Active target tracking using a mobile robot in the USARSim

Aksel Ethembabaoglu
0154644
aethemba@science.uva.nl

June 29, 2007

Supervised by Arnoud Visser

in partial fulfillment of the requirements

for the degree of

Bachelor of Science in Artificial Intelligence

FACULTEIT DER NATUURWETENSCHAPPEN, WISKUNDE EN INFORMATICA

Abstract

In this thesis the Mean-shift tracker is analyzed by quantitative measurements in the USARSim. The Mean-Shift tracker is combined with a basic PD control structure for active tracking. The aim is to expose the limitations of the tracker in the simulation and to suggest improvements for future work.
The results of the experiments indicate that the Mean-Shift tracker is not robust enough as a reliable tracker in the simulation.

Contents

1 Introduction

This thesis describes the graduation project for the Bachelor of Science in Artificial Intelligence (AI) at the University of Amsterdam (UvA). It comprises four weeks in which the student is required to work full-time on an AI related project. For this project I contacted Arnoud Visser who supervises the RoboCup Rescue team of the UvA. During this project I focused on the analysis of the Mean-Shift tracking algorithm.

1.1 Research goals

The primary goal of the research is to quantitatively test the usability and the limitations of the Mean-Shift tracking algorithm in a controlled simulation environment. This thesis will be concluded with suggestions to improve the algorithm.
A long term goal of the project is the analysis of more sophisticated tracking algorithms in a controlled environment. The controlled testing environment provides a proving ground for new developed algorithms and their use in the RoboCup.

1.2 Thesis organization

The organization of this thesis will now briefly be discussed. In the following section the relevance of the research is illustrated by discussing the RoboCup Rescue competition. Next, various aspects of the Mean-Shift tracker will be discussed. This includes alternative algorithms, pseudo-code and a mathematical model. In the section thereafter the system architecture of the simulation will be described. In section five the experiments and the results are discussed. Finally, the thesis is concluded with suggestions on future work.

2 RoboCup Rescue

In the RoboCup Rescue virtual robot competition, robots are placed in an unknown environment where they need to locate victims of a disaster. The environments come in three difficulties related to a certain challenge. For the Victim Detection challenge the Yellow, Orange and Red Arena represent environments in which victims can be located with increasing difficulty. In the Red Arena, victims could be moving around. This is were the tracking algorithm becomes useful. More information on RoboCup Rescue can be found at: http://www.robocuprescue.org/

3 Tracking a human

The ability to track a human being is valuable in a number of applications besides the RoboCup Rescue. Human-computer interaction, surveillance, military purposes and medical applications are examples of areas where this ability proves useful. The main tasks in tracking a human are, distinguishing moving objects from static objects and differentiating between target objects and other moving objects. To track successfully, it is vital that the tracker is robust, i.e. it should be able to operate under a variety of conditions. Influential conditions include, among other things, target distance and lighting. These conditions are not always easy to reproduce which makes the proper testing of algorithms a daunting task.
A variety of algorithms have been developed to track a human. These will be discussed in the following subsection.

3.1 A basic description of the Mean-Shift algorithm

The basic idea of the Mean-Shift algorithm is very simple. Given a certain color distribution (histogram)of a target, find, in the image, the most similar color distribution within a certain window. This window will be referred to as a kernel or region of interest. This kernel is located over an image to track, in the original image, similar color distributions as provided by the target histogram. The Mean-Shift algorithm creates a back projection image in which pixels of the image that are similar to the target histogram are colored white. This produces a binary image with white clouds or blobs of white pixels. The *mean* is given as the location in the kernel from which the average distance to all white pixels is minimal. When the target model moves, this blob, or cloud, of pixels is translated in the backprojection of the next frame. This will yield a new location of the mean. The algorithm tracks a target by placing the center of the kernel on the new mean.
The algorithm can be enhanced by performing a number of mean-shift iterations. After an initial new mean has been found, rather then moving the kernel immediately, the algorithm again performs a new mean calculation based on the previously calculated mean location. Eventually, the shifting of a new mean location will be below a certain threshold indicating the algorithm to terminate. Alternatively, a maximum number of iterations can be given. This method will allow the algorithm to track faster moving objects because it is able to track clouds of pixels that have shifted farther away, it comes however at a

computational cost. This suggested improvement proved insignificant in our implementation.
The pseudo-code for the Mean-Shift algorithm is depicted in algorithm 1.

