Active Tracking Using Kernel-Based Vision Processor and Robust Fuzzy Control

Alireza Mustafa [1] Moteaal Asadi Shirzi [1] Muhammad Reza Hairi Yazdi[1]

[1] Control and Robotics Lab., University of Tehran

Abstract

In this paper we introduce a practicable system by combining a vision processing algorithm and a fuzzy controller to obtain an efficient active tracker. Target tracking performance is heavily dependent on a good blend of vision algorithm and control. Because of performance and computational complexity, in practice, many visual tracking algorithms cannot be linked with control systems to track objects in real-time. Robustness and speed are the two major bottlenecks of current visual tracking algorithms. In this paper, the target's visual model is used along with a kernel-based searching algorithm to predict the target position. A model update is also incorporated to recognize when the target's appearance is changing due to its pose change. In case of target track loss, a search algorithm sweeps the space in vicinity of last target position to recover the lost target. A motion detection module used in our tracking system not only helps to initiate the tracking process automatically, it also finds the target's presence after track loss. A fuzzy control is also synthesized to reach our control performance objectives. The idea is implemented in an active camera system to track moving targets. In addition, we've used the parallel processing technique for vision and control to reach acceptable speed and accuracy in real-time tracking.

Keywords: Tracking, Fuzzy Control, Mean Shift Algorithm, Kernel-Based Tracking, Parallel Processing

1. Introduction

Target tracking has a wide array of applications in computer vision such as surveillance, vision-based control, smart rooms, robotics, military and driver assistance. Although it has been studied for dozens of years, object detection and tracking remains an open research problem. Tracking under the constraint of low computational complexity presents a challenge in real time tracking applications.

Recently, the mean shift algorithm was employed for object tracking, measuring the similarity between consecutive frames based on Bhattacharyya coefficient [1]. A major limitation of the mean shift tracker is that it uses one object model to track the target during the entire tracking process. Hence, it fails to track the target as it undergoes severe deformations and rotations.

To achieve robust tracking, model updates should be considered. In [3], adaptive Kalman filter is used to filter the object kernel histogram and obtain the optimal estimate of the object mode. Another problem is also experienced with mean shift tracker. It cannot track objects apart for more than the size of the target model between two consecutive frames. This means that if the camera is unable to capture the target movement between these two frames, it virtually fails to track it.

An extension of mean shift algorithm employs spatiograms (instead of histograms) which improves the performance of the mean shift tracker under wild rotations of the target [6]. However, it does not solve the second problem mentioned previously. Thorough global template search for the target is not a practical solution as it burdens the computer in real-time. More importantly, it is not suitable for real time tracking because of the controller stability issues that can result from erratic sampling rates. However, use of spatiogram has one evident advantage over histograms due to the integration of distance information that helps distinguish between different objects of similar color distribution.

Active tracking involves one or more cameras mounted on a mutable platform whose motion is to be controlled to keep target's location in a certain position within the image. To achieve high performance visual tracking, a well-balanced integration of vision and control is particularly important [7]. System dynamics is essential to optimize the performance of the design.

Recent years have witnessed rapidly growing use of fuzzy control systems in engineering applications.

The numerous successful applications of fuzzy control have sparked a flurry of activities in the analysis and design of fuzzy control systems. [4]. Stability and robustness are the two major points that our active tracking system can benefit from fuzzy controller. In this paper, a slightly modified version of mean shift algorithm is coupled with a real-time fuzzy controller to achieve an efficient visual tracker. In fact the vision processor determines the position of target and controller provides the suitable control signal for motors. It's crucial to use an adaptive controller, since, if camera fails to keep up with the target, the vision processor loses the target's sight and the tracker can not continue any more.

In case of target loss, an intelligent local search algorithm sweeps the scene in vicinity of last target position. A motion detection module implemented alongside our tracking system not only helps to initiate the tracking process automatically, it also finds the target's presence after track loss. In addition, we've used the parallel processing technique to reach acceptable speed and accuracy in real-time tracking. We have implemented this idea on our active camera system and obtained promising results.

