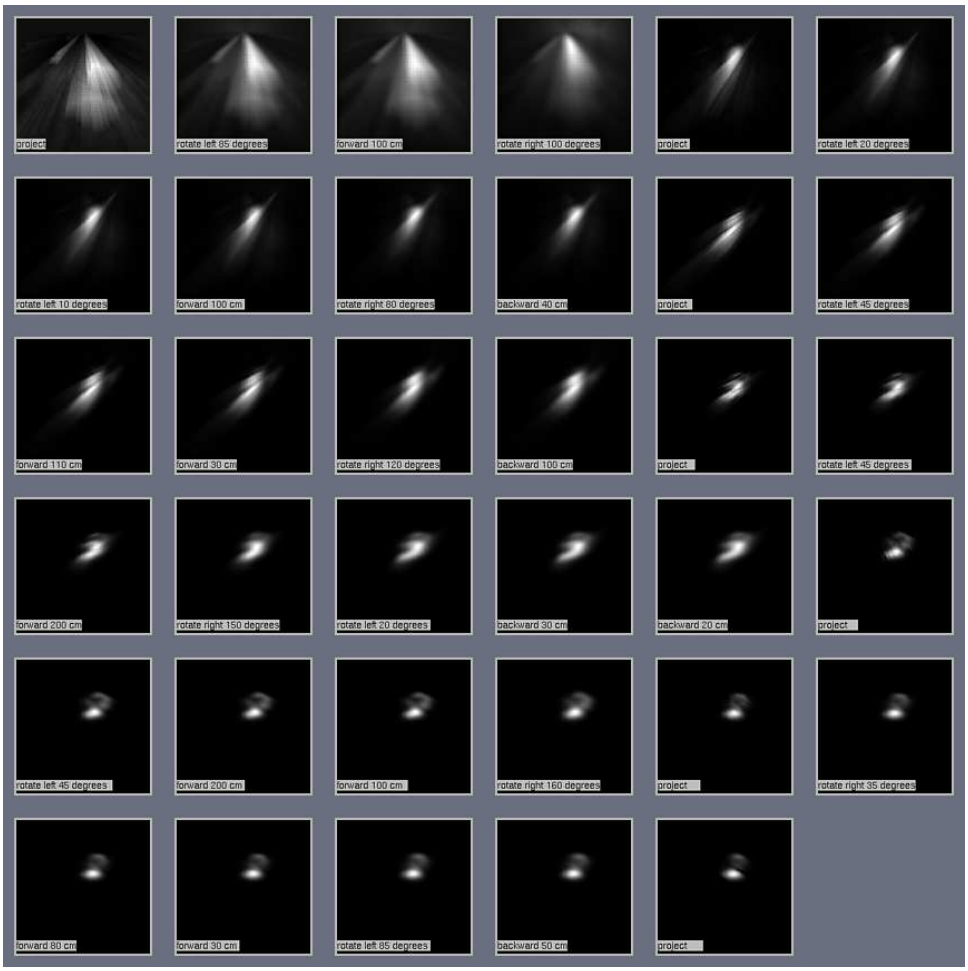

Figure 6: Cumulative effects on motion and probability map projection on grid as viewed along the x-axis (that is running perpendicular from the door facing the interior of the room in the pictures in Fig. 5). The two distinct parts of the chair (back and seat) are discernible.

motion consisting of rotations and forward or backward motions in the direction it is facing. As such we model rotational and translational uncertainty in the magnitude of the velocity.

The updating of the grid using the above procedure is shown in Fig. 6 for one run. This sequence of beliefs corresponds to a situation where a robot faces a chair. The grid size used is $100 \times 100 \times 100$ and each unit along any direction corresponds to 4cm in the real world. All beliefs in Fig. 6 are projected horizontally.

As can be seen in Fig. 6, the initial location of the target object(s) is unknown.

After taking a first image, the robot's belief is a conjunction of pyramids, corresponding to the output of the decision tree. As the robot moves, it loses information. As it takes the second snapshot from a different perspective, the belief is refined. After taking seven images, the location and the shape of the target object are reconstructed with high accuracy. As these results demonstrate, our approach can accurately determine the location of the target object. It is also robust to errors in the robot's odometry. This robustness is a result of incorporating our probabilistic model of robot motion.