Algorithm 1 Calculate Mean-Shift

1: Initialize color-model of a target object using a kernel over the image

2: Compute the color-model of the target object on the current center position of the target

3: Match color-model of target with pixel values of image in the kernel

4: Calculate movement, by calculating the new mean, of the target object and correct current center of kernel

5: Optionally, repeat step 2 to 4, until movement of new kernel location falls below a certain threshold

6: Continue on next frame

3.2 Alternative tracking algorithms

The Mean-Shift tracker is a kernel based tracking algorithm. It tracks objects on an image by use of a kernel. Tracking a color distribution of a target over a series of frames is not the only tracking approach.
An alternative is the feature point tracker [1]. Here, feature points are detected on an image and these are tracked over a seriess of images.
Yet another alternative is the Kalman filter [2] which estimates a targets position and velocity based given previous observations. It can also be used for control functions.
The Mean-Shift tracker was chosen as a suitable candidate for analysis because of the relative simple and efficient design. It was also expected to provide a robust performance when a distinctive histogram of the target object was given.

3.3 Mean Shift tracking - A mathematical model

In the next subsections the Mean-Shift algorithm described earlier will be mathematically formalized.
Mean-shift tracking is a kernel based tracking algorithm. It is designed to track a target by masking it spatially with an isotropic kernel, defining a similarity function, and then searching the basin of attraction of this function to perform target localization. Because of the similarity function, gradient descent searching can be performed to localize the target. The Bhattacharyya coefficient is used to determine the similarity between the target model and the target candidate in the next frame.

3.3.1 Target representation

To track an object, a target first has to be defined and for this a feature space needs to be chosen. Next, the *target model* is then identified by its probability density function q in the feature space. For this tracker the color feature of the target was used. The target is considered to reside on spatial location y_0. In the next frame, a target candidate is defined at location y_1. This target candidate is characterized by the color distribution $p(y_1)$. The color distributions are computed using m-bin histograms. Histograms are used to comply to low computational costs for real-time processing.
This can be denoted as follows:

Target model: $\hat{q} = \{\hat{q}_u\}_{u=1...m}$ with $\sum_{u=1}^{m} \hat{q_u} = 1$

Target candidate: $\hat{p}(y_1) = \{\hat{p}_u(y_1)\}_{u=1...m}$ with $\sum_{u=1}^{m} \hat{p_u(y_1)} = 1$

We will further denote:

$$\hat{p}(y) = \rho[\hat{p}(y), \hat{q}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y)\hat{q}_u} \tag{1}$$

Equation 1 denotes a similarity function between $\hat{p}$ and $\hat{q}$. Here, $\hat{p}(y)$ has the role of a likelihood and the local maxima of this function indicates the presence of an object in the second frame that has a similar representation as $\hat{q}$, as defined in the predefined histogram.
The similarity function is regularized by masking the object with an isotropic kernel in the spatial domain. When the kernel weights which carry continuous spatial information, are used $\hat{p}(y)$ becomes a smooth function in y.

3.3.2 Target model

Let $\{x_i^*\}_{i=1...n}$ be the *normalized* pixel locations in the region defined as the target model. The region is centered at location 0. An isotropic kernel is used with a convex and a monotonic decreasing kernel profile $k(x)$. In our case a Gaussian kernel is used. The reliability of the density function is now increased because peripheral pixels are considered less reliable. They are considered less reliable because they are more vulnerable to background noise and clutter.
The function $b : R^2 \rightarrow \{1...m\}$ associates to the pixel at location x_i^* the index $b(x_i^*)$ of its bin in the quantized feature space. The probability of the feature, $u = 1...m$, in the target model is then computed as:

$$\hat{q}_u(y) = C \sum_{i=1}^{n} k(\|x_i^*\|^2)\delta[b(x_i^*) - u] \tag{2}$$

where δ is the Kronecker delta function. The normalization constant C is derived by imposing the condition $\sum_{u=1}^{m} \hat{q}_u = 1$ where

$$C = \frac{1}{\sum_{i=1}^{n} k(\|x_i^*\|^2)}, \tag{3}$$

since the summation of delta functions $u = 1...m$ is equal to 1.

