Visual Servoing of a Miniature Robot Toward a Marked Target *

Patrick Rößler, Sascha A. Stoeter, Paul E. Rybski, Maria Gini, Nikolaos Papanikolopoulos
Center for Distributed Robotics, Department of Computer Science and Engineering
University of Minnesota, Minneapolis, U.S.A.
{roessler, stoeter, rybski, gini, npapas}@cs.umn.edu

Abstract. The paper describes a system that servos a miniature robot toward a marked target based on visual information from the robot's own camera. This task is divided into two subtasks: recognition of the target and driving toward it. The system deals with the problems introduced by noisy camera images and unstructured, non-static backgrounds. A description of the control software is given and experimental results are discussed.

1 Introduction

Team-based behaviors are an important part of a distributed robotic system. Cooperating robots can tackle tasks that are impossible to solve for a single robot. In this work, miniature robots called Scouts [5], are used. A Scout (Fig. 3) is capable of rolling with its wheels and jumping small heights with a spring foot mechanism.

To accomplish cooperating tasks, a Scout must be able to find other team members or any other previously known object of interest. As the Scouts are small and receive their energy from batteries, the choice of sensors for this task is limited. A CMOS cameras is an ideal sensor for a Scout: It is both small and economic in power consumption, and it can be used for visual servoing tasks (e.g., [2]).

In this work, we demonstrate that a Scout is capable of detecting a known static targets, another Scout, with its camera. This task is non-trivial because of the Scouts' size and their transparent shells. This structure makes it hard to detect the robot in realistic environments whose backgrounds are typically non-static. Worse yet, the camera's images are extremely noisy. A single marker is attached to the target Scout to simplify detection. The marker is a white square, 3.5 cm on a side, with a black circle of 2 cm in diameter in the middle. The small size of the Scout does not allow for attaching several markers whose positions could be used to estimate the target's relative location to the camera.

2 The System

The proposed method was integrated into the existing distributed control architecture [7]. This architecture has four main subsystems. The *User Interface* gives the user the possibility to interact with the system. The *Mission Control* hosts prioritized behaviors which model the solution for a mission. The *Resource Pool* manages the resource controllers that model all physical and logical resources. These subsystems are tied together by the *Backbone*.

Two main components had to be added: the behavior ServoToTargetB and the resource controller TargetFindRC (Fig. 1). While the behavior controls the servoing of the Scout and estimates the position of the target based on the information given by the resource controller, all image processing is done in TargetFindRC. The video frames are grabbed and processed. As those tasks are part of a distributed environment, it is possible that they run on different computers that may be connected through slow connections. Therefore, it is important to keep the communication overhead of the system low. TargetFindRC only transmits a list containing the likely positions of the target to ServoToTargetB.

2.1 The Resource Controller

TargetFindRC grabs incoming video frames from the Scout. After these frames are preprocessed, it tries to find likely positions of the target in the image. To reduce the influence of noise and the complexity of the search for the marker in the image, a two-step algorithm is used. The first step locates regions that have a high evidence of containing a Scout. The second step evaluates these regions to determine the actual position and size of the target.

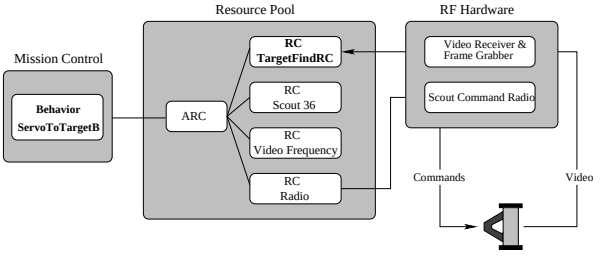


Figure 1: Overview of the system.

*This material is based upon work supported by the Defense Advanced Research Projects Agency, Microsystems Technology Office (Distributed Robotics), ARPA Order No. G155, Program Code No. 8H20, issued by DARPA/CMD under Contract #MDA972-98-C-0008.

2.1.1 Preprocessing

The images acquired from the Scout's camera are typically very noisy and have low contrast. In addition, interference of the electromagnetic motors with the transmitter affects the image quality significantly. From time to time, the connection is lost completely. These problems have to be addressed in the preprocessing step. Frames with a lost connection or a very weak signal are identified. These frames are not processed as they would have a negative impact on the performance of the system.

