An Adaptive Model for Tracking Objects by Color

Christopher Rasmussen and Gregory D. Hager

Department of Computer Science
Yale University
51 Prospect Street
New Haven, CT 06520-8285

`rasmuss@powered.cs.yale.edu`, `hager@cs.yale.edu`

Abstract

This paper discusses an adaptive approach to tracking objects based primarily on their color. By limiting our attention to areas of approximately uniform reflectance on nearly Lambertian surfaces, we can assume that pixels corresponding to surface patches on the object form a linear cluster in color space. Performing principal components analysis on user-selected sample pixels parametrizes an ellipsoidal model of this distribution that can be used to quickly and robustly track many objects through a range of orientations and scales. The apparent color and intensity of an object may change as it moves, though; we present a technique for tracking an object's color distribution through color space while tracking it in the image. Finally, by using weak spatial information we demonstrate successful tracking in the presence of similar-colored background distractors. In combination with a tracking failure recovery procedure, the basic technique has been successfully applied to tasks as diverse as head and hand tracking, following juggled and thrown balls, and constructing vision-based input devices.

1 Introduction

Tracking is a common visual task with many uses. By maintaining focus on someone's face as they walk and talk, we can continually read their facial expressions; by following a tennis ball off of an opponent's racquet and across the net, we can position ourselves properly for a return; by keeping our eye on a doorknob as we approach it, we can ensure that we grasp it smoothly when it is within reach.

The difficult problem in visual tracking is performing fast and reliable matching of the target from frame to frame. A large variety of tracking techniques and algorithms have been reported in the literature, but most tracking algorithms ultimately rely on one of three low-level image processing techniques: edge extraction, variations on region-based correlation, or simple segmentation techniques, often referred to as "blob" tracking. Although edge-based and region-based techniques are generally more accurate and robust than segmentation-based techniques, they also incur a much higher computational cost. Conversely, blob tracking is computationally efficient and practical, particularly as standard desktop processors have become more powerful. This has led to it becoming the preferred method of performing tracking for many practical applications.

Traditionally, blob tracking has been implemented on gray-scale images using thresholding or simple functions of gray-scale value [1, 2, 4, 12]. These techniques usually rely on a structured environment (e.g., a black backdrop) so that the target (e.g., a white ping-pong ball) is unique. With the advent of inexpensive color cameras and digitizers, it seems likely that color-based thresholding techniques will provide a practical and robust alternative.

To date, little has been published on practical color tracking algorithms. A simple color tracking technique is presented in Du and Crisman [5]. They pick a set of arbitrary "categorical" colors in RGB space and construct membership volumes for each one based on nearest neighbor calculation. Objects are characterized by their histograms over these volumes, permitting multi-colored regions to be tracked. It is unclear what update function is used to determine where to move the tracking window, and the results are based only on synthetic, abstract image sequences.

Pfinder [17] and Perseus [8] are specialized systems for tracking people. Pfinder uses a statistical characterization of color variation in a static image to perform color-based change detection; Perseus uses color histograms (as one of a suite of "feature maps"). Prior terms on flesh color assist the systems in automatically finding skin and "bootstrapping" a model whenever someone appears in the scene. Models for body parts and constraints governing their interaction contribute to a high-level manager of the tracking primitives. To achieve its results, Pfinder does complete low-resolution (subsampled to 160 x 120 pixels) analysis of every frame, and runs at 10 Hz on an SGI Indy; Perseus uses DataCube vision hardware to work at frame-rate. Pfinder depends on a static background for accurate segmentation, ruling out camera movement.

In this paper, we describe a general-purpose technique for color blob tracking within the XVision real-time vision software package [7]. The primary attributes of our approach are that it does not depend on a fixed camera, it uses an empirical sampling of the color space and a physically plausible model of color variation, and it exhibits fast performance on standard workstations. The technique is general enough to apply to varied problem domains, and as a primitive is suitable for the construction of more complicated systems.

2 Background

The principle issue in any blob tracking algorithm is to perform an accurate segmentation of a set of candidate pixels into those that belong to the target and those that do not. In principle, any standard color segmentation algorithm could be used to solve this problem. However, most such algorithms attempt to segment an entire static images with little or no *a priori* knowledge of the color distribution over the scene. Typically, a high priority is placed on finding accurate borders for color regions in order to provide useful information for subsequent stages of processing.

The task of tracking an object presents a different set of demands. First, although the background has an unknown color distribution, we can assume that the object to be tracked is initially completely known in a color sense. Second, real-time considerations limit the complexity of computation that can be performed on each newly-acquired image, although the initialization stage can often be leisurely. Finally, because the primary goal of tracking is object localization, completely accurate segmentation is not as important as eliminating false positives. It is acceptable to be conservative and only classify part of the object as the object, as long as a large enough fraction is found to be distinguishable from noise.

The theoretical basis for our segmentation algorithm is given by Klinker et al. [9]. They present a thorough analysis of the physics of object color properties in the context of an off-line approach to segmenting unknown colors in a scene. In particular, they show that the sample color distribution of an object with both Lambertian and specular reflectance components should be a tube-like cluster about the ideal ray (the Lambertian component), with branches or curvature toward light source colors (the specular component). Our own empirical observations of a variety of naturally occuring objects have confirmed these findings.

For our work, we assume that the reflectance of an object is primarily Lambertian and that its surface is of nearly uniform color. Theoretically, we would then expect that the color distribution over the object is a scattering of points in RGB space lying somewhere along a ray from the origin (black) toward the intrinsic color of the object. In practice, though, this is not the case due to object color variation, some specularity, and camera noise. However, by choosing objects with minimal specularity or choosing color sample locations on them away from specularities, we have found that a single tubal cluster often captures color variation very well. We model this cluster by doing principal components analysis (PCA) [11] on a user-chosen sample color distribution. The results give the tracker a bounding ellipsoid for determining pixel membership in the object's dominant color.

This approach can easily be extended to handle objects composed of multiple uniform color patches. With knowledge of which pixel belongs to which color, we can partition the object into its constituent colors and fit an ellipsoid to each one. Object membership is the union of the computed ellipsoids.

Our assumption of uniform coloration and Lambertianness is an adequate approximation for many objects, but only up to a point. In particular, the orientation and location of clusters in color space corresponding to objects depends on the intensity and spectral composition of the light incident upon them. Furthermore, the mixture of light emanating from radiant surfaces in the environment and ultimately entering the camera may change as an object translates or simply rotates in place. For example, objects may move in and out of shadows; they may rotate and change their apparent shading; and they can move away from a window, where the light is very yellow, toward a brick wall that is reflecting much redder light. All of these effects cause the apparent color of the object to change, possibly rendering a static color model formulated at one time, in one place invalid during subsequent tracking. Research on color constancy is addressed toward solving this problem, but has not advanced far enough to be applicable to most real-world situations [6].

We can often overcome this problem by exploiting an attribute of the tracking domain: as an object moves, its apparent color varies more or less continuously. This property is very different from how most color constancy problems are posed, as discontinuities in light composition falling on a fixed scene. Such an assumption may be violated in environments with hard shadows or focused light sources, but in most environments these are rare. When it holds, though the original ellipsoid may cease to be an object's best color model from frame to frame, the best ellipsoid for the next frame is likely nearby in color space. This property of relatively slow change suggests that tracking an object in color space as well as in image space may be a feasible way to achieve the equivalent of color constancy.

Moving the ellipsoid around in color space is reasonable for clusters that are well separated from those of other colors in the image, but abutting clusters present a problem. If a nearby cluster is denser than the one associated with the object being tracked, adaptation may lead to the membership ellipsoid slipping onto it, causing mistracking. This problem, distraction, is similar to what correlation-based tracking methods can encounter in image space. Various techniques might be employed to detect nearby clusters in color space and somehow evade them, but often the clusters overlap enough as to be virtually indistinguishable. This problem can be largely avoided by appealing to the spatial characteristics of the object to differentiate it from the background.

In this paper we will discuss three related color tracking algorithms. All are predicated on a PCA-based notion of color membership. The first, which we will call static color model tracking (**SC**), uses the PCA-derived ellipsoid directly for classifying pixels by color and repositioning a tracking window to stay centered on the object. The second, adaptive color model tracking (**AC**), is **SC** plus the ability to continue tracking an object despite changing illuminant intensity and color composition. Finally, adaptive, distraction-resistant color model tracking (**ADC**) is **AC** plus a weak spatial constraint to help disambiguate similarly-colored background pixels that might otherwise be erroneously classified as part of the object. We will outline how these methods differ in the next section, and detail the capabilities of all of them in the Results, as well as the situations in which the first two break.

3 Methods

In our formulation, there is one *tracker* associated with each *object* to be tracked. An object is a three-dimensional entity that projects onto a spatially compact set, or region, of similarly colored pixels in the image. A tracker is a process that attempts to position a subwindow in the image at every time step such that the center of the object remains inside it. The tracker is bound to its object initially by giving it a sample of the object's color and the object's image location. The tracker formulates a model of the object's color, positions its window at the initial image location, and repeatedly classifies pixels in the window as either matching its color model or not. Those pixels which match the model are interpreted as belonging to the object and are called its *support* after [17]; the tracking window is repositioned to be centered on them.

3.1 Initialization

Initial information about an object $\mathcal{O}_i$ is supplied to the tracker through human agency. A user indicates the object to be tracked by pointing at different parts of it on a live video display of the camera's view and clicking each time to record a color sample. Each click grabs a 16 x 16 pixel subimage centered at the pointer location. The aggregate sampled pixels' color values (minus any saturated pixels) form a distribution in RGB space from which the color model of $\mathcal{O}_i$ is derived using PCA.

The ellipsoid computed for an object $\mathcal{O}_i$ can be described mathematically as a scaling $\mathbf{S}_i$, rotation $\mathbf{R}_i$, and translation $\mathbf{T}_i$ of the unit sphere in RGB space. A particular image pixel $\mathbf{p}_j = (x_j, y_j, R_j, G_j, B_j)^T$ has both a spatial and a color component. It is in the color membership ellipsoid of $\mathcal{O}_i$ if the following function $\mathcal{C}_i$ maps it to 1:

$$\mathcal{C}_i(\mathbf{p}_j) = \begin{cases} 1 & \text{if } \|\mathbf{S}_i^{-1}\mathbf{R}_i^{-1}((R_j, G_j, B_j)^T - \mathbf{T}_i)\| \leq \rho \\ 0 & \text{otherwise} \end{cases}$$

ρ indicates how many standard deviations of the original sampled points the ellipsoid accounts for. Adjusting ρ allows the tracker to be more or less conservative in its classification of candidate points; we used $\rho = 2$ as a threshold for the results presented in this paper.

One to three clicks capturing the extremes of a gradient of intensity in the object generally suffice. Choosing samples appropriately is most important in **SC**, because we want the ellipsoid to be large enough to accomodate some color variation as $\mathcal{O}_i$ moves, yet not so large that other colors are incorrectly classified as the same. For **AC** and **ADC**, the color sample should be chosen only to avoid a very tight cluster, because the volume of the computed ellipsoid is analagous to the size of the tracking window in that if it is too small, even slow color shifts may be too quick to track.

3.2 Update

SC Tracking an object $\mathcal{O}_i$ consists of updating the image location (x_i, y_i) of a tracking window $\mathcal{W}_i$, where $\mathcal{W}_i$ has a fixed height and width (here $h_i = w_i = 128$ pixels). At tracking cycle $t = 0$, we initialize tracking of $\mathcal{O}_i$ by centering $\mathcal{W}_i^0$ on the mean image location of the color sample clicks. For subsequent tracking cycles, every pixel $\mathbf{p}_j$ in $\mathcal{W}_i^t$ is evaluated by $\mathcal{C}_i$; the set of all such $\mathbf{p}_j$ for which $\mathcal{C}_i(\mathbf{p}_j) = 1$ constitutes the object's color support $\mathcal{CS}_i^t$ at time t. In the **SC** algorithm, the mean image location of the pixels in $\mathcal{CS}_i^t$ is used as the center of $\mathcal{W}_i^{t+1}$. For reasons detailed in the Results section below, no attempt is made to vary the size of the tracking window in order to estimate object scaling.

AC The **AC** algorithm tracks the color distribution of an object $\mathcal{O}_i$ as it changes by updating the position and orientation of the membership ellipsoid during each tracking cycle. An intuition for how to do this stems from the observation that the internal distribution of pixels in $\mathcal{CS}_i^t$ may be asymmetric; for instance, many of the pixels may be clustered at one of the ellipsoid, leaving the other end sparse. This can be interpreted as a signal that the color distribution is moving in the direction of the density. We capture this notion by rerunning PCA on $\mathcal{CS}_i^t$ to get new estimates of the ellipsoid's orientation $\mathbf{R}_i^{t+1}$ and position $\mathbf{T}_i^{t+1}$ ($\mathbf{S}_i$ is ignored because using it may lead to instability). These new estimates may not be optimal, but if the color distribution is not moving too quickly, they will converge over the course of a few tracking cycles to a stable extremum. These new transformation matrices are then used in the next cycle by plugging them back into the color membership function to get $\mathcal{C}_i^{t+1}$.

ADC The above approach works well as long as there are few *distractors*—pixels in $\mathcal{W}_i$ that are not part of $\mathcal{O}_i$, but whose color is similar enough that they are part of $\mathcal{CS}_i$. Unfortunately, when $\mathcal{O}_i$ is not highly saturated, such as when it is someone's face, there are usually a number of nearby colors in the image. Empirically, we have found that distractors are often unconnected or clumped in small groups around the periphery of the tracking window, while the object is relatively compact, large, and centered. In the **ADC** algorithm, we make use of this fact to filter $\mathcal{CS}_i$ by taking its largest connected component and using only members of it to update $\mathcal{C}_i$ and $\mathcal{W}_i$. We have also experimented with using PCA on the spatial components of the members of $\mathcal{CS}_i$ to get an image-spatial membership ellipse. This would accomplish the same sort of filtering, but it is sensitive to outliers and is often too inclusive. Throwing out all pixels but those in the largest connected component is a crude but effective way to deal with spatial outliers efficiently.

3.3 Recovery

Occasionally a tracker will lose its object $\mathcal{O}_i$ by allowing it outside $\mathcal{W}_i$ due to excessive object speed or occlusion. We call $\mathcal{O}_i$ "lost" when the size of $\mathcal{CS}_i$ falls below an aggregate image area threshold α ($\alpha = 1$ here). A tracker can often recover from such events by resampling the entire image at a lower resolution and performing a global search [15]. Using 8 x 8 subsampling, a global search incurs about 5 times the computation of a normal tracking cycle. Though this causes the update rate to dip, we have found this to be a rare and therefore acceptable cost. Pixels satisfying $\mathcal{C}_i$ in this search are weighted according to formula (1), below, where L is the point of loss. (1) approximates winner-take-all clustering to decide where to saccade; nearness to L promotes stability in that an object lost due to speed is likely to be reacquired if there are distractions elsewhere in the image.

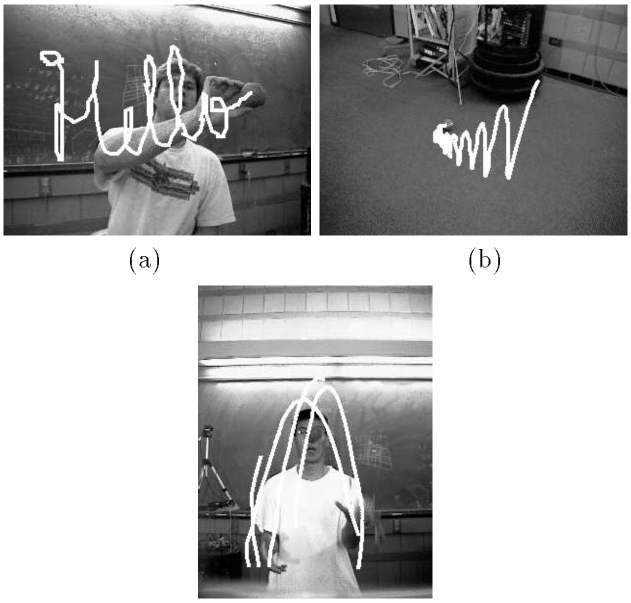
For all pixels $\mathbf{p}_j$ that are in $\mathcal{CS}_i$ over the entire image, let $win(j)$ be the set of pixels both in the support and also lying within a hypothetical h_i x w_i tracking window centered at $\mathbf{p}_j$; let $win_k(j)$ be the image location of the kth pixel in this set. A heuristic for the best candidate location for the lost object is given by:

$$best = \operatorname*{argmax}_j \sum_k \frac{1}{1 + ||win_k(j) - L||} \tag{1}$$

If the size of $win(best)$ is large enough to satisfy the area threshold α, then tracking is restarted at p_{best}. Otherwise, the search is repeated (as a thread if other objects are still being tracked).

4 Results

The three trackers that we present are more qualitative than precise. No explicit information about the scale or orientation of $\mathcal{O}_i$ is generated as a by-product of the update cycle, as is the case with many correlation-based methods [13]. An approximation to such information can in principle be calculated by fitting an ellipse to $\mathcal{CS}_i$, but it is not guaranteed to be meaningful because of $\mathcal{O}_i$'s three-dimensionality. The size and shape of $\mathcal{CS}_i$ may change as $\mathcal{O}_i$ rotates if it is anything other than a uniformly colored sphere. When tracking a person's face, for example, if the subject