2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Efficient visual servoing with the ABCshift tracking algorithm

Rustam Stolkin, Ionut Florescu, Morgan Baron, Colin Harrier and Boris Kocherov

Abstract— **Visual tracking algorithms have important robotic applications such as mobile robot guidance and servoed wide area surveillance systems. These applications ideally require vision algorithms which are robust to camera motion and scene change but are cheap and fast enough to run on small, low power embedded systems. Unfortunately most robust visual tracking algorithms are either computationally expensive or are restricted to a stationary camera.**

This paper describes a new color based tracking algorithm, the Adaptive Background CAMSHIFT (ABCshift) tracker and an associated technique, mean shift servoing, for efficient pan-tilt servoing of a motorized camera platform. ABCshift achieves robustness against camera motion and other scene changes by continuously relearning its background model at every frame. This also enables robustness in difficult scenes where the tracked object moves past backgrounds with which it shares significant colors. Despite this continuous machine learning, ABCshift needs minimal training and is remarkably computationally cheap. We first demonstrate how ABCshift tracks robustly in situations where related algorithms fail, and then show how it can be used for real time tracking with pan-tilt servo control using only a small embedded microcontroller.

Index Terms— **ABCshift, CAMSHIFT, Meanshift, tracking, servoing, adaptive background model.**

I. INTRODUCTION

Robot vision has important applications to mobile robots and wide area surveillance. For example, consider a robot vehicle which is visually guided to follow a moving target, or the use of a large number of cheap pan-tilt surveillance cameras scattered over a region of interest to monitor pedestrians or vehicles. For military and other operations we can envisage a combination of these tasks, (e.g., sending out a large number of small, cheap mobile surveillance units to penetrate and survey a hostile area). In these kinds of applications, we need a visual tracking system that is robust to scene change associated with motion of the camera as well as the target. However, the vision algorithm must also be fast and computationally inexpensive so that it can be implemented at real time frame rates on cheap, lightweight, low power embedded systems.

Popular methods of tracking moving targets include variations on the theme of background subtraction. Research in this area has focussed on methods of adaptively updating the background model to cope with gradual scene changes [1], [2], [3] . Unfortunately these methods fail unless the camera is stationary. The background model is only relearned very slowly over many frames, so that when the camera moves and the scene changes instantly, large parts of the image will be falsely detected as "moving target regions". Thus, these methods are unsuitable for robotic vision systems which involve motorized camera motion.

Fundamental work in visual servoing [4], [5] detect moving targets from a moving camera system by segmenting coherent regions of optical flow. This research demonstrated impressive speed and robustness, but tended to rely on expensive hardware, because the optical flow calculations are computationally intensive and because accurate rotation sensors are needed in order to accurately measure the camera motion which is then subtracted from the motion field observed by the camera. In contrast, our method is robust to arbitrary, unknown camera motion. Other visual servoing work, [6], is also robust to camera motion, but is limited to tracking flat, non-deforming targets.

Other researchers have also attempted to control a moving camera on robot arm arrangement with respect to a tracked object. The method in [7] relies on best fitting a model of the tracked object to segmented lines and edges. The method of [8] avoids the need for high resolution edges by fitting the object model directly to segmented image regions. However, both of these methods depend on detailed 3D models of the tracked object, so are not suited to applications where new objects must be rapidly detected and learned without human intervention. These methods are also unable to track deformable objects. Other research ([9], [10]) approximates deformable bodies (of known types) to kinematic chains of rigid bodies, but at considerable additional computational expense and the same need for prior knowledge of the tracked object.

Other popular and effective approaches to tracking moving targets include various approaches to tracking deformable blobs. This type of tracking eliminates the need for a 3D model of the object (which may be unavailable in practice) and generally uses the features of the object read from the image itself. To quote only some of the recent algorithmic development we mention color blob tracking (CAMSHIFT [11], [12], Mean Shift[13]), deformable boundary tracking (active contour model [14], CONDENSATION algorithm [15]) and many others. Of these, CAMSHIFT stands out as the fastest and simplest. CAMSHIFT was designed for close range face tracking from a stationary camera but has since been modified for a variety of other tracking situations[16], [17].