3.3.3 Target candidates

Let $\{x_i\}_{i=1...n_h}$ be the *normalized* pixel locations of the target candidate centered at y_1 in the current frame. The normalization is inherited from the frame containing the target model. Using the same kernel profile $k(x)$ but with a bandwidth h, the probability of the feature $u = 1...m$ in the target candidate is given by

$$\hat{p}_u(y) = C_h \sum_{i=1}^{n_h} k\left(\| \frac{y - x_i}{h} \|^2\right) \delta[b(x_i) - u], \tag{4}$$

where

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k(\| \frac{y-x_i}{h} \|^2)} \tag{5}$$

is the normalization constant. The bandwidth h defines the scale of the target candidate, i.e., the number of pixels considered in the localization process.

3.3.4 Target localization

An object is localized by maximizing the similarity function between the two probability density functions. The search for the new target location is starts from location y_0 of the target in the previous frame. Then, the kernel is recursively moved from the location y_0 to the new location y_1, therefore:

$$\hat{y}_1 = \frac{\sum_{i=1}^{n_h} x_i.w_i.g\left(\| \frac{\hat{y}_0 - x_i}{h} \|^2\right)}{\sum_{i=1}^{n_h} w_i.g\left(\| \frac{\hat{y}_0 - x_i}{h} \|^2\right)} \tag{6}$$

where $g(x) = -k'(x)$. The mean shift algorithm calculates the similarity function for the target model and target candidate in subsequent frames and then uses equation 6 to localize the target candidate. The process is repeated until the similarity function goes down and the estimated location of the target changes less then a certain threshold.

4 USARSim and tracker implementation

The USARSim is used as a controlled environment for quantitative analysis the Mean-Shift tracker. The USARSim is designed as a high fidelity simulator of urban search and rescue (USAR) robots and environments as a tool for research. The simulator is build on top of the Unreal game engine which provides realistic graphics with accurate physics. This design enables researchers to focus primarily on robot control.

4.1 USARSim Overview

The USARSim will not be discussed in detail in this thesis, however, some knowledge is needed to properly understand the workings of the simulation and its environment. In figure 4.1 the system architecture is shown.

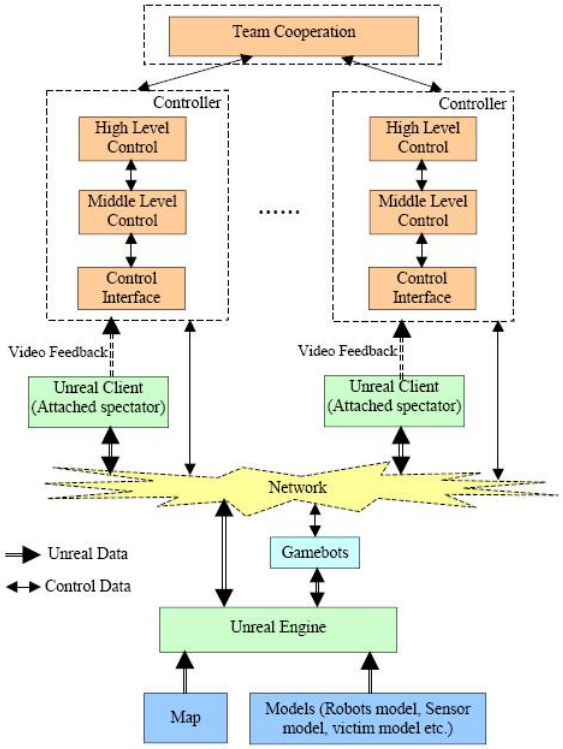

Figure 1: System architecture

For the USARSim environment [3], a framework by the name of USARCommander, has been developed by the UvA RoboCup team. USARCommander communicates with the USARSim Unreal Engine by sending and receiving strings of messages. USARCommander is being developed and improved on a daily basis and now covers most low-level control operations on the simulated robots. For accuracy and realism, there are no high-level controllers in the basic architecture. A few high-level controllers have been developed (MOAST, Pyro, Player), but generally users are required to develop their own controllers. Currently, there is no high-level controller publicly available for the ERS-7 AIBO robot or other legged robot.

