Wilking, D., Röfer, T. (2005). Real-time Object Recognition Using Decision Tree Learning. In: 8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences), Lecture Notes in Artificial Intelligence. Springer, im Erscheinen.

Realtime Object Recognition Using Decision Tree Learning*

Dirk Wilking[1] and Thomas Röfer[2]

[1] Chair for Computer Science XI, Embedded Software Group, RWTH Aachen
wilking@informatik.rwth-aachen.de
[2] Center for Computing Technology (TZI), Universität Bremen
roefer@tzi.de

Abstract. An object recognition process in general is designed as a domain specific, highly specialized task. As the complexity of such a process tends to be rather inestimable, machine learning is used to achieve better results in recognition. The goal of the process presented in this paper is the computation of the pose of a visible robot, i. e. the distance, angle, and orientation. The recognition process itself, the division into subtasks, as well as the results of the process are presented. The algorithms involved have been implemented and tested on a Sony Aibo.

1 Introduction

Computer vision is a multistage task. On the lowest level, the digitized image is searched for certain, domain-dependent features. The next step combines these features to more meaningful objects. Thereafter, a classification of the features computed so far must take place, but the concrete implementation concerning this classification is not as predefined as the low-level segmentation techniques.

* The Deutsche Forschungsgemeinschaft supports this work through the priority program "Cooperating teams of mobile robots in dynamic environments".

Fig. 1. Colored markers used on Sony Aibo robots.

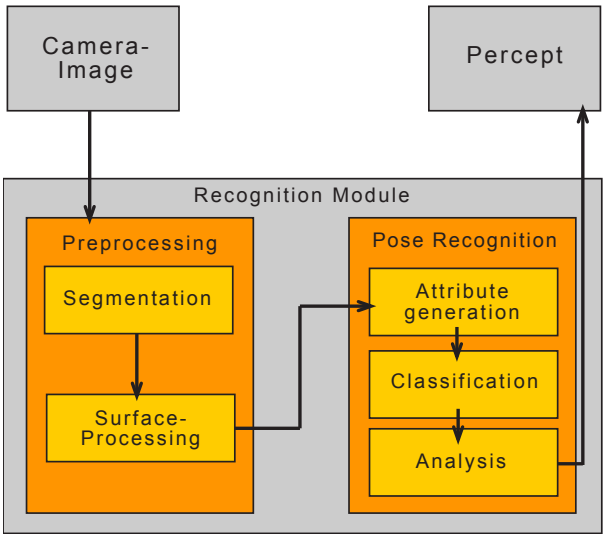

Fig. 2. Overview of the recognition module

In order to find an algorithmic base for the recognition of robots, the problem of pose determination is solved regarding an image of a single robot. The most obvious feature of a robot in the Sony Four-Legged Robot League (SFRL) are the colored markers placed on different parts of the body (cf. Fig. 1).

Apart from the unique color which can be used easily to find a robot in an image, the geometric shapes of the different parts provide much more information about the position of the robot. The shapes themselves can be approximated using simple line segments and the angles between them.

2 The Recognition Process

The framework GT2003 that is used for this work is provided by the GermanTeam ([2]). The process is embedded into a recognition module that receives the robot's current image as input and delivers the poses of the robots perceived on the field. As shown in Fig. 2, the preprocessing stage consists of the segmentation and surface generation. The pose recognition deals with the higher level functions of attribute generation, classification, and analysis of the symbols generated by the classification.

The recognition begins with iterating through the surfaces that have been discovered by the preprocessing stage. For every surface, a number of segments approximating its shape and a symbol is generated (e. g. head, side, front, back, leg, or nothing) as shown in Fig. 3. The symbols are inserted into a special 180° symbol memory which is shown in Fig. 3d) at a position that depends on the rotation of the head of the robot, e. g., a robot looking to the right will insert all

