Adaptive Visual Face Tracking for an Autonomous Robot

Herke van Hoof [a][b] Tijn van der Zant [b] Marco Wiering [b]

[a] *TU Darmstadt, FB Informatik, Hochschulstraße 10, D-64289 Darmstadt*
[b] *University of Groningen, AI dept., P.O. Box 407, NL-9700 AK Groningen*

Abstract

Perception is an essential ability for autonomous robots in non-standardized conditions. However, the appearance of objects can change between different conditions. A system visually tracking a target based on its appearance could lose its target in those cases. A tracker learning the appearance of the target in different conditions should perform better at this task. To learn reliably, the system needs feedback. In this study, feedback is provided by a secondary teacher system that trains the tracker. The learning tracker is compared to a baseline non-learning tracker using data from an autonomous mobile robot operating in realistic conditions. In this experiment, the learning tracker outperforms the non-learning tracker. This shows that we successfully used the teacher system to train a visual tracking system on an autonomous robot.

1 Introduction

For an autonomous robot operating in a non-standardized dynamic environment, it is crucial to reliably perceive the world around it. Perception has to be both real-time and robust for the robot to operate safely. Robots can be equipped with many different types of sensors, but visual perception of the environment is a popular choice. As the world is dynamic, robots have to be able to learn about new objects while performing their duties, even in non-standardized conditions.

Robot vision and tracking. Compared to the task of computer vision, the problem of robot vision is harder in some aspects as the system has to process images real-time using limited on-board computing power. Furthermore, the camera of a robot often has a limited resolution and quality. However, in some aspects the problem is also easier as subsequent images are related to each other. Tracking approaches exploit this: processing can be constrained to a small region of the image where the target is expected. Constrained processing enables real-time performance. Particle filters are often used to track visual targets, for example in [3, 10, 12, 15].

Most tracking systems have a model of the appearance of the target. This appearance depends on both internal properties (shape, reflectance properties and pose) and external properties (illumination conditions, camera parameters, and occlusion) [8, 12]. During the tracking task, the appearance of the target model can change. Handling these changes is sometimes considered the main challenge of visual tracking [12]. Appearance change can be reduced by evaluating the system under standardized conditions. However, this is not an option when the robot operates in a non-standardized real-world environment.

Adaptation with and without feedback. In the literature on visual tracking multiple techniques are used to be able to handle changing circumstances. First, it is possible to try to make the tracker model invariant to appearance changes, which was done in [8]. That study needed prior training on a large set of images under different conditions, so it is not suitable for learning new objects as they are encountered in the world.

A second approach is to adapt the model to the appearance of the target in detections in previous frames. In contrast to the approach described in the previous paragraph, this approach does not need a separate training phase but learns on the task continuously, making it possible to track novel objects for which no training database is available. This was the approach used by [12] to incrementally learn a low dimensional eigenspace representation to track faces or objects.

The danger of this approach is that the target might disappear or the tracker might slowly drift away from the target, with the model adapting to the background rather than to the target. Some visual tracking models take specific measures to prevent this. In [14], the color model was adapted only to those pixels that the current model could 'confidently' classify as belonging to the target. In [9, 10] the color model was only adapted if the current detection was sufficiently similar to the previous model. In [4, 7] the color model was only adapted to slow appearance changes to prevent learning when the target disappears or is occluded. In [15] the color features were adapted based on the best-matching shape features and vice-versa to prevent the tracker from drifting away.

In short, there are many different approaches to solve the problem of adapting the appearance model to data that does not actually correspond to the target. However, those approaches are not guaranteed to solve this problem. For instance, even if sharp changes are prevented from impacting the model, the tracker might still slowly drift away, adapting to the background. Also, the appearance of the target can change very fast, for instance if lights are turned on or off, in which case adapting to fast change is desired. To be certain the tracker adapts only to the target, feedback is needed.

Feedback mechanism. Feedback can be provided by another system. This might be a detector system that would be too computationally demanding to steer the robot in real time. In that case it might run on the robot's hardware at a lower temporal frequency, providing feedback in just some of the frames. Another solution would be to run such a system off-board on a more powerful machine. However, if the connection to the off-board system is not perfectly reliable, continuous feedback is still not guaranteed.