4.2 Organization

The processes in USARSim consists of two main parts. The first is the simulator server, the second part are the clients. The clients control the robot in the simulated world, each client controls one robot. Clients can easily connect to the server due to the network organization of the simulation. This decentralized distribution of computational operations allows for efficient client handling enabling more robots in the simulator.
Rendering of visual camera data is done by a separate game client. It does so by raeding images directly from the graphics card buffer. These images are caught

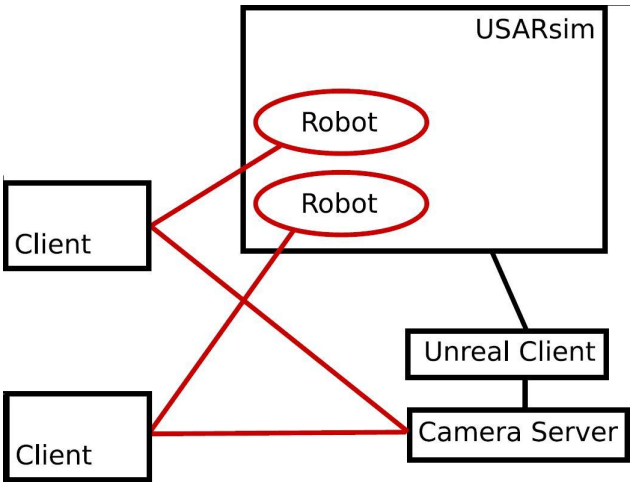

Figure 2: Organization of USARSim processes

by an image server, who compresses them and sends them to the appropriate client.
The organization is depicted in figure 4.2.

4.2.1 Mean-Shift Tracker module in USARSim

Client processes control the virtual robots. Such a process is also referred to as an Agent, controlling a single robot. The Agent organizes many tasks, e.g. it communicates with the simulator by hosting the network connection, it receives sensor data from the simulator, and it keeps track of all connected sensors and actuators. The Mean-Shift tracker works as a specialized agent with vision capabilities working on the received camera sensor images.

5 Experiments

The main goal of this research project was a quantitative analysis of the Mean-Shift tracker. Two experiments have been conducted to perform the analysis.

5.1 Experimental setup

The limitations and the usability of the Mean-Shift tracker are investigated using experiments in a controlled environment. The USARSim enabled changing individual conditions, so experiments could easily be reproduced. The algorithm is quantitatively tested on two conditions, the target speed, and various lighting conditions. Besides varying these conditions, we also tweaked internal parameters to obtain optimal performance. During each experiment we used the P2AT robot 3 as the tracker, and the ATRVJr 4 as the target robot. The tracker robot is equipped with a histogram of the target robot and does not need to acquire new histograms of the target. Active Vision is added to let the mobile robot adjust his view of the target.

Figure 3: P2AT robot

Figure 4: ATRVJr robot

The tracker receives images from the image server, which is set to deliver good quality jpegds sized at 480*360. The frame rate is set at 6 images per second to allow for real-time processing. For each experiment a frame rate of 6 frames per second has been used. A higher frame-rate influences the trackers performance and is therefore kept constant. This influence is omitted in the results below because this factor has remained constant during all testing.

The 3D histograms were constructed with 30 bins for every color dimension in RGB color space with range (0,255). Manual tweaking showed that 30 bins was an effective discretization of the color space. An image of the RGB cube is depicted in 5.1. Note that the RGB cube in the image uses 8 bins for each color dimension.