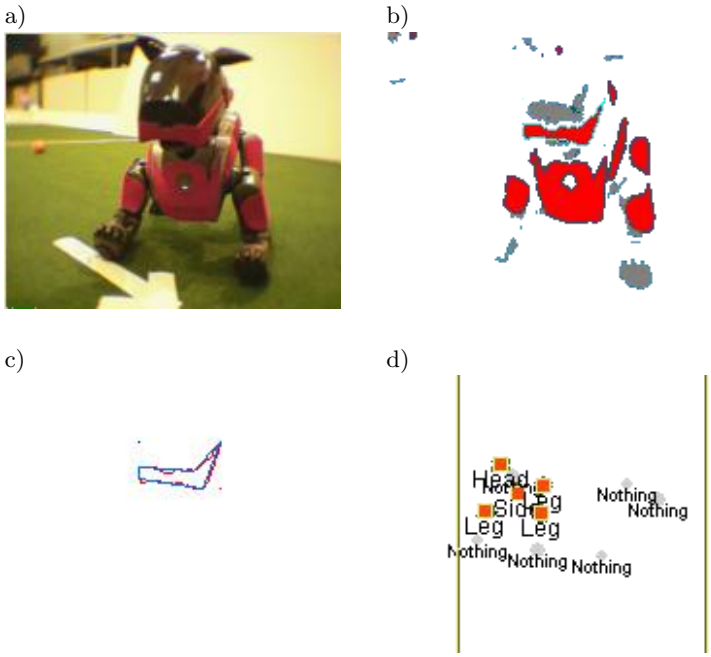

Fig. 3. From original image to symbol representation. a) Original image. b) Result of segmentation. c) Boundary of a singe segmented surface. d) The 180° symbol memory.

symbols on the right side of the memory. The analysis step that computes the real pose only uses information provided by the 180° memory.

2.1 Segmentation and Surface Detection

The segmentation of the original image is done only on pixels which are important for the recognition of robots. This is reflected by a grid that is used to process the image. The idea is that only pixels on the grid are tested and if they belong to a relevant color class, the underlying, more complex algorithms are started.

Relevant pixels are determined by color segmentation using color tables (see e.g. [1]). When a pixel is found to be relevant, a flood-fill algorithm is started. This way, surfaces—along with their position, bounding box, and area—are computed. Some of the information gathered during this step is used in the later processing stage of attribute generation.

The contour of the surface is computed with a simple contour following algorithm. The direction that is used to add pixels is changed in a clockwise fashion.

Having computed the contour of the surface, the generation of line segments is started. The iterative-end point algorithm as described in [6] is used to compute the segments.

2.2 Attribute generation

As an important part of machine learning, the choice of the attributes to be learned must be considered. Different kinds of attributes have been implemented. Simple attributes, as proposed in [3] and [4], comprise, e. g., color class, area, perimeter, and aspect ratio. Regarding the representation of the surface as line segments, the number of corners, the convexity and the number of different classes of angles between two line segments are provided as attributes. In addition, the surface is compared to a circle and a rectangle with the same area.The discretization of continuous values is achieved with a simple algorithm that searches for an optimal split value in a brute force manner.

A different approach to compute shape based attributes is presented in [5]. The underlying idea, described as conjunctions of local properties, consists of the merging of neighboring features as one single attribute. This is adopted as sequences of adjacent angles.

2.3 Classification

As classification algorithm, the decision tree learning algorithm (cf. [7]) is chosen. This algorithm creates a tree consisting of attributes as described above and symbols which form the leafs of the tree. The tree is built by calculating the attribute with the highest entropy which depends on the number of occurrences of different attribute values.

The problem of over-fitting is solved using χ^2-pruning. The basic idea behind χ^2-pruning is to determine whether a chosen attribute really detects an underlying pattern or not. For the probability that the attribute has no underlying pattern a value of 1% is assumed.

2.4 Analysis

The analysis starts with inserting symbols into the 180° memory (cf. Fig. 4). Using this memory, groups of symbols representing a robot each are combined using the following heuristics:

- Find a side, front, or back symbol in the 180° memory. This symbol is the first symbol.
- Add all head symbols that are near to the first symbol. Test the distance, size, and relative position of the head.
- Add all leg symbols and perform the same tests on them.
- Add other body related symbols and perform the same tests on them.

The surface area of a group is used to determine the distance to the robot. The direction to the robot is computed by the group's position in the 180° memory. The relative position of the head within the group and the existence of front or back symbols indicate the rough direction of the robot. A more precise value is calculated using the aspect ratio of the initial symbol.

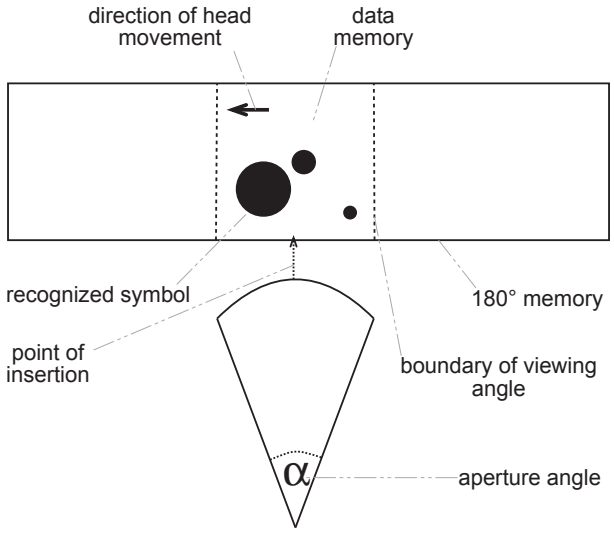

Fig. 4. Structure of the 180° memory used to process symbols

3 Results

The following results have been achieved with an opponent robot as recognition target under the normal lighting conditions of the SFRL.