Robust and flexible tracking algorithms, requiring minimal training and computational resources, are highly desirable for

R. Stolkin is with Center for Maritime Systems, Stevens Institute of Technology, Hoboken, NJ , 07030 USA rstolkin@stevens.edu

I. Florescu is with the Department of Mathematical Sciences, Stevens Institute of Technology, Hoboken, NJ , 07030 USA ifloresc@stevens.edu

M. Baron, C. Harrier and B. Kocherov are undergraduate students in the School of Engineering and Sciences, Stevens Institute of Technology, Hoboken, NJ , 07030 USA

applications such as robot vision and wide area surveillance, both of which necessitate moving cameras. Unfortunately CAMSHIFT often fails with camera motion, figure 3, since it relies on a static background model which is unable to adequately represent changing scenery. We address these difficulties by introducing a flexible background representation which can be continuously relearned. The resulting algorithm, which we call the Adaptive Background CAMSHIFT (or ABCshift) algorithm, tracks robustly in two situations where CAMSHIFT fails; firstly with scenery change due to camera motion and secondly when the tracked object moves across regions of background with which it shares significant colors, figures 1–4. Other recent work, [18], also attempts to track the target based on finding image regions that are not only "target-like" but are also different from the local background. This approach appears to be significantly more computationally expensive than our method. It is not clear if the approach has been made to work at real time frame rates or if this method would be appropriate for the very large camera motion, and associated extreme scene changes, which we attempt to tackle in this paper.

Despite its continuous machine learning and relatively sophisticated tracking, ABCshift is surprisingly computationally cheap. We have managed to implement the ABCshift algorithm on a cheap camera endowed with a very simple embedded Arm7 processor capable of only integer level computations and with very small memory. We demonstrate the utility of the algorithm by using it on such a rudimentary platform and we view this work as a small contribution towards creating viable vision guided robots.

II. Bayesian mean shift tracking with color model

For each frame of an image sequence, the meanshift type trackers look at pixels which lie within a subset of the image defined by a search window (represented by the green box in figures 1–4. Each pixel in this window is assigned a probability that it belongs to the tracked object, creating a 2D distribution of object location over a local area of the image. The tracking problem is solved by mean shifting [20], [13] towards the centroid of this distribution to find an improved estimate of the object location. The search window is now repositioned at the new location and the process is iterated until convergence.

The tracked object is modeled as a class conditional color distribution, $\mathbf{P}(C|O)$. Depending on the application, 1D Hue, 3D normalized RGB, 2D normalized RGB, Luv or Lab histograms may all be appropriate choices of color model, the important point being that these are all distributions which return a probability for any pixel color, given that the pixel represents the tracked object. These object distributions can be learned offline from training images, or during initialization, e.g. from an area which has been user designated as object in the first image of the sequence. Alternatively, the ABCShift algorithm can be initialized by using a stationary camera with a background subtraction approach (e.g. [1],[2], [3]) to detect and learn the color features of a new moving target.

The object location probabilities can now be computed for each pixel using Bayes' law as:

$$\mathbf{P}(O|C) = \frac{\mathbf{P}(C|O)\mathbf{P}(O)}{\mathbf{P}(C)} \tag{1}$$

where $\mathbf{P}(O|C)$ denotes the probability that the pixel represents the tracked object given its color, $\mathbf{P}(C|O)$ is the color model learned for the tracked object and $\mathbf{P}(O)$ and $\mathbf{P}(C)$ are the prior probabilities that the pixel represents object and has the color C respectively.

The denominator of equation (1) can be expanded as:

$$\mathbf{P}(C) = \mathbf{P}(C|O)\mathbf{P}(O) + \mathbf{P}(C|B)\mathbf{P}(B) \tag{2}$$

where $\mathbf{P}(B)$ denotes the probability that the pixel represents background.

Bradski[11], [12] recommends values of 0.5 for both $\mathbf{P}(O)$ and $\mathbf{P}(B)$. We find this choice difficult to justify since we take these terms to denote the expected fractions of the total search window area containing object and background pixels respectively. Hence we assign values to object priors in proportion to their expected image areas. If the search window is resized to be r times bigger than the estimated tracked object area, then $\mathbf{P}(O)$ is assigned the value $1/r$ and $\mathbf{P}(B)$ is assigned the value $(r-1)/r$.