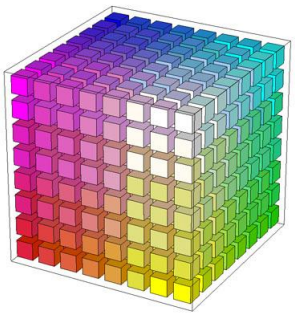

Figure 5: The RGB color cube

5.1.1 Active Vision

The Active Vision component enabled the tracker robot to adjust himself to keep the target centered in his view. For this, a simple proportional-derivative (PD) controller [4] was developed to adjust the speed of rotation to center the target in the field of view. The equation used for determining the output, in this case the rotation speed, is given by:

$$Output(t) = P_{contrib} + D_{contrib}$$

where P and D are the feedback values of the controller.
P gives the amount of error between the desired signal and the measured signal. An image width of 480 was used, centering the target implied centering the region of interest to pixel location 240. The value of P thus gives the amount of pixels the region of interest is away from the center in a single frame. This value can be multiplied by a proportional gain to emphasize significance. A higher value for P means a higher output and thus yields a faster response of the controller. Such a constant was not used.
The D value contributes to the equation by adding the derivative of the error vs. time. This ensures that the tracker adapts its rotation speed when a target accelerates or decelerates. More formally:

$$D_{contrib} = T_d \frac{de}{dt}$$

where T_d is the derivative of time and was set at 5. The derivative of the last 3 frames was used.
The Active Vision component was manually tweaked for optimal performance.

5.1.2 Condition I: changing the target speed

The speed of the target and the ability to track it relates to one important factor, the distance of the target. In the experiments, the target starts in the center of the trackers field of view and always leaves the field of view of the tracker with the same angle. However, at a short distance this angle is traveled faster when the speed of the target remains the same. The distance of the tracker to the target can thus cause large gaps of movement of the target in the recorded frames. When these gaps become too large, the tracker loses the target. The measurements are quantified by varying the speed of the target to a fixed distance and measured the ability to keep track of the target during each frame. We repeated the experiment for several distances. A screen shot of the setup is shown in 5.1.2.

Figure 6: Experiment setup

5.1.3 Condition II: changing lighting

One of the advantages of working in a controlled simulation is the ease of reproducing experiments. The advantage of implementing the tracker in the USAR-

Simulation is largely due to ease of reproducing lighting conditions. Lighting conditions are hard to control in real environments.
In order to utilize the advantage of working in a simulator we need to change the lighting conditions within the simulator, rather then editing the images received from the image server. This can be done using a map with varying lighting conditions. We used 5 different maps, each with a different lighting condition. The optimal distance and speed are repeatedly used, measured with the first experiment, to determine the effect of a lighting condition.

5.2 Results

In the following subsections the results from the experiments are given.

5.2.1 Results Condition I: changing the target speed

The speed of the target was varied at a fixed distance between the target and the stationary tracker. Distances in the experimental maps are expressed in Unreal Units. 250 Unreal Units correspond to 1 meter.
Active vision was not disabled during the experiment because it influenced the quantified measurement of the performance of the tracker. However, the Active vision calculations were performed but not signaled to the actuators to account for the active trackers total performance.
The performance is measured by dividing the number of frames that the tracker successfully tracks the target by the total amount of frames the target was in sight, $\frac{successfultrackedframes}{Totalframes}$. The measurements are repeated at different distances. The results are depicted in table 1.

Wheel speed in radials per second							
Distance (m)	0.5	0.75	1.0	1.25	1.5	1.75	2.0
2	1	1	$\frac{4}{7}$	$\frac{2}{5}$	$\frac{2}{4}$	$\frac{2}{4}$	$\frac{1}{3}$
4	1	1	1	$\frac{4}{7}$	$\frac{3}{6}$	$\frac{3}{6}$	$\frac{1}{4}$
6	1	1	1	1	$\frac{4}{7}$	$\frac{4}{7}$	$\frac{3}{7}$
8	1	1	1	$\frac{5}{10}$	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{1}{5}$
10	1	1	1	$\frac{9}{12}$	$\frac{8}{12}$	$\frac{6}{12}$	$\frac{5}{12}$

Table 1: Results varying target speed and distance

The results are plotted in figure 7.