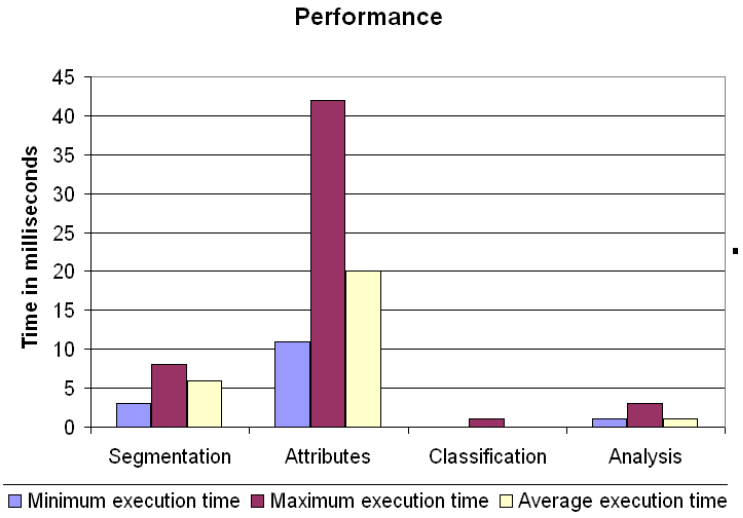

Fig. 5. Performance results for different subtasks

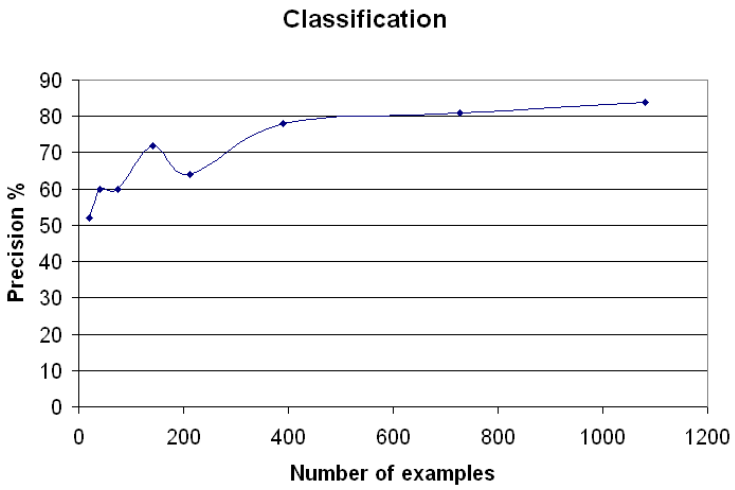

Fig. 6. Classification precision of the recognition process

The first question concerns the overall speed of the process. With an average execution time of 27 milliseconds, the usage on the Aibo robots of the SFRL is possible. Fig. 5 shows that the slowest subtask is the generation of attributes. The reason for that is mainly the usage of the iterative-end point algorithm which consumes much of the execution time using many floating point operations. In