5 Related Work

Decision trees [Qui86, Qui93, Mit97] are one of the most popular inductive machine learning method to date. The early algorithms were only applicable to problems with discrete input and output spaces. Decision tree learning algorithms for real-valued input spaces were proposed in [FI93, MKS94]. Tree-based regression methods for real-valued input and output spaces can be found in [Fri91, Moo90]. The work presented in this paper provides an example where a decision tree is used to learn a conditional probability density function. Like the approaches presented in [FI93, MKS94], it partitions a real-valued high-dimensional input space into hypercubes. The output nodes, however, represent conditional densities, which are estimated using a frequentist approach [CB90]. This is related to results reported in [TLS89, Mac92, Mit97], which show that under appropriate assumptions, artificial neural networks approximate conditional probability density functions.

The mathematical approach for integrating information is adopted from the statistical literature [CB90, Pea88]. The approach presented in this paper also bears close resemblance to occupancy grids [Mor88, Elf89]. Occupancy grid approaches are popular techniques for learning models of mobile robot environments from sensor data. Just like the approach proposed here, they represent the environment using fine-grained, evenly spaced grids. Each grid point is annotated by a probability, which describes the evidence that a location contains an object/obstacle. The vast majority of existing approaches differs from the one proposed here in three aspects. First, they model occupancy, not the location of a specific target object. Second they are usually constructed from range measurements (e.g., sonar, laser), not from camera images. Third, they are usually two-dimensional. There are, however, notable exceptions. The approaches described in [MM94, TBB$^+$98] construct occupancy grids from sequences of camera images. Moravec and Martin's approach [MM94] has probably been the first to construct 3D grids, instead of the commonly used 2D representations. Both approaches, however, used stereo vision to estimate the loca-

tion of obstacles. Stereo vision generates distance estimates, which greatly facilitates the construction of the maps. The approach reported here estimates distance indirectly, through integrating multiple camera images recorded at different locations. Unfortunately, the approach in [MM94] is incapable of dealing with error in the robot's odometry.

Our approach is similar to Markov localization [BFHS96, NPB95, SK95, Thr98a], a method for probabilistically estimating the pose of a mobile robot in a (known) environment. Markov localization relies on the same statistical principles for integrating multiple sensor readings into a single belief. In fact, the approach in [BFHS96] uses the same basic representations as our approach: evenly spaced grids. Markov localization, however, rests on the assumption that there is exactly one object (*i.e.*, the robot) whose location is to be estimated. Our approach can handle situations that contain a variable (unknown) number of target objects.

Finally, the problem of finding and manipulating objects has received considerable attention within the AI community (see [Hor94] and various papers in [Sim95, KBM98]). For example, Buhmann et al. [BBC+95] described an approach where a robot could be trained to recognize specific objects. Most existing approaches in the mobile robot community, however, make the assumption that the object is located in floor-height, in which case camera coordinates can directly be converted to real-world coordinates. Our approach is specifically designed to find objects at arbitrary locations in space. This is important in many real-world applications, as objects may frequently be found in tables, chairs, etc.

6 Discussion and Future Research

This paper presented a novel approach to estimating the 3D location of an object with a mobile robot. Individual camera images are interpreted using a decision tree method, which maps image regions (tiles) into probabilistic estimates for the presence of target objects. Based on a straightforward geometric consideration, these probabilities are mapped into 3D pyramids in global world coordinates. Multiple pyramids, obtained from camera images recorded from different viewpoints, are integrated using Bayes rule into a single probabilistic model of the object location. Noise in robot motion is accounted for by a probabilistic model of robot motion. Experimental results demonstrate that the method can robustly localize objects in 3D space.

A key advantage of the current approach is its generality. No assumption is made concerning the typical location of objects (e.g., they are not assumed to lie on the floor). The approach can also be trained easily to recognize new, user-specified

objects. While our current implementation uses color as the primary cue for object recognition, the method can equally be applied to a much richer range of image features, making it fit for a large class of target objects (i.e., objects that can be recognized from local image features).