The image is smoothed to lower the influence of noise. As the marker may be very small, a filter that is too strong might smoothen the image so much that the marker cannot be found. Thus, a three-by-three pixel arithmetic mean filter is used. Then, to improve the contrast, a histogram equalization algorithm is applied. This will simplify the task of detecting edges especially in the region of the marker.

2.1.2 Binary Template Matching

A template matching algorithm is applied to find the target in the image [3]. The template matches the area of the original image with the highest cross-correlation. First an edge operator is applied and then the edge image is thresholded to obtain a binary image. The chosen edge operator is the three-by-three Sobel operator [6] as it is relatively fast and noise has little influence on it. A histogram of the edge image is computed in order to obtain the threshold. A threshold value is picked from the histogram that is greater than a given fraction of all gray-values in the image. In this case, a fraction of 95 % was empirically chosen. This leaves only the most significant edges in the image, while all others are deleted. The edge directions are computed for use in later steps.

To obtain the template, a picture of the target is taken in front of a uniformly colored background. The area containing the target is cropped manually and an edge image is created as described above.

Finding the best matching position of a gray-scale template in a gray-scale image can be done by maximizing the cross-correlation R_{ft} of the two images:

$$\arg\max_y R_{ft}(y) = \arg\max_y \sum_x f(x+y)t(x) \qquad (1)$$

where y is the displacement of the template in the image, x is the position in the template, $f(y)$ is the intensity of the image at position y, and $t(x)$ is the intensity of the template at position x.

For binary images with $f(y) \in \{0,1\}$ and $t(x) \in \{0,1\}$, this can be simplified to

$$\arg\max_y R_{ft}(y) = \arg\max_y \sum_{\{x|t(x)=1\}} f(x+y) \qquad (2)$$

enabling fast algorithms that only visit specific pixels set in both images. Though the complexity of convolving the edge template with the binary image is of the same order as before, it is considerably faster as only approximately 5 % of the pixels in both the image and the template have to be visited. Furthermore, the multiplication and summation of the gray-values of both images is simplified to counting the number of matching pixels.

An additional acceleration is gained by running this algorithm on down-scaled versions of the images. The images can be scaled down by a factor of three along both axes from their original size of 320×240 pixels with no considerable loss of accuracy.

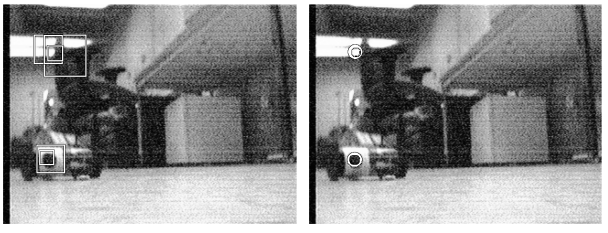
The estimated position of the target is relatively invariant against the size of the template. Thus, only four template sizes must be matched representing distances of approximately 30 cm, 40 cm, 60 cm, and 120 cm (Fig. 2). The areas with a high correlation to one of the templates are considered to be areas with a high probability to contain the target with a size of approximately the template size. These areas will be examined further in step 2.

Figure 2: The largest and the smallest templates.

2.1.3 Circle Detection

In the second step, the circular marker is searched for in the areas of high confidence found in step 1 (Fig. 3(a)). This task is also accomplished using the edge information of the image. Given the design of the marker, the edge has a high gradient value and is clearly visible in the thresholded edge image. An edge thinning algorithm is used on these areas to obtain edges that are exactly one pixel wide. A Hough Transform [4] is subsequently applied to find circles in the thinned search windows. Ballard proposes to incorporate the directional information of the edges found previously to increase the accuracy of the computation and to minimize the number of circles [1].

All search-windows are examined for three possible radii. Only circles with a minimum number of fitting edge pixels are kept. The minimum number is determined by a radius-dependent heuristic.



(a) Regions of interest (b) Detected circles

Figure 3: Results from the proposed algorithm.