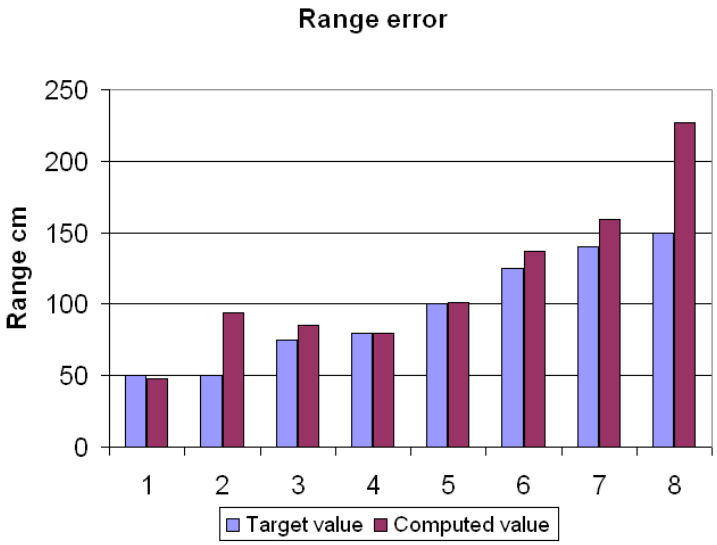

Fig. 7. Precision of the calculation of the range

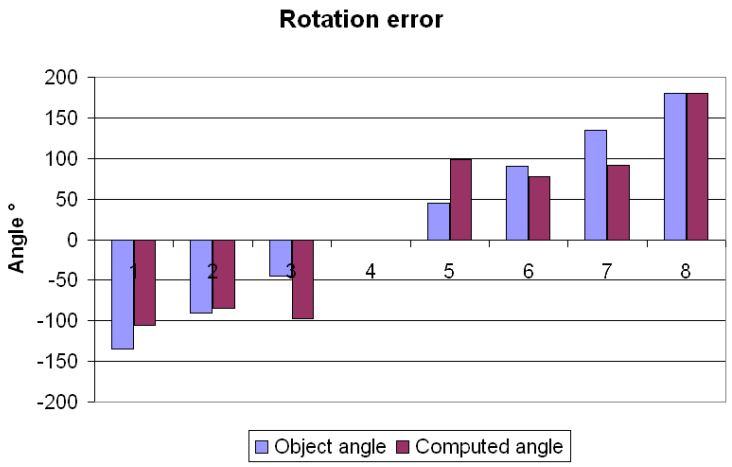

Fig. 8. Precision of the calculation of the rotation

contrast, the decision tree classification as well as the analysis of symbol groups are quite fast. Although only started when required and driven by a flood fill algorithm, the segmentation constantly uses about a fifth of the execution time. The decision tree classification has a precision of 84 % using 1080 examples to classify 5 classes (cf. Fig. 6). The importance of the number of examples decreases fast. The difference in precision between 730 and 1080 examples is 2 %. Even with the low number of 25 examples a classification precision of over 50 % is reached.

To determine the error of the range and rotation calculation, eight different positions have been measured about ten times each. The averaged results are presented in Figures 7 and 8. The second and the last case of the range test are the only calculations associated with a high error compared to the original value. The reason were wrong classified symbols. The calculation of the robot's rotation suffered even more from that problem.

In addition, the function that provided a more precise value for the rotation within a quadrant suffered from the problem that the aspect ratio of the initial symbol was coarse.

4 Conclusions

This paper presents a robot recognition process based on decision tree classification. Starting with preprocessing tasks such as color class table based segmentation, contour detection, and line segment generation, a computation of different attributes (e. g. surface area, perimeter, neighboring angles) is presented. The decision tree classification and the related χ^2 pruning are discussed in greater detail. The analysis of the resulting symbols uses a 180° short-term memory for combination purposes. Then, symbols are grouped together based on a heuristics

that makes extensive use of the structure of the target objects. Finally, symbol groups are processed to get the distance and direction to the robot. The rotation is computed again based on the structure of symbols and the aspect ratio of the main surface.

However, due to the complexity and length of the process, some parts could be streamlined. Especially the heuristics used during the analysis step can be improved using a skeleton template based, probabilistic matching procedure. This procedure could deal both with the problem of occlusion and missing symbols. In addition, improvements concerning the speed of the attribute generation can be achieved. Considering the process architecture, the dependence of the recognition tasks among each other must be examined. Especially the classification task could benefit from a slowly, rather parallel[3] growing recognition process that provides expected values for the classification.

References

1. J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 00), volume 3, pages 2061–2066, 2000.
2. Burkhard, H.-D., Düffert, U., Jüngel, M., Lötzsch, M., Koschmieder, N., Laue, T., Röfer, T., Spiess, K., Sztybryc, A., Brunn, R., Risler, M., v. Stryk, O. (2001). GermanTeam 2001. Technical report (41 pages, only available online).
3. Shepherd, B. A.. An apprasial of a Decision Tree approach to Image Classification. In Proc. of the Eighth International Joint Conference on Artificial Intelligence, pages 473–475, 1983.
4. Gerhard-Helge Pinz. Bildverstehen. Springer, 1994.
5. Cho, K., Dunn, S. M.. Learning Shape Classes. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 1994.
6. Duda, R. O., Hart, P. E.. Pattern Classification and Scene Analysis. Wiley, New York, 1972.
7. Quinlan, J. R.. C 4.5: Programs for Machine Learning. Morgan Kaufman Publishers, San Mateo, 1993.

[3] In contrast, the symbols are currently generated without any context knowledge.