If there is no feedback system available, feedback can be provided by alternative sources. The robot can ask people near the robot to label unknown objects, or it can get feedback from physical interaction with an ambiguous object [11]. These types of feedback are however not always available. A service robot is not very valuable if its owner has to supervise it continuously. On the other hand, interaction with the environment requires the robot to invest time that might be better spent. So, the system has to be able to function with a limited amount of feedback. The limited feedback can be used to construct a stronger target representation over time. As feedback comes in, the tracker learns the appearance of its target in different conditions. This makes the tracker more robust, requiring less feedback as it is slowly able to handle more situations by itself.

Overview. In this study a tracker that uses feedback to learn a robust representation of its target is developed. Many kinds of targets could be used, but in the current research we will use faces as tracker targets. Our research question is whether the learning tracker is better at tracking faces in sensor data from an autonomous, mobile robot than a non-learning tracker.

The next section describes the tracker and the way it is trained by the teacher system. Section 3 describes how a robot is used to evaluate the tracker and the way the performance is measured and analyzed. The results of the evaluation are presented in section 4. Concluding, a discussion of the results and possibilities for future work is presented in section 5.

2 Vision-based Tracking System

Our tracking system consists of two main parts. A particle filter is used to track the target visually based on an appearance model. Feedback is provided by a separate teacher system, to enable the complete system to learn reliably. The particle filter uses its past experiences to select a target model appropriate for the conditions the system faces. Over time, the system experiences a variety of conditions, and it learns how to handle them. In this section we describe the tracker system first, after that we describe the teacher system and the way feedback is handled.

2.1 Tracker System

The tracker estimates the probability distribution over the state at time t. This non-Gaussian state-density is approximated using a particle filter [3]. The particle filter represents the state-density at time t with a sample set consisting of N samples $s_t^{(k)}$ and their corresponding weights $\pi_t^{(k)}$, with $k = 1, \ldots, N$. The samples represent the (x, y, width, height) parameters of a rectangle corresponding to a possible target location. The number of samples $N = 100$ was chosen as adding more particles hardly improved performance.

The particles are weighted with the probability of observing the current features if the target were present in the area covered by the particle. Like in [10], the area containing the target of the system and the areas covered by the particles in the current image are represented using three dimensional histograms of (R,G,B) values. The histogram uses $8 \times 8 \times 8$ bins that each contain the count of pixels with corresponding (R,G,B) values[1].

The weight of a particle is based on a comparison of histograms q (the appearance model of the target) and $p_{s_t^{(k)}}$ (the model of the area of particle $s_t^{(k)}$). The distance measure used to compare these histograms is the Bhattacharyya distance shown in equation 1, which sums over all m bins of the histograms. If the distance between two histograms is small, the corresponding particle's weight should be high. The weight is calculated using equation 2, which was used successfully in [10].

$$d\left(p_{s_t^{(k)}}, q\right) = \sqrt{1 - \sum_{u=1}^{m} \sqrt{p_{s_t^{(k)}}^{(u)} q^{(u)}}} \tag{1}$$

$$\pi_t^{(k)} \propto \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{d^2\left(p_{s_t^{(k)}}, q\right)}{2\sigma^2} \quad . \tag{2}$$

In equation 2, σ is a free parameter. With high σ, all particles get nearly equal weights. However, with a very low σ, all the weight centers around a couple of particles with a small distance, which means the face might be lost if these particles are not on the target. $\sigma = 0.1$ worked well in our case.

After weighting the particles, N samples are drawn with replacement by choosing a particular sample with a probability proportional to its weight. Then, the location of the target is estimated by taking the weighted mean of all particles. Each of the samples is then propagated according to the motion model. A simple 0^{th} order model is used, to keep the dimensionality of the search space low. This means that a Gaussian component is added to each particle. This component has a mean of $\vec{0}$, as there is no expected bias in scaling or translation. The variance of the Gaussian component corresponds to the expected amount of movement. Standard deviations of 20 pixels for x and y components, and 8 pixels for the height and width worked well. The sample set now corresponds to the state distribution of the next time step, given all observations up to now. As the next observation comes in, a new estimate can be made by repeating the entire process.