Bradski[11], [12] suggests learning the expression (2) offline (presumably building a static $\mathbf{P}(C|B)$ histogram from an initial image). While it is often reasonable to maintain a static distribution for the tracked object (since objects are not expected to change color), a static background model is unrealistic when the camera moves. The CAMSHIFT algorithm can rapidly fail when the background scenery changes since colors may exist in the new scene which did not exist in the original distribution, such that the expressions in Bayes law will no longer hold true and calculated probabilities no longer add up to unity.

Particular problems arise with CAMSHIFT if the tracked object moves across a region of background with which it shares a significant color. Now a large region of background may easily become mistaken for the tracked object, figure 1.

III. Adapting the background

We address these problems by using a background model which can be continuously relearned. Rather than using an explicit $\mathbf{P}(C|B)$ histogram, we build a $\mathbf{P}(C)$ histogram (of all pixels within the search window) which is recomputed every time the search window is moved. $\mathbf{P}(C)$ values, looked up in this continuously relearned histogram, can now be substituted as the denominator for the Bayes' law expression, equation (1), for any pixel. Since the object distribution, $\mathbf{P}(C|O)$, remains static, this process becomes equivalent to implicitly relearning the background distribution, $\mathbf{P}(C|B)$, because $\mathbf{P}(C)$ is composed of a weighted combination of both these distributions, equation (2). Relearning the whole of $\mathbf{P}(C)$, rather than explicitly relearning $\mathbf{P}(C|B)$, helps

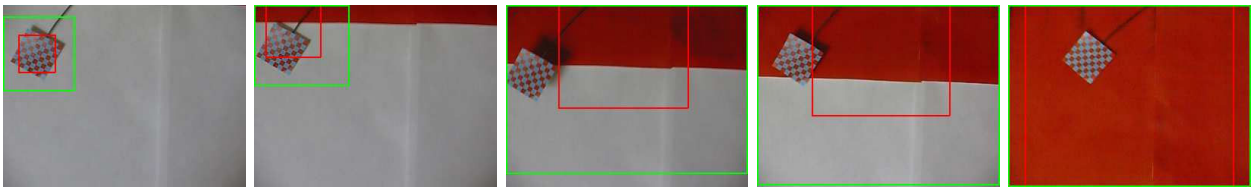

Fig. 1. A simple blue and red checkered object, moving from a region of white background into a region of red background. CAMSHIFT fails as soon as the object moves against a background with which it shares a common color. Frames 350, 360, 380, 400, and 450 shown. Green and red squares indicate the search window and estimated object size respectively. This movie, RedWhite1CAMSHIFT.avi, can be viewed at our website[19].

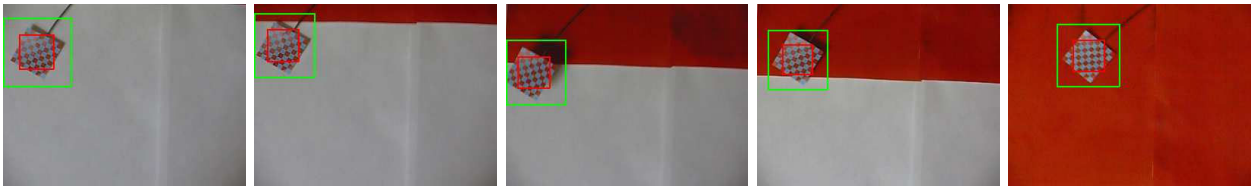

Fig. 2. ABCshift tracks successfully. Frames 350, 360, 380, 400, and 450 shown. Green and red squares indicate the search window and estimated object size respectively. This movie, RedWhite1ABCshift.avi, can be viewed at our website[19].

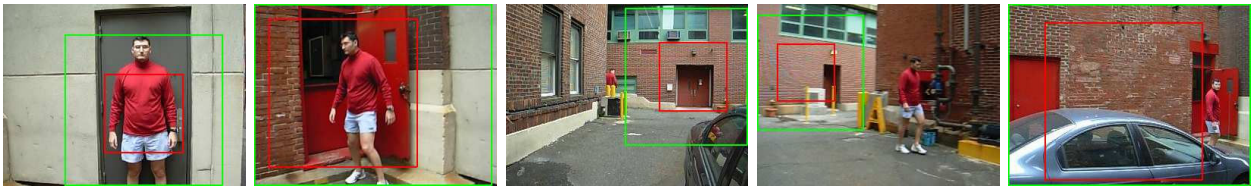

Fig. 3. Person tracking with CAMSHIFT from a moving camera in a cluttered, outdoors environment. Frames 1, 176, 735, 1631, and 1862 shown. Since the tracked person wears a red shirt, CAMSHIFT fixates on red regions of background, including brick walls and doors, and repeatedly loses the tracked person. Green and red squares indicate the search window and estimated object size respectively. This movie, PeopleTracking1CAMSHIFT.avi, can be viewed at our website[19].