2. Introduction

2.1. Target representation [1]

To characterize the target, first a feature space is chosen. The reference target model is represented by its pdf (probability density function), q, in the feature space. For instance, the reference model can be chosen to be the color pdf of the target. In the subsequent frame, a target candidate is defined at location y, characterized by the pdf p(y). To satisfy low computation cost imposed by real time processing, discreet densities or m-bin histograms should be used. Thus, we have:

$$\hat{q} = \{\hat{q}_u\}_{u=1\ldots m} \qquad \sum_{u=1}^{m} \hat{q}_u = 1$$
$$\hat{p}(y) = \{\hat{p}_u\}_{u=1\ldots m} \qquad \sum_{u=1}^{m} \hat{p}_u = 1 \tag{1}$$

2.2. Target model [1]

A target is represented by an ellipsoidal region on the image. To eliminate the influence of different target dimensions, all targets are first normalized to unit circle.

Let $\{x_i\}_{i=1\cdots n_h}$ be the normalized pixel locations of the target region in the current frame. The probability of feature u in the target model is calculated from:

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

where $b(x_i)$ is the index of the bin at the pixel location X_i, δ is the Kronecker delta function and k is an isotropic kernel profile. The presence of such a kernel function assigns more weight to central pixels and reduces the effect of peripheral pixels. The constant, C, is computed by using the fact that the sum of all probabilities over the histogram is one.

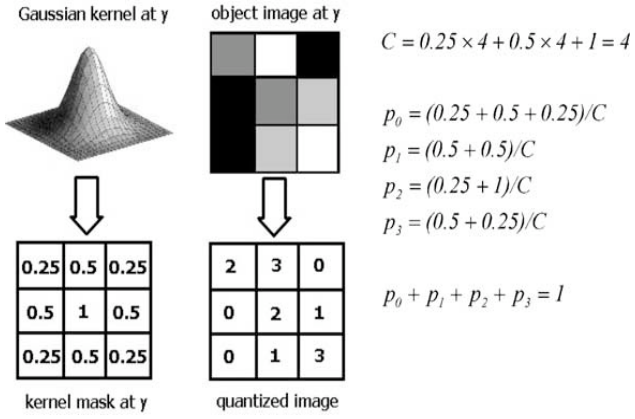


Fig. 1: Example of kernel histogram construction [3]

2.3. Target Candidates [1]

Let $\{x_i\}_{i=1\cdots n_h}$ be the normalized pixel locations of target candidate, centered at y in the current frame. The probability of feature u in the target candidate is given by:

$$\hat{P}_u(y) = C_h\sum_{i=1}^{n_h} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)\delta\left[b(x_i)-u\right] \tag{3}$$

where h is the bandwidth (number of pixels in the region) of target candidate and C_h can be calculated the same as C. The similarity function ρ is defined by the Bhattacharyya coefficient as:

$$\rho(y) = \sum_{u=1}^{m}\sqrt{\hat{P}_u(y).\hat{q}_u} \tag{4}$$

2.4. Target localization [1]

Localization maximizes the similarity function between the two probability density functions. The search for new target location starts at the location y_0 of the target in the previous frame. The Taylor expansion of $\rho(y)$ around $\rho(y_0)$ yields:

$$\rho(y) \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\hat{P}_u(\hat{y}_0).\hat{q}_u} + \frac{1}{2}\sum_{u=1}^{m}\hat{P}_u(\hat{y})\sqrt{\frac{\hat{q}_u}{\hat{P}_u(\hat{y}_0)}} \quad (5)$$

Taking the partial derivative of $\rho(y)$ with respect to y and setting it to zero results in the mean shift equation. In this procedure, the kernel is recursively moved from the current location y_0 to the new location y_1, therefore:

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

where $g(x) = -k'(x)$. Kernels of Epanechnikov profile are typically used, i.e.:

$$k(x) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-x) & if\ x \le 1 \\ 0 & otherwise \end{cases} \quad (7)$$

where C_d is the volume of unit d-dimensional sphere and d is the dimension of the state space (2 in our case) [5]. In this case, derivative of the kernel profile, $g(x)$, is constant and equation (6) reduces to:

$$\hat{y}_1 = \frac{\sum_{i=1}^{n_h} x_i.w_i}{\sum_{i=1}^{n_h} w_i} \quad (8)$$

Hence, the mean shift algorithm calculates the similarity function for the target model and target candidate in subsequent frames and then, estimates the location of the target candidate using (8). This process is repeated until the similarity function goes up and the estimated location of the target changes less than a desirable threshold.

2.5. Target model update

Updating the target model during the tracking process is important if this process is to sustain for a long period of time. The generic target model update procedure can take on the following form:

$$q_{t+1} = c \cdot (q_t + f(\rho(q_t, p_t), \alpha, ...) \cdot p_t) \quad (9)$$

where c is an arbitrary constant that is usually one; f is a function of the similarity, ρ, between the current target model and candidate model; α,… are other visually extractable elements such as the perimeter of object in pixel. In [11], f is taken as an exponential function as follows:

$$f = e^{-a[1-\rho(q_t, p_t)]} \quad (10)$$

where a is the update rate and was taken as 10. In this paper we also used an update function in a similar fashion. The function has the advantage of having the least effect on target model when the similarity function is obtained too low. Hence, even if the target suddenly is buried in scene clutter the updating procedure will have no significant effect on our previous target model and it still holds the visual characteristics of last-seen target

2.6. Simulation results

To objectively measure the performance of mean shift tracker, a computer program was written in Microsoft ® Visual Basic ®6. Experiments with different targets showed promising results. The following pictures capture the results of one tracking example:

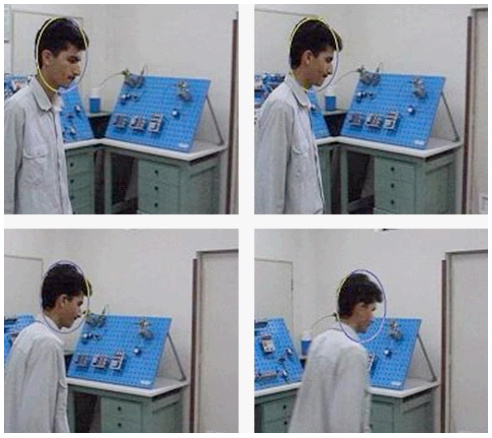


Fig. 2: Mean shift tracking example for frames number 1(top left), 14(top right), 28(bottom left), 31(bottom right); the ellipse around the head shows the target region.

3. Motion detection [10]

There are many approaches for detecting motion in video. Almost all of them rely on comparing the current frame with preceding frame also called background. By performing a simple subtraction between the present frame and previous one as background, and then applying a threshold filter to

eliminate the effects of noise, a poor yet simple and cost effective algorithm is obtained. The following figure shows the result.

Fig. 3: Motion detection (first method)

Note that the red channel of current frame has been merged with motion region. The disadvantage of this algorithm is that if the object moves slowly it gives no result at all. To solve the problem, the first frame can be used as background. However, this method is still weak when objects leave the scene. Therefore, updating the background frame should be made based on the criterion described later on in this paper. Using the sequence of difference, threshold, opening and edge detection the following result has been achieved (Fig. 4).

Fig. 4: Motion detection (second method)

To get better results, before processing the current or background frames, a mosaic filter has been applied. This method can exploit the mosaic-ed frames to improve the computational cost of the algorithm based on the idea that mosaic-ed pictures have less dimensionality. The following figure (Fig. 5) shows the result of the described method.

Fig. 5: Motion detection (third method)

An optimized version of the third approach can outperform any other motion detection algorithm in speed and efficiency. To reach better results, other decision-making rules must be embedded. For example, using simple edge following algorithm, one can determine the location of each object. Compare the location of each object between consecutive frames and decide whether to update the background frame and treat those slow moving object as background and/or decide if an object has left the scene.