2.2 Teacher System and Feedback Handling

In the current research, the required feedback will be provided by a teacher system. This is a face detector that processes every frame of the video sequence from the robot's camera independently. This detector employs a boosted cascade of Haar-like features, as described in [5, 13]. This detector is implemented in the OpenCV library [1], which also contains cascades pre-trained on several datasets. In the current experiment, the 'default frontal face' cascade is used. The parameters that are used in the current research are OpenCV's suggested parameters for still images.

Feedback available. Every time the teacher system detects the target, two histograms are constructed. One of these models the target as it appears in the current image. The other histogram is a model of the complete image. This represents the environment conditions associated with the target model, and will be called the 'background model'. These two histograms are stored together in a database with earlier pairs of target model and background model. Furthermore, to prevent the tracker from drifting away, whenever the detector finds a face, all particles are re-initialized to the face location.

No feedback available. Not in every frame feedback is given. The teacher system can fail to detect a target, and the feedback system could be unavailable as discussed in the introduction. In every frame where no feedback is received, the best target model in the database has to be found. First a histogram is made of the entire current image. This background histogram is matched against the database of earlier background

[1] In a preliminary test, different representations were used based on the gradient and the gray scale value of each pixel, but these representations were less successful.

histograms. The distance measure used for this nearest neighbor search is the Bhattacharyya distance that was also used to weight the particles. This measure is shown in equation 1 on page 3.

We expect that the target model associated with the best matching background histogram was made under conditions similar to the current ones. This makes that target model the most appropriate appearance model for the current image, so it is used to weight the particles of the particle filter.

3 Robot and Experimental Setup

In this section, the robot which is used in the experiment is described, as well as its control architecture. Then we describe how the tracker is tested, and how the tracker's performance is measured and analyzed.

3.1 Autonomous Robot

The hardware platform used in the experiment is a Pioneer 2 mobile platform. The platform carries, among others, a high definition video camera used to provide input to the tracker system[2].

The tracker provides data to the robot control program that has the goal to follow a person. This program makes the robot turn to the detected face and approach it to a distance of approximately one meter. It does this by keeping the face in the middle of the visual field and at an apparent size that corresponds to a distance of one meter. If the apparent size of the detected face is smaller (the face is farther away) or the detected face is not centered in the middle of the field of view, a control signal proportional to the magnitude of the error (the difference between the actual value and the target value) will be sent to the robot to decrease the error.

First, the difference between the middle of the detected face to the middle of the horizontal field of view is calculated. If this difference is positive, the face is detected in the right half of the visual field. The speed for the left wheel is set to $K_{turn} * error$, the speed for the right wheel is set to $-K_{turn} * error$, where K_{turn} is the proportionality constant for turning and the difference is expressed in rad. This centers the target in the middle of the horizontal field of view.

Secondly, the difference between the desired apparent size of the face (corresponding to a distance of approximately one meter) and the actual apparent size of the face is calculated. A positive value means the apparent size is too small, indicating the robot has to approach. Both wheel speeds are then set to $K_{forward} * error$, with $K_{forward}$ the proportionality constant for approaching and the error expressed in pixel difference. If the difference is negative (indicating the apparent size is too big), output is suppressed as the user can be approaching the robot. Also, driving backwards might be dangerous.

The proportionality constants should not be too large, to avoid overshoot, but large enough to provide an adequate response to differences between actual values and target values. Finding the right value is a process of trial and error. We found that using $K_{turn} = 200$ mm / s per rad and $K_{forward} = 3$ mm / s per pixel yielded appropriate behavior. The output of both control actions are summed and the command is executed by the Pioneer platform. The sum was capped at a maximum of 200 mm / s to ensure safe operation of the robot.