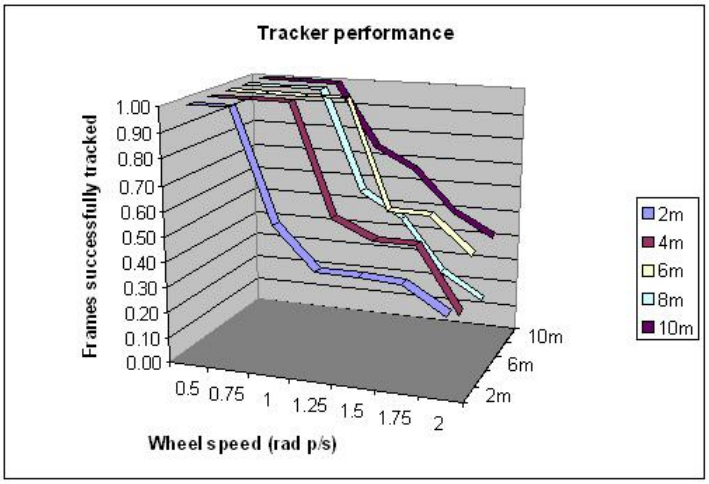

Figure 7: Plotted results

The results show that a target at 6m is tracked relatively long at a high speed. With this distance the target is perfectly in sight of the tracker's region of interest. A degrading performance is displayed when the target is at close range or far away. At close range, the tracker can become unstable because every movement of the region of interest finds the histogram of the target. When the target is far away, the size of the target is so small in the trackers region of interest that the tracker becomes sensitive to background noise.

5.2.2 Results condition II: varying lighting conditions

To test the effect of different lighting conditions, a distance of 6m and a wheel speed of 1 radials per second was used. These were optimal conditions for the tracker in the previous experiment.
Five different lighting conditions were used, standard lighting, hardly any lighting, standard outdoor lighting, forest lighting, and snow lighting. These conditions were easily producible using different simulator maps.
For each lighting condition, the tracker successfully tracked all frames, even when there was hardly any lighting. A table with results is omitted as every entry contained the value 1.
A screen shot is depicted in figure 8.
Lighting conditions are a serious obstacle in real world tracking applications. The conducted experiments might not have entirely captured that difficulty as the provided target histogram was very distinct, even with poor lighting. One may expect a degrading performance if the background colors are more similar to the target histogram. Lighting can distort background colors making them look similar to target histogram colors, however the current histogram was so distinct this effect seems to be evaded. As a result the performance of the tracker was not affected.

Figure 8: Hardly any lighting

5.3 Overall results with Active tracking

The experiments have been conducted to analyze the performance of the Mean-Shift tracker by itself. When the Mean-Shift tracker was combined with the Active vision component the tracker robot was autonomous able to track a target. The robot was released in the simulator and was following the target robot.
The behavior of the robot with active tracking confirmed the results from the experiments. When the target robot was moving fast the tracker robot would lose the target. Active Vision could sometimes correct the failing tracker by accidentely overshooting the target, the target would then pass through the region of interest of the tracker and the tracker was able to successfully track again. However, it could also backfire, the tracker could overshoot and, because of the local target search, could not relocate the target again if it did not pass through the region of interest.
The framerate of 8 frames per second also allowed for great changes of the target object between frames. This is due to the other computations of the USARCommander. A higher framerate would increase performance because movement changes of the target would be Besides losing a target based on the targets speed, the tracker could also lose the target when a background color distribution showed great similarity with the targets color distribution. In such a case the new mean location would not entirely translate to the center of the target but only by a limited amount. In the frames thereafter the tracker would lose the target object because the target object had shifted increasingly out of the trackers region of interest.
The results illustrated the importance of a typical distance. To maintain an optimal distance to the target a simple pixel counter was used that counted the amount of red pixels in the region of interest. The hue range, 355-360, 0-5, was used to detect red values. A low amount of red pixels indicated the target was moving away. The tracker robot responded by moving forward. When the red pixel count fell below a threshold the target was considered not in sight and a global search was perform to relocate the target.
Despite the attempts to maintain an optimal distance the tracker robot often

lost the target robot.