The next task is to find the circle that matches the marker on the Scout. The circular marker is the only one or at least one of the best visible circles in the search windows. Due to noise and roughly circular obstacles in the background, usually more than one circle with the same number of matching pixels are found. Sometimes, the circle corresponding to the marker is not among those with highest number of matching pixels but it still has a high number. For this reason, all circles with the two highest numbers of fitting pixels are passed back to the behavior for further processing. Fig. 3(b) shows these circles.

2.2 The Behavior

Servoing toward the target and keeping track of its actual position is accomplished by the behavior ServoToTargetB. The target's position is extracted from the list of likely positions. A time-dependent confidence-driven algorithm is introduced to solve this task. A simple fuzzy controller servos the Scout toward the target while the estimation for the target position is constantly updated.

2.2.1 Estimating the Target Position

The behavior has to estimate the actual target position from the list of likely positions computed by the RC. If there actually is a target visible in the processed frame, the possibility that the position of the marker is among the detected circles is very high. The other detected circles are induced by noise and roughly circular objects in the background. If, however, the target is not visible in the image, only the aforementioned false positives are found. They can be filtered out easily as their positions in subsequent frames is almost completely random.

A system of two confidence values is used. The value *confidence* displays the confidence of the algorithm to follow the same target in subsequent frames whereas *doubt* measures the confidence that the target is not visible in the examined frame. The image frame is divided into nine rectangular regions (three rows and three columns) to determine the similarity between two likely target positions. If the Scout did not move between frames, the targets are assumed to be similar if their centers are in the same region and the difference of their circle radii is small. Easier to meet criteria are used after the Scout has moved as the Scouts' movements are very inaccurate. Besides the similarity of the radii, the circles only have to be in the same row in the image. Fig. 4 demonstrates the update process of the two confidence values.

2.2.2 Servoing

A simple fuzzy control mechanism is used for servoing the Scout. The image is divided into three columns that are used to specify the relative position of the target and to select the appropriate drive command. The three drive commands *turnLeft*, *turnRight*, and *driveForward* are used to approach the target. The fourth command *explore* examines unknown parts of the environment.

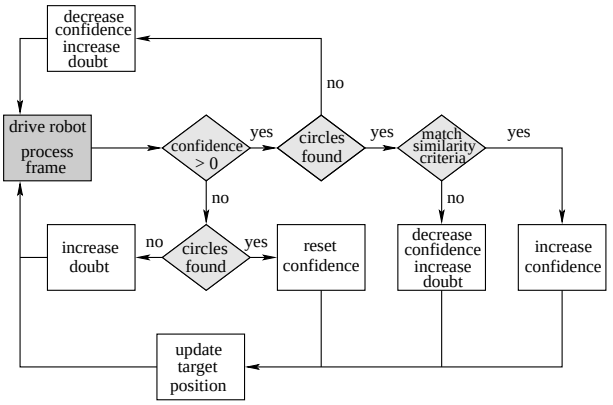


Figure 4: Scheme for updating the target position and the confidence values.

In order to approach the target, *confidence* has to exceed a threshold. If the target is located in one of the outer regions, the Scout tries to center it by turning slightly toward the appropriate direction (*turnLeft* or *turnRight*). While doing this, the expected target area is adapted. If the target is already centered and still far away, the Scout approaches it by using the *driveForward* command.

If the confidence value *doubt* exceeds a threshold, no target is visible in this frame and the *explore* command is executed. This is a relatively wide turn to the left so that the Scout can observe a previously invisible part of the environment. Both confidence values are reset to zero before processing the next frame.

3 Experimental Results

The performance of the proposed approach was evaluated with a set of experiments. A set of five experiments each with a total of 30 runs was conducted to test the different abilities of the software (Fig. 5). All experiments were run in a real lab environment. The system is supposed to recognize the target, approach it to a distance of 30 cm, and then stop.

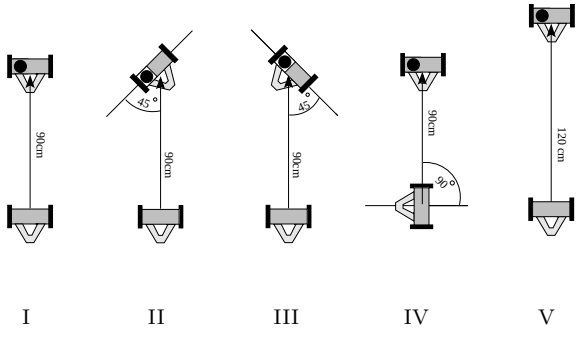


Figure 5: Setup for the experiments I–V.