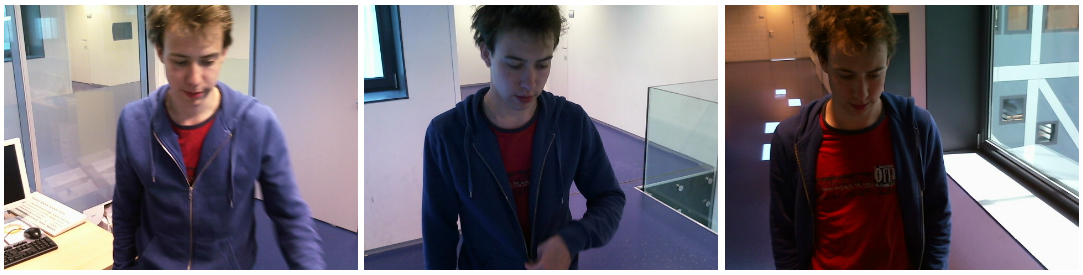
3.2 Experimental Setup

Data collection. In the data collection phase, the robot is guided by a person walking backwards, facing the robot. The robot follows the face using the tracker. The tracker gets feedback when available. The detections of the detection subsystem and all images are saved to disk to be used later.

The robot was guided along one long 'training' path to allow it to learn. The training sequence contains 1263 images with 1021 successful detections by the teacher face detector system. It was also guided along 18 'testing paths' to collect data for evaluation. The test sequences contain between 231 and 577 images, with between 134 and 366 frames in which a successful detection was made.

Each path contains at least one section in the university's robot laboratory and one section in the corridor. The illumination in these areas is very different: the robot laboratory has more artificial light, and the sections in the corridor get very close to the windows, causing varying light intensity as the windows are passed. Figure 1 illustrates the kind of images in the dataset and the different types of illumination encountered along a path.

[2]The BORG team description paper describes the platform in detail: http://www.ai.rug.nl/crl/uploads/Site/BORG_TDP_2011.pdf.



(a) frame 142 (b) frame 333 (c) frame 434

Figure 1: Some stills from a test sequence illustrating the different illumination conditions.

Training phase. In the training phase, the training sequence is processed by the tracker system. Whenever the teacher system detects the target, a color model of the entire image is stored together with a color model of the target found by the teacher system. All these models together form a database which can be used by the tracker to improve its performance.

Testing phase. To evaluate the quality of the trained system, we test the performance of the tracker in the absence of feedback. The teacher system is only used to initialize the tracker at the start of the path. After that, the tracker is not corrected anymore. Being able to perform in the absence of feedback makes the system tolerant to faults and to limited availability of feedback. Besides evaluating the trained tracker, we also evaluate the performance of a non-learning tracker that did not store any data in the training phase. The non-learning tracker has only one target model, representing the appearance of the target in the initial frame of the sequence. The trained tracker and the non-learning tracker are evaluated on the 18 testing sequences of recorded frames.

3.3 Data Analysis

In the testing phase, the feedback system is not used to correct the trackers. Instead, it is used to find the 'true' location of the face. This is done because there is no manual annotation of the ground truth in the sequences.

The face location estimates of both trackers are compared to the 'true' location as detected by the teacher system. The performance score suggested in [6] is used. It is a measure of overlap between the feedback system's detection (regarded as ground truth) and the tracker's estimate. The performance measure is shown in equation 3 with A_{true} the area of the true face location (in this case the feedback system's detection), $A_{estimate}$ the area of the box found by the tracking system, and $A_{overlap}$ the area of the overlap between those two boxes. In frames where the feedback system is not able to make a detection, no performance score is calculated.

$$Score = \frac{A_{overlap}}{\sqrt{A_{true} \times A_{estimate}}} \tag{3}$$

Comparing the learning tracker and the non-learning tracker. The average of all performance scores of each sequence is taken for both the trained learning tracker and the non-learning tracker. Then the number of sequences in which average performance of the learning tracker was higher than the average performance of the non-learning tracker is counted.

Under the null hypothesis, the learning and non-learning trackers perform equally well on average so the expected number of sequences in which the learning tracker performs better is 9 (half of 18 sequences). The two-sided binomial test is used to test whether the number of sequences where the learning tracker performed better is significantly higher than expected under the null hypothesis.