5.4 Suggestions for improvements

The target speed appeared to be a decisive factor in successful tracking. Ideally, the target robot should remain at a certain distance to cope with fast movement. The tracker is less affected by fast movement when the target robot is further away.
An improvement would be to calculate the distance of the robot based on the size of the region of interest. A small region of interest would then indicate a distant target. Another improvement would be to use a different shape of the region of interest, for instance an ellipsoidal shape. A varying shape could indicate different angles of a robot, e.g. when it turns. This information could then be used for more efficient and robust positioning.
A method for a varying kernel size and shape was developed by [5]. This approach is currently being developed on in the USARCommander framework. A more reliable tracking performance is expected with this approach.
[6] optimized the Mean-Shift tracker by using masked histograms. This however is not a robust solution for the RoboCup Rescue domain because many kernel templates would be required.
Another improvement is the combination of two different tracking approaches. Martijn Liem research [7] is currently combining a KLT feature tracker with the varying shape and size of the kernel method mentioned earlier. This approach should yield enormous improvements because the new method would complement the strengths of the two different tracking approaches. The USARSim with the USARCommander framework is an excellent platform to quantitatively measure the performance of the new approach.

6 Conclusions

The Mean-Shift tracker has been tested with quantitative measurements on target speed and lighting conditions. The results indicate that the tracker depends heavily on the targets speed and distance. The results illustrate the degrading performance when the target speed increases and when the target distance is low or high.
The results of varying lighting conditions proved another interesting observation. Because the target histogram was predefined, it was a very clear and explicit color distribution. Such a histogram was easy to track, even in poor lighting conditions. In a real world setting, such an explicit target histogram is hard to acquire. Eventually, the robot has to make real-time dynamic histograms of targets, in case of the RoboCup Rescue of victims. The results illustrate the importance of a good histogram. A *good histogram* implies an explicit representation of the target in the color space. A good histogram can greatly increase the robustness of the tracker. In the RoboCup Rescue, indicators of a victim, like the RFID sensor, could assist in the positioning of the robot for the creation of a victim histogram.
When the tracker robot was released in the simulator it did not track the target robot on a reliable basis. Overshooting, background distraction, and a fast

moving target would make the tracker robot lose the target. It would then try to relocate the target robot but would again soon lose it.
Clearly, the current approach to track a moving target does not qualify as a reliable tracking method. Because of this, the current approach is not suitable for use in the RoboCup Rescue Virtual Robot competition.
A number of improvements have been mentioned, use of a dynamic kernel, and combining two tracking approaches. Implementations of these improvements could easily by analyzed in the USARSim with the USARCommander framework.

7 Future work

Much remains to be done for a reliable USARSim tracker for the RoboCup Rescue Virtual Robot competition. The current USARCommander framework lacks a great deal of standard Computer Vision functions and libraries. These functions and libraries should be developed to allow for greater complexity of future tracking algorithms. The extension of the math library would be a good start.
For a more pleasant testing experience the user interface of USARCommander can be enhanced. The interface is currently sufficient but more flexible control would allow easier testing. The USARCommander can also be extended by incorporting high-level procedures for robot control. There is currently no high-level robot control for virtual legged robots, like the ERS-7 AIBO or QRIO. Developing walking-models for such robots would expand the range of capabilities of the USARCommander.
When the USARCommander framework extends, more complex tracking algorithms can be incorporated. These more complex algorithms, like the EM-Shift tracker, the KLT feature tracker or the fusion algorithm of Martijn would enable better tracking of a target. When incorporated in USARCommander these algorithms can be used for the RoboCup Rescue to provide a more reliable tracker. For researchers the performance of these algorithms can be measured quantitatively under a variety of easily reproducible conditions available in the USARSim.

References

[1] C. Tomasi and T. Kanade, School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-91–132 (1991).

[2] G. Welch and G. Bishop, ACM SIGGRAPH 2001 Course Notes (2001).

[3] J. Wang and S. Balakirsky, USARSim Sourceforge homepage - http://sourceforge.net/projects/usarsim .

[4] D. Sellers, (2001).

[5] Z. Zivkovic and B. Krose, Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on **1**, .

[6] E. Ben-Israel, Project report .

[7] M. Liem, Master thesis progress report .