It is necessary for the camera to be static while motion detection is being performed; hence this algorithm is currently suitable for automatic target tracking initiation and for recovery of tracking in case of a target track loss.

4. Control synthesis

Owing to the interconnection of vision and control system, the optimization problem of active tracking is rather nontrivial. For instance, if robustness of tracking algorithm implies more latency, the control system would become unstable. Whereas, high overshoot of the closed loop system will cause the vision processing program to lose sight of the target. The following figure shows the block diagram of the control loop (Fig. 6):

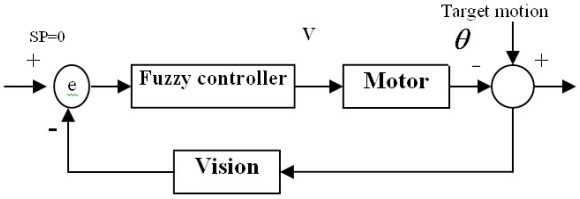


Fig. 6: Closed loop model of active tracking system

Now suppose that the target moves certain degrees off the center of the camera. The vision algorithm calculates this amount of eccentricity and sends a proportional signal to the fuzzy controller for compensation. Hence, the control loop keeps the target in line with the center of camera. To compute the eccentricity of the target, the following formula is used:

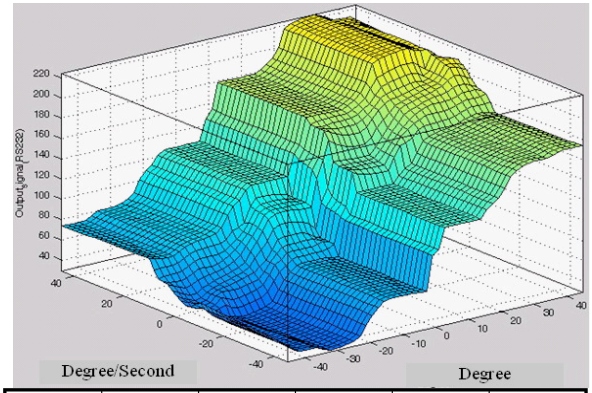
$$\Delta\theta_p = \left(x_c - x_o/w\right).\delta_p \,,\; \dot{\theta}_p = \frac{\Delta\theta_p^1 - \Delta\theta_p^2}{\Delta t_{1\to 2}} \qquad (11)$$

$$\Delta\theta_t = \left(y_c - y_o/h\right).\delta_t \,,\; \dot{\theta}_t = \frac{\Delta\theta_t^1 - \Delta\theta_t^2}{\Delta t_{1\to 2}}$$

where x_c and y_c refer to the location of the center of target in the image, x_o and y_o refer to the location of center of the image; h, w are the total height and

width of captured image in the local coordinate system of vision program (all in pixel or similar unit). $\Delta\theta_p$ and $\Delta\theta_t$ are the pan and tilt angle eccentricities, δ_p and δ_t represent the field of view of camera in the pan and tilt directions. The $\Delta\theta_p$, $\Delta\theta_t$, $\Delta\theta'_p$, $\Delta\theta'_t$ are the inputs of fuzzy controller. The fuzzy box of both pan and tilt directions has fifty five fuzzy rules. The fuzzy rules have been refined according to the observed performance of system. As for the optimization of these rules, we ran the real active camera frequently and tweaked the parameters of rules accordingly to reach better performance.

The fuzzy controller is implemented in Matlab ®6.5. By parallel processing the suitable control signal is produced at acceptable times. Although there are some problems such as delays in our hardware, the combination of fuzzy controller and visual processing can produce the control signal that guaranteed stability and robustness.