Effect of number of detections in the training file. We also take a closer look at the effect of the number of detections in the training file on the performance of the learning tracker. We use each of the 18 testing sequences as training data, and then evaluate the trained tracker on the other 17 files. The correlation coefficient between the number of detections in the training sequence and the performance of the tracker

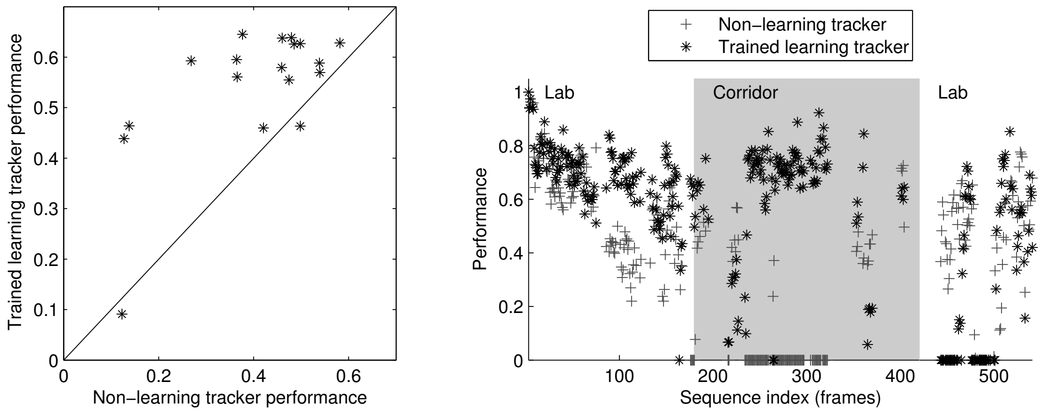


Figure 2: (a) Performance on all test sequences for the learning tracker and the non-learning tracker. Points above the $x = y$ line have a higher performance for the learning tracker. (b) Performance of both tracker types over time evaluated on one test sequence. The non-learning tracker suffers as conditions change. Performance is a measure of overlap between detection and tracker estimate, as explained in section 3.3.

can be calculated. The correlation coefficient can be transformed to a statistic with a distribution that is approximately normal using the Fisher transformation [2]. The resulting value can be compared to an expected value of 0 with a Z test to test whether the null hypothesis that the number of detections in the training set and the performance of the tracker are uncorrelated can be rejected.

4 Results

The results of the comparison of the trained learning tracker and the non-learning tracker are shown in figure 2(a). The learning tracker had a higher performance than the non-learning tracker in 16 out of 18 sequences. Under the null hypothesis that both trackers perform equally well, the number of sequences in which the learning tracker performs better should follow a binomial distribution $B(18, 0.5)$. Under this distribution, the probability of observing a similar or more extreme result than the observed 16 out of 18 sequences is < 0.005. So, the number of sequences in which the learning system outperforms the non-learning system is significantly higher than expected under the null hypothesis in a two-sided binomial test.

The performance of the trained tracker and the non-learning tracker over time is shown in figure 2(b). Within the robot laboratory, both trackers degrade in performance over time but the non-learning tracker suffers more. The trained tracker has different models for different illumination conditions so it is more robust to the slowly changing conditions. When the corridor is entered around frame 180, illumination conditions change abruptly and the non-learning tracker performance loses its target, while the trained tracker can switch to a different model and still has a good performance. When the laboratory is entered again, the initial model of the non-learning tracker fits the illumination again and it can recover.

Comparing different sizes of the training set. Learning tracker performances on training sets after training on different sequences are shown in figure 3. The correlation coefficient r between the number of detections in the training set and the performance score of the learning tracker is 0.536. This means $z(r) = 0.602$ according to the Fisher transformation [2]:

$$z(r) = 0.5 \ln \frac{1+r}{1-r} \tag{4}$$

Under the null hypothesis, the population correlation coefficient ρ is expected to be 0. There are 18 data points, so $\hat{\sigma}_{z(r)} = 1/\sqrt{18-3} = 0.258$. $Z = (z(r) - z(\rho))/\hat{\sigma}_{z(r)} = 2.33$. This means the null hypothesis that performance is independent of the number of detections in the training set can be rejected at the $\alpha = 0.05$ significance level: there is a positive correlation between the number of detections in the training set and the trained tracker's performance.