Both the ability to locate the target and the accuracy of distance estimations were tested. This divides the results into four classes. Class A contains only completely successful experiments. The runs in which the target is found and approached, but in which the Scout does not stop, define class B. These runs are considered partially successful. Class C includes all cases in which the target is located, but its estimated distance is too small resulting in premature stops. Class D holds the remaining runs.

Table 1 shows a summary of the experimental results. It can be seen that the Scout is recognized in most runs. Experiments II and III show that the system can handle tilt well. The exploration strategy is verified by experiment IV. Experiment V shows that the target is recognized even for the extreme distance of 120 cm. The results of the different classes are equally distributed among the five experiments.

class	result	occurrences
A	found target and stopped	10
B	found target, did not stop	16
C	stopped too early	2
D	complete failure	2

Table 1: Summary of the experimental results.

In an average run of Class A or B the scout has to process between 20 and 30 frames until the target is reached. The maximum number of processed frames was 140. In this run the scout had to do three complete exploration-turns because the target was lost several times. In most runs however the scout did not have to do any complete turn to find its target.

Among the results is a noticeable number of cases of class B. The reason for this behavior is a tendency to underestimate the size of the target while approaching it. The Hough Transform usually finds several circles that are slightly smaller or bigger than the real one. The algorithm picks the estimation for the circular marker that is the most similar to the one found in the previous frame. But as the Scout approaches the target, the size of the real circle in the image grows only slowly and the algorithm underestimates the size of the target.

4 Conclusion

The experiments show that the target is found in distances from 30 cm to 120 cm and that the Scout manages to servo toward it. The method works in unstructured non-static backgrounds and is invariant against tilt of the target. The exploration strategy ensures finding the target even if the Scout is initially not facing the target or if it is lost during the approach.

Two problems that became obvious during the experiments are underestimating the size of the target while approaching it and computation speed. The experiments were run on a Pentium II with 450 MHz resulting in a processing speed of just one frame per second. Computation time could be significantly reduced by using more up-to-date hardware or running parts of the algorithms on specialized DSP boards.

5 Future Work

Future work aims at improving the existing system. With a perfected termination criterion, the experimental results could be enhanced significantly.

At present, every frame is completely searched for every scale of the template. The search space could be reduced significantly. After the target is found once, only a part of the image could be examined with just the appropriate template depending on the size of the target found in the last frame.

It would be very interesting to find out if the various parameters and thresholds used in the system can be estimated during runtime. This would improve its ability to adapt to different environments.

An important task is generalizing the system to find a wider spectrum of targets without using an active marker and to recognize and follow moving targets.

References

[1] D. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] P. I. Corke. *Visual Control of Robots: High Performance Visual Servoing.* Research Studies Press, 1996.

[3] E. Huang, W. Yau, and L. Fu. An edge based visual tracking for target within complex environment. In *Proc. of the American Control Conference*, pages 1993–1997, Chicago, IL, U.S.A., June 2000.

[4] J. Illingsworth and J. Kittler. A survey of the hough transform. *Computer Vision, Graphics & Image Processing*, 44(1):87–116, 1988.

[5] P. E. Rybski, N. P. Papanikolopoulos, S. A. Stoeter, D. G. Krantz, K. B. Yesin, M. Gini, R. Voyles, D. F. Hougen, B. Nelson, and M. D. Erickson. Enlisting rangers and scouts for reconnaissance and surveillance. *IEEE Robotics and Automation Magazine*, 7(4):14–24, Dec. 2000.

[6] I. Sobel. *Camera Models and Machine Perception.* PhD thesis, Stanford University, U.S.A., 1970.

[7] S. A. Stoeter, P. E. Rybski, M. Gini, D. F. Hougen, and N. P. Papanikolopoulos. Verteilte Steuerung heterogener mobiler Roboter. In *Proc. of Autonome Mobile Systeme*, Informatik aktuell, pages 270–277, Karlsruhe, Germany, Nov. 2000. Gesellschaft für Informatik, Springer.