Angle	*Angular Velocity*	*Controller Signal*	*Angle*	*Angular Velocity*	*Controller Signal*
Very low	Low	Too much high	Medium	High	Low
	Medium	Too high	High	Low	Medium
	High	High		Medium	Low
Low	Low	Too high		High	Too low
	Medium	High	Very high	Low	Low
	High	Medium		Medium	Too low
Medium	Low	High			
	Medium	Medium		High	Too much low

Fig. 7: The surface of fuzzy logic rules in pan and tilt directions (top), some of the fuzzy rules implemented in our controller (bottom)

5. Our contributions

To achieve an automatic target tracker, motion detection, target tracking and control must be combined. One can employ the information resulted from the motion detection algorithm between two consecutive frames to automatically create a target model (based on the histogram of each identified entity in scene), and track the object of interest accordingly.

The mean shift method's kernel-based similarity function gives a criterion to reject other objects and identify the actual target. When the target is temporarily lost, the camera is stopped and moving targets are scanned using motion detection to find any similarity with our previous visual target model. If the algorithm, cannot find the potential target, it continues to move the camera in the direction of last-recorded velocity and continues its search procedure afresh.

To account for target changes, the model update based on the Bhattacharyya coefficient is used as described above. Since our camera is able to capture at 20 fps and our computation power allows to process at this rate, our model update hardly ever fails to capture the latest changes in case of typical target such as persons, runners and, cars. When the target is hidden behind another object, the similarity function obtains a low value for the candidate model and hence the update procedure virtually does not update it last target model.

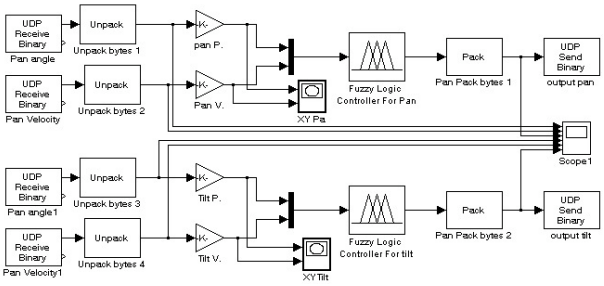


Fig. 8: The schematic of the Fuzzy Controller in Simulink® (Matlab)

6. Implementation and Experimental results

To assemble the active tracker, vision processor and fuzzy controller are each implemented on two separate computers. This way allows each portion to usurp full computation power of its own processor.

To link these two components, we have parallelized the computers by the UDP protocol for communications. The vision processor extracts the target's position from the image streams delivered by the camera and packages the data to be sent over the predefined UDP sockets. On the other end, the Matlab ® sits in wait for the UDP packets, calculates the control signals by the fuzzy controller and relays commands to our active camera. The camera could track different objects such as human face or body excellently in both tilt and pan directions. Because of some unknown hardware flaws, sometimes, delays occurred in our tracker and caused the target to be lost. This problem is not the inherent problem of our

algorithms or implementation problems and can be lifted by using a more powerful hardware.

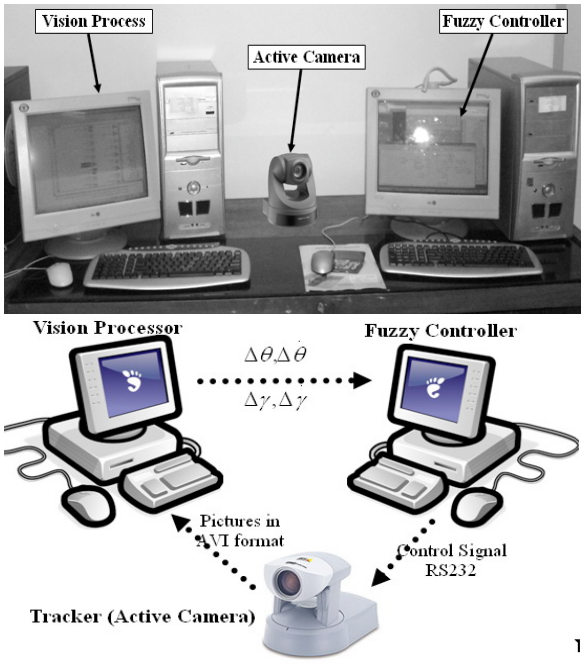


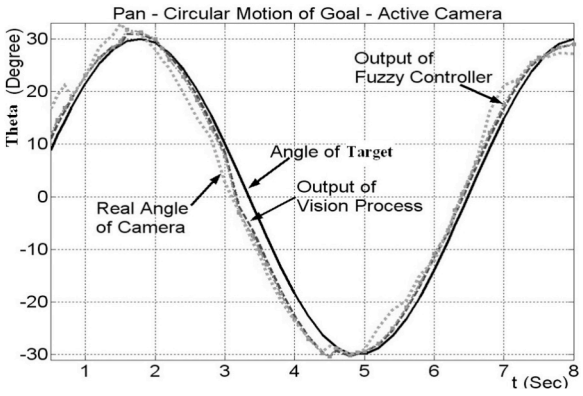
Fig. 9: Using the parallel processing, the computational load is divided between two computers and delay times can decrease. The schematic view is show on the bottom and the real setup to test the proposed algorithm on top.

Two different experiments are considered to test the fuzzy controller and vision algorithm. In the first experiment, the active camera tracks a black spot on a rotating circular plate which paces back and forth between two limits. The real position of the black spot is:

$$\theta = \tan^{-1}(\frac{r}{l})\sin(\omega t) \quad , \quad \alpha = \tan^{-1}(\frac{r}{l})\cos(\omega t) \quad (12)$$

where θ is the pan angle and α is the tilt angle; r refers to the radius of rotation of the black spot and l is distance between the active camera and the circular plate; ω represents the angular velocity of plate and finally t denotes time. Fig. 10 shows the results in terms of angle of camera and outputs from vision and controller that the active camera has tracked.

In the second experiment, a person with a constant velocity, moves around the camera and camera track his face. Fig. 11 shows the results of this experiment in the same way as Fig. 10.



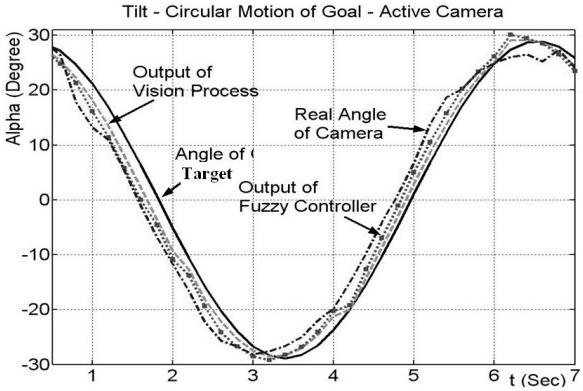


Fig. 10. Absolute angles of camera, target and outputs of vision and fuzzy controller systems are shown in both pan (on top) and tilt (on bottom) directions for the black spot experiment.

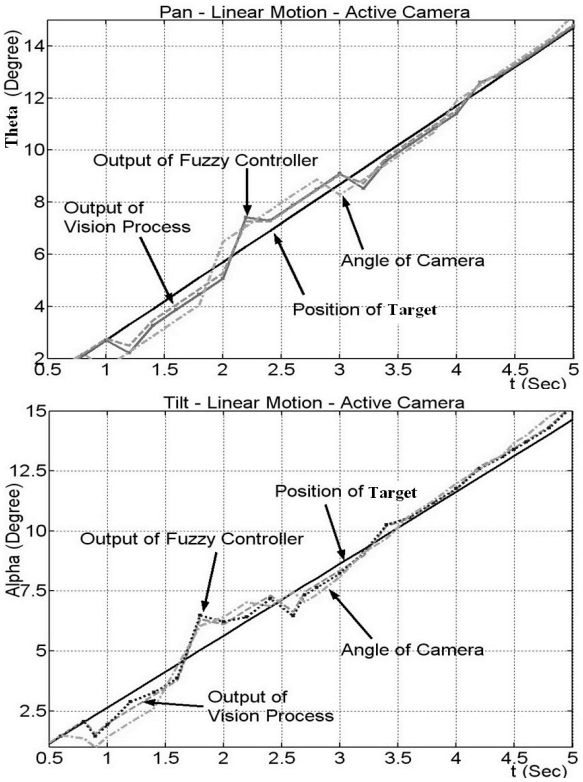


Fig. 11. Absolute angles of camera, target and outputs of vision and fuzzy controller systems are shown in both pan (on top) and tilt (on bottom) directions for the constant-velocity person.

7. Conclusions/Recommendations

An active target tracking control system was synthesized. Discussions were also made on the basic functions on motion detection and target tracking algorithms. An algorithm was presented that modified the weakness of the mean shift tracker with its target model update that together with fuzzy controller were able to track relatively fast moving targets. The target tracker was then tested in a real-world experiment and

the system achieved reasonable results in terms of overshoot and rise time.

Since the above mean shift tracker works upon histograms, it does not contain geometric information. In fact, another object with similar color distribution can behave much like the tracked target resulting in wrong performance. To overcome this problem, use of spatiograms that contain distance-like information is recommended. However, the computational load incurred on the computer makes it a less attractive approach. It involves more operations (multiplication and division) and shares some of the problems associated with the mean shift algorithm. Further works are to be conducted to make improvements and solve these problems.

The fuzzy controller also helped greatly to mitigate many of the problems in the hardware design. When the tracker loses the target, the fuzzy controller builds the suitable control signal to find the target. In the future, we will build a better hardware with fewer backlashes and model the effects of all such shortcomings in simulations.

8. References

[1] Comaniciu, D., Ramesh, V. and Mere, "Kernel-based object tracking," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003.

[2] Carlos P., Oscar R., M. Asunción Vicente, "Robot hand visual tracking using an adaptive fuzzy logic controller", Industrial System Engineering Department, Miguel Hernández University, 2004, http://wscg.zcu.cz/wscg2004/Papers_2004_Poster/M31.pdf

[3] Peng, N.S. and Yang, J. and Liu, "Mean shift blob tracking with kernel histogram filtering and hypothesis testing", Pattern Recognition Letters, 2005.

[4] Tanaka, K. and Wang, H. O., "Fuzzy Control Systems Design and Alanysis", John Wiley & Sons, 2001.

[5] Porikli, F.M., "Human body tracking by adaptive background models and mean shift analysis", IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, 2003

[6] Birchfield, S.T. and Rangarajan, S., "Spatiograms versus histograms for region-based tracking", IEEE conference on computer vision and pattern recognition (CVPR), San Diego, California, 2005.

[7] Barreto, J. P. and Peixoto, P. and Batista, J. and Araujo, H.,"Integrating Vision and Control to Achieve High Performance Active Tracking", 1998, www.isr.uc.pt/~jpbar/Publication_Source/iros1998.pdf

[8] Paul, Y. O. and Rares, I. S., "Enhancing Camera Operator Performance with Computer Vision Based Control", International Journal of Signal Processing, Vol. 1, No. 2, ISSN: 1304-4494, 2004.

[9] Kirillov, A, "Motion Detection Algorithms", (2005), www.codeproject.com/cs/media/Motion_Detection.asp

[10] Li, T.-H.S., & Shih-Jie Chang, & Wei Tong, "Fuzzy target tracking control of autonomous mobile robots by using infrared sensors", IEEE Transactions on Fuzzy Systems, 2004.

[11] Babu, R.V. Perez, P. Bouthemy, P., "Robust tracking with motion estimation and kernel-based color modeling", IEEE International Conference on Image Processing, ICIP 2005

[12] Christer Carlsson and Péter Majlender, "On Fuzzy Real Option Valuation", IAMSR Academy University, 2005 http://www.realoptions.org/submission/papers2005/PAris2005Article.pdf

[13] H. Wu, Q. Chen, and M. Yachida, "Face detection from color images using a fuzzy pattern matching method", Pattern Analysis and Machine Intelligence, IEEE Transactions, 1999.