Our approach rests on several limiting assumptions. First of all, it assumes that object does not move. To accommodate moving objects, our approach would have to be extended by a probabilistic model of object motion. Such a model might characterize the *typical* motion speed of the target object. It is unclear, however, if such an approach would be able to gather sufficient information to estimate the location of a moving object with the necessary accuracy.

Our approach also assumes that the training images accurately represent the situation during testing. In our experiments, we usually enriched the training set by a small number of pictures recorded at random locations in our lab. These pictures were used as negative training examples when growing the tree. We found that these additional images increased the robustness of the image analysis, thereby improving the overall estimation results. However, the method might fail if the robot encounters an object which similar to the target object, but which has not been part of its training set.

The spatial resolution in the experiments described in this paper is low, due the enormous complexity involved in updating 3D grids. By choosing a 4cm resolution, the computational overhead was manageable. Denser and larger grids are desirable, but unfortunately the computational cost of of updating the grid is cubic in the number of grid cells. An interesting extension of the current approach would be to use variable-resolution representations, such as oct-trees [Sam89b, Sam89a, Moo90], for representing object location. Such representations could balance the computational and memory resources, by modeling regions coarsely that are unlikely to contain a target object. If the density of target objects is low (which is usually the case), such an extension could improve the computational efficiency of the approach substantially.

Another promising extension of the current approach would be to devise methods that *actively* control the robot so as to maximize information gain. In the experiments presented here, a human manually positioned the robot. In our previous work [Thr98b], however, we already developed successful methods for active information gathering, which were applied in the context of learning 2D occupancy grid maps. In the context of object localization, such methods could lead to a behavior where a robot investigates the object from multiple viewpoints, in order to estimate its location accurately. The development of such methods is subject to future research.

References

[BBC+ 95] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The mobile robot Rhino. *AI Magazine*, 16(1), 1995.

[BFHS96] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, August 1996. AAAI, AAAI Press/MIT Press.

[CB90] G.C. Casella and R.L. Berger. *Statistical Inference*. Wadsworth & Brooks, Pacific Grove, CA, 1990.

[Elf89] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

[FI93] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of IJCAI-93*, Chamberry, France, July 1993. IJCAI, Inc.

[Fri91] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–141, March 1991.

[Hor94] I. Horswill. Specialization of perceptual processes. Technical Report AI TR-1511, MIT, AI Lab, Cambridge, MA, September 1994.

[KBM98] D. Kortenkamp, R.P. Bonassi, and R. Murphy, editors. *AI-based Mobile Robots: Case studies of successful robot systems*, Cambridge, MA, 1998. MIT Press. to appear.

[Mac92] D. J. C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, Pasadena, California, 1992.

[Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MKS94] S.K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–33, 1994.

[MM94] H.P. Moravec and M.C. Martin. Robot navigation by 3D spatial evidence grids. Mobile Robot Laboratory, Robotics Institute, Carnegie Mellon University, 1994.

[Moo90] A. W. Moore. *Efficient Memory-based Learning for Robot Control*. PhD thesis, Trinity Hall, University of Cambridge, England, 1990.

[Mor88] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[NPB95] I. Nourbakhsh, R. Powers, and S. Birchfield. DERVISH an office-navigating robot. *AI Magazine*, 16(2):53–60, Summer 1995.

[Pea88] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Sam89a] H. Samet. *Applications of Spatial Data Structures*. Addison-Wesley Publishing Inc., 1989.

[Sam89b] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Publishing Inc., 1989.

[Sim95] R. Simmons. The 1994 AAAI robot competition and exhibition. *AI Magazine*, 16(1), Spring 1995.

[SK95] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of IJCAI-95*, pages 1080–1087, Montreal, Canada, August 1995. IJCAI, Inc.

[TBB+ 98] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998. to appear.

[Thr98a] S. Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 1998. to appear.

[Thr98b] S. Thrun. Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, 1998. to appear.

[TLS89] N. Thishby, E. Levin, and S. A. Solla. Consistent inference of probabilities in layered networks: predictions and generalizations. In *Proceedings of the First International Joint Conference on Neural Networks, Washington, DC*, San Diego, 1989. IEEE TAB Neural Network Committee.