Fig. 4. ABCshift successfully tracks throughout the sequence and is not distracted by red regions of background, despite being initialised in image 1 which contains no red background. Frames 1, 176, 735, 1631, and 1862 shown. Green and red squares indicate the search window and estimated object size respectively. This movie, PeopleTracking1ABCshift.avi, can be viewed at our website[19].

ensure that probabilities add up to unity (e.g. if there are small errors in the static object model).

Adaptively relearning the background distribution helps prevent tracking failure when the background scene changes, particularly useful when tracking from a moving camera, figure 4 on page 3. Additionally, it enables objects to be tracked, even when they move across regions of background which are the same color as a significant portion of the object, figure 2 on page 3. This is because, once $\mathbf{P}(C)$ has been relearned, the denominator of Bayes' law, equation (1), ensures that the importance of this color will be diminished. In other words, the tracker will adaptively learn to ignore object colors which are similar to the background and instead tend to focus on those colors of the object which are most dissimilar to whatever background is currently in view.

It is interesting to note that the continual relearning of the $\mathbf{P}(C)$ histogram need not substantially increase computational expense. Once the histogram has been learned for the first image it is only necessary to remove from the histogram those pixels which have left the search window area, and add in those pixels which have newly been encompassed by the search window as it shifts with each iteration. Provided the object motion is reasonably slow relative to the camera frame rate, the search window motion will be small, so that at each iteration only a few lines of pixels need be removed from and added to the $\mathbf{P}(C)$ histogram.

If the $\mathbf{P}(C)$ histogram is relearned only once every frame, the speed should be similar to that of CAMSHIFT. However, if the histogram is relearned at every iteration, some additional computational expense is incurred, since to properly exploit the new information it is necessary to recompute the $\mathbf{P}(O|C)$ values for every pixel, including those already analyzed in previous iterations. Theoretically, updating at each iteration should produce more reliable

tracking, although we have observed good tracking results with both options.

In practice, ABCshift often runs significantly faster than CAMSHIFT. Firstly, the poor background model can cause CAMSHIFT to need more iterations to converge. Secondly, the less accurate tracking of CAMSHIFT causes it to automatically grow a larger search window area, so that far greater numbers of pixels must be handled in each calculation.

IV. Finding the centroid position

At the beginning of each iteration we calculate the probabilities for each pixel in the search window, of representing the tracked object, according to the formula (1). Then we calculate:

$$M_0 = \sum_i P(O|C = c_i), \tag{3}$$

where i runs through all the pixels in the search window and c_i is the color of respective pixel i. Then we calculate the position of the centroid at the current iteration according to:

$$\begin{cases} x_c & = \frac{1}{M_0} \sum_i x_i P(O|C = c_i) \\ y_c & = \frac{1}{M_0} \sum_i y_i P(O|C = c_i), \end{cases} \tag{4}$$

where (x_i, y_i) is the position of pixel i in the search window. At the end of the iteration the center of the search window is shifted to the new position (x_c, y_c) and the procedure is repeated until two consecutive center positions are within ε of each other. This final position is the output center position of the object in the current frame. This is in fact the mathematical expectation of the object location, conditional on the color information within the new image frame.

The ABCshift algorithm is summarised as:

1) *Identify an object region in the first image and train the object model,* $\mathbf{P}(C|O)$.
2) *Center the search window on the estimated object centroid and resize it to have an area* r *times greater than the estimated object size.*
3) *Learn the color distribution,* $\mathbf{P}(C)$*, by building a histogram of the colors of all pixels within the search window.*
4) *Use Bayes' law, equation* (1)*, to assign object probabilities,* $\mathbf{P}(O|C)$*, to every pixel in the search window, creating a 2D distribution of object location.*
5) *Estimate the new object position as the centroid of this distribution and estimate the new object size (in pixels) as the sum of all pixel probabilities within the search window.*
6) *Repeat steps 2-6 until the object position estimate converges.*
7) *Return to step 2 for the next image frame.*

V. Application of the ABCshift algorithm to robotics

The algorithm we created adapts itself to the background changes, therefore it would be particularly suitable to specific applications where the camera is moving. Such applications might include robot vision tasks such as automatic servoing of a motorized pan-tilt camera platform or vision based guidance of a robotic vehicle where due to the construction the camera is moving constantly. We describe next a conventional approach and a different one where we make full use of the previously described center positioning method.

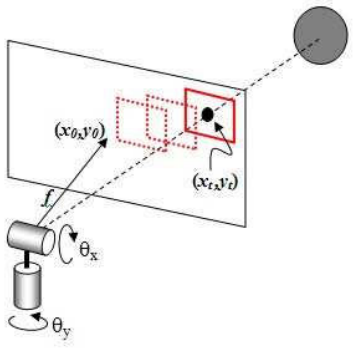

Fig. 5. *Conventional pan tilt servoing using ABCshift.* After several iterations the algorithm converges on a new object centroid and the camera is redirected to point in this direction.

A. Conventional servoing for pan/tilt tracking

Once a vision system is capable of tracking an object, it is a relatively simple matter to servo a motorized pan/tilt platform to keep the camera targeted on the object as it moves. Conventional servoing with the ABCshift tracker would involve multiple iterations of shifting the object window or template region across the image until convergence, and then repositioning the camera so that its optical axis is pointed directly towards the final estimate of the object centroid (see figure 5).

If sophisticated hardware is available, with accurate rotation sensors and either spare processing power or dedicated control circuits, then a conventional approach such as PID control should work well for controlling the motors. Instead, we attempt to use extremely cheap and simple hardware, such as $10 hobbyist servo motors, partly due to our lack of budget and partly to demonstrate the robustness of the vision algorithm. We have thus made use of a simple speed control rule:

$$\dot{\theta}_x \propto \arctan\left(\frac{x_t - x_0}{f}\right)$$
$$\dot{\theta}_y \propto \arctan\left(\frac{y_t - y_0}{f}\right),$$

where (x_t, y_t) are the converged centroid position of the moving target in image coordinates, (x_0, y_0) is the center of the image and θ_x and θ_y are angles of rotation about motor axes which are aligned with the optical center of the camera. If the focal length of the camera is unknown, then these equations can be reasonably approximated with:

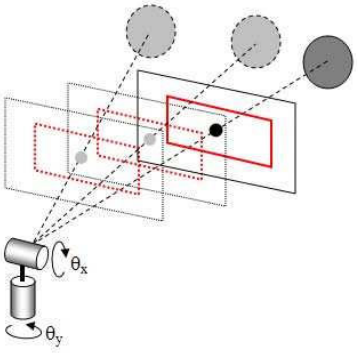

Fig. 6. *Meanshift servoing.* Now only one ABCshift iteration is performed for each frame. The camera itself is meanshifted with a search window which remains centered in the middle of each image.

$$\dot{\theta}_x \propto (x_t - x_0)$$
$$\dot{\theta}_y \propto (y_t - y_0)\,,$$

since angles are typically small, and this avoids the need for any camera calibration.

B. Meanshift servoing

Unfortunately, when attempting real time tracking on a very small embedded system, especially with the kind of continuous machine learning inherent in the ABCshift tracker, the cpu may not be fast enough to process more than one meanshift iteration per frame and still achieve useful frame rates. Instead we need to find a way of minimizing the image analysis for each frame. We solve this problem with a simple technique which we call "mean shift servoing".

Now, the ABCshift search window remains permanently fixed in the center of the image. Instead of iteratively shifting the search window to be centered on the expected target centroid, we simply use motors to move the entire camera (see figure 6). In effect, we have saved cpu time by devolving effort from the vision system to the motors. We need only use the center part of each image and the remaining portion can be discarded. Effectively, this reduced image has become the ABCshift search window, and, in essence, we are now meanshifting the entire camera over a motion space rather than meanshifting a window inside an image. For added robustness, while conserving efficiency, the size of this centered search window area can be updated at each frame to be proportional to the speed of the moving target.

Summary of mean shift servoing procedure:

1) *Define a search region to be centered at the middle of the image (i.e. the point of intersection of the optic axis with the image plane).*
2) *Assign target probabilities to all pixels in the search region according to the Bayes law of equation 1 on page 2.*
3) *Find the centroid of this distribution according to equation* (4)*.*
4) *Move the camera to be centered at this centroid position.*
5) *Capture new image and repeat.*

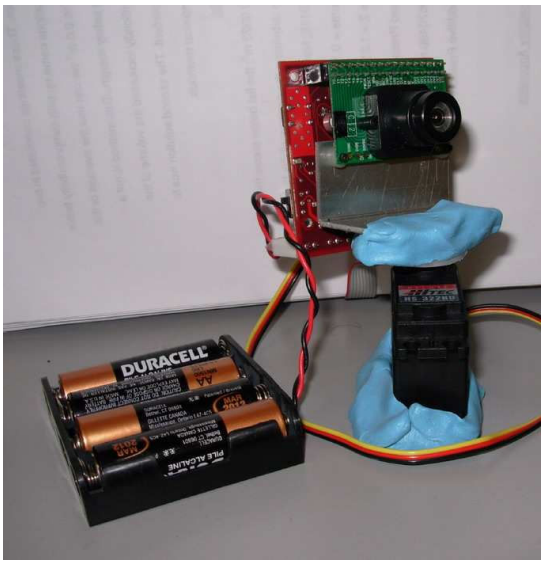

Fig. 7. CMUcam mounted on servo motor for automatic panning using ABCshift tracking algorithm and meanshift servoing procedure. With ABCshift, robust tracking with a moving camera can be performed with very little computational expense.

Note that in this case also, a simple speed control rule, similar to that described above can be used. Note also that this servoing technique should work for other meanshift based tracking techniques such as [12] or [13].

VI. Proof of concept

We have implemented the meanshift servoing technique with the ABCshift tracking algorithm, on the tiny *CMUcam3* embedded vision system [21], incorporating a small *Arm7* processor which has very limited memory and can only run integer arithmetic C programs (figure 7). At present, this system is able to track objects of around 50 pixels by 50 pixels at 176×144 resolution, at around 6 frames per second, while continuously relearning the background model, and can control a servo motor to automatically pan the camera to follow a moving target. We hope to improve this frame rate substantially in the near future by finding more efficient ways to access the frame buffer. Using the conventional servoing approach, with multiple mean-shift iterations per frame (section A), we have also tracked/servoed comfortably at 30fps using a laptop computer and USB webcam (figure 8). For simplicity of coding, we currently relearn every pixel of the $\mathbf{P}(C)$ histogram at every iteration, instead of only relearning the small number of pixels that change with each small shift of the search window. Since there are typically four such shifts per frame, we hope to achieve better than 100fps once this is re-coded more efficiently.

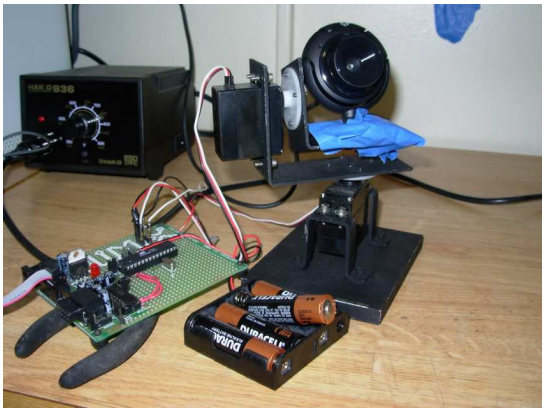

Fig. 8. USB webcam mounted on servo motors for automatic tilt-pan using ABCshift tracking algorithm and conventional servoing procedure. For servoing movie please see the video associated with this paper

VII. Conclusions and extensions.

In this article we presented a novel, simple yet robust algorithm, the ABCshift which can track objects in the presence of background changes. We implement the algorithm on a very rudimentary platform to demonstrate its capability to work even with very small CPU designs. We present a new method of steering the platform (meanshift servoing) to continuously follow the object tracked which is a direct result of the simplicity of the algorithm.

Future work will concentrate in both developing further the algorithm as well as modifying the platform and the mechanical issues involved.

In terms of algorithmic developments, we are currently investigating a way to resize the search window to cope with variations in size of the object, this in turn will provide better performance (by processing only as many pixels as are needed) as well as a better estimate of the object itself and the changes that may subsequently occur. The CAMSHIFT algorithm which is designed for situations when the background does not change interprets the quantity M_0 in (3) as the expected number of pixels that contains the object and then re-scales the search window accordingly. In our situation, because of the innovation we introduced in order to adapt to the changes in the background distribution as the camera moves makes M_0 lose its nice interpretation and introduce numerical instabilities with time. The video associated with figure 4 shows our partially successful attempts to employ this kind of resizing while also correcting for some of the sources of instability (see [22]). This work is ongoing and will be the subject of future papers.

Related with this last part we are also investigating ways of relearning the tracked object's color distribution, in addition to relearning the background distribution. This will allow us to cope with situations when the object changes its color distribution (due to changes in illumination, viewpoint, etc.) thus improving the robustness of the algorithm. However, this improvement will no doubt come at additional computational expense.

In terms of future robotics applications of ABCshift, we plan to mount a pan tilt unit on a mobile robot platform and explore combinations of ABCshift with steering algorithms to enable a robotic vehicle to follow a tracked object.

References

[1] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson, "Advances in cooperative multi-sensor video surveillance," in *Proceedings of the 1998 DARPA Image Understanding Workshop*, vol. 1, 1998, pp. 3–24.

[2] W. Grimson, C. Stauffer, R. Romano, and L. Lee, "Using adaptive tracking to classify and monitor activities in a site," *Computer Vision and Pattern Recognition*, June 1998.

[3] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proceedings of the IEEE Frame Rate Workshop*, 1999.

[4] D. Murray, P. McLauchlan, I. Reid, and P. Sharkey, "Reactions to peripheral image motion using a head/eye platform," *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pp. 403–411, 11-14 May 1993.

[5] K. Daniilidis, C. Krauss, M. Hansen, and G. Sommer, "Real-time tracking of moving objects with an active camera," *Real-Time Imaging*, vol. 4, no. 1, pp. 3–20, 1998.

[6] E. Malis and S. Benhimane, "A unified approach to visual tracking and servoing," *Robotics and Autonomous Systems*, vol. 52, no. 1, pp. 39–52, 2005.

[7] T. Drummond and R. Cipolla, "Real-time tracking of complex structures with on-line camera calibration," in *Proc. British Machine Vision Conference*, 1999.

[8] R. Stolkin, A. Greig, J. Gilby, and M. Hodgetts, "An em / e-mrf algorithm for adaptive model based tracking in extremely poor visibility," *Journal of Image and Vision Computing*, p. in press, 2007.

[9] T. Drummond and R. Cipolla, "Real-time tracking of multiple articulated structures in multiple views," in *European Conference on Computer Vision*, 2000.

[10] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2000.

[11] G. R. Bradski, "Computer video face tracking for use in a perceptual user interface," *Intel Technology Journal*, Q2 1998.

[12] ——, "Real time face and object tracking as a component of a perceptual user interface," in *Proc. 4th IEEE Workshop Applications of Computer Vision*, 1998, pp. 214–219.

[13] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 564–577, 2003.

[14] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models." *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.

[15] A. B. M. Isard, "Condensation-conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[16] J. G. Allen, R. Y. D. Xu, and J. S. Jin, "Object tracking using camshift algorithm and multiple quantized feature spaces," in *Proceedings of the Pan-Sydney area workshop on Visual information processing*, ser. ACM International Conference Proceeding Series, vol. 100. Darlinghurst, Australia: Australian Computer Society, Inc., 2004, pp. 3–7.

[17] N. Liu and B. C. Lovell, "Mmx-accelerated real-time hand tracking system," in *IVCNZ 2001*, Dunedin, New Zealand, 2001, pp. 381–385.

[18] H. T. Nguyen and A. W. Smeulders, "Robust tracking using foreground-background texture discrimination," *Int. J. Comput. Vision*, vol. 69, no. 3, pp. 277–293, 2006.

[19] "http://www.math.stevens.edu/~ifloresc/ABCshift.htm."

[20] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[21] A. Rowe, A. Goode, D. Goel, and I. Nourbakhsh, "CMUcam3: An open programmable embedded vision sensor," Robotics Institute, Carnegie Mellon University, technical report CMU-RI-TR-07-13, 2007.

[22] R. Stolkin, I. Florescu, and G. Kamberov, "An adaptive background model for camshift tracking with a moving camera," in *Advances In Pattern Recognition*, ser. Proceedings of the Sixth International Conference, Indian Statistical Institute, Kolkata, India, 2007, pp. 147–151.