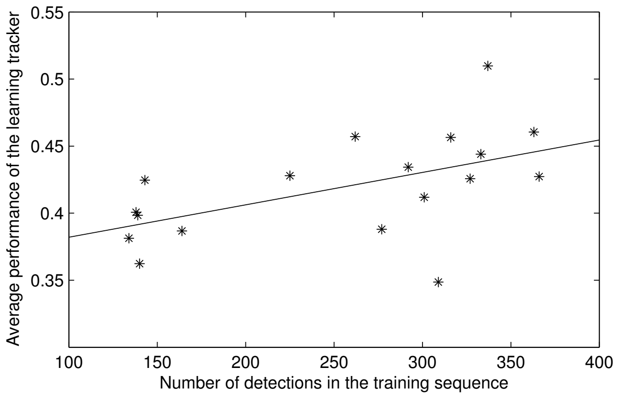


Figure 3: Learning tracker performance score depends on the amount of training data. Performance score measures overlap between detection and tracker estimate, as explained in section 3.3. The line shows the best least-squares fit of a first-order polynomial to the data.

5 Discussion

A tracker was developed that uses a learning algorithm that attempts to use the best target model given the color distribution of the current frame. This was hoped to provide robustness against illumination changes, as target models for different illumination conditions were available.

The trained learning tracker estimated the position of the target better than the non-learning tracker in 16 out of 18 test sequences. This is significantly higher than the number expected under the null hypothesis of doing better in half of the test cases. We found a significant positive correlation between the number of detections in the training set and the learning tracker's performance. So, one perceptive system successfully trained another perceptive system to become better over time at controlling an autonomous robot.

The described learning method is able to work continuously: it stores pairs of target and background models as feedback comes in over time, even while the robot is performing its task. Despite this, the experiment was split up in a train and a test phase, and evaluation was done on recorded sequences rather than on the working robot. This was done to be able to evaluate the learning and non-learning systems on the same data to get a reliable comparison of their performance. However, during the data collection phase, the robot was learning while tracking the target. This shows that our system is capable of continuously learning while executing the task. The tracking task itself was executed well by the robot. As long as turns were made slowly to allow the robot to keep up, the robot was able to follow the person correctly.

5.1 Future Work

The learning algorithm. The learning method described in this study was not very complex: pairs of target and background models were associated using instance-based learning. However, this instance-based learning method does not scale up well, as the memory requirement goes up linearly with the amount of training data. To scale this method up, clustering could be used to group similar training instances together or a different learning technique could be used.

Using prediction to improve tracking. In the current system, selected particles are updated with a Gaussian component with a mean of $\vec{0}$. An improvement would be to predict in which direction the target will move based on the action executed by the robot: if the robot turns left, the target is expected to move to the right in the visual field. This knowledge can be integrated quite easily in the particle filter by setting the mean of the Gaussian component to the expected translation and scaling. This extension to the tracker system is expected to situate particles closer to the actual location of the target, and so it is expected to improve the performance of the system.

Acknowledgment. Preliminary work on this research was initially performed at the Robot Cognition Laboratory, INSERM, Lyon, France, and we thank P.F. Dominey for support in that work.

References

[1] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.

[2] P.R. Cohen. *Empirical Methods for Artificial Intelligence*, pages 130–132. The MIT Press, 1995.

[3] M. Isard and A. Blake. Condensation-conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[4] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1296–1311, 2003.

[5] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 900–903, 2002.

[6] B. Martinkauppi, M. Soriano, S. Huovinen, and M. Laaksonen. Face video database. In *Proceedings of the First European Conference on Colour in Graphics, Imaging and Vision*, pages 380–383, 2002.

[7] S.J. McKenna, Y. Raja, and S. Gong. Tracking colour objects using adaptive mixture models. *Image and Vision Computing*, 17:225–231, 1999.

[8] H. Murase and S.K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[9] T. Nakamura and T. Ogasawara. Online visual learning method for color image segmentation and object tracking. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 222–228, 1999.

[10] K. Nummiaro, E. Koller-Meier, and L. Van Gool. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, 2003.

[11] R. Pfeifer and C. Scheier. Sensory–motor coordination: The metaphor and beyond. *Robotics and autonomous systems*, 20(2-4):157–178, 1997.

[12] D.A. Ross, J. Lim, R.S. Lin, and M.H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1):125–141, 2008.

[13] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[14] Y. Wu and T.S. Huang. Color tracking by transductive learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 133–138, 2000.

[15] Y. Wu and T.S. Huang. Robust visual tracking by integrating multiple cues based on co-inference learning. *International Journal of Computer Vision*, 58(1):55–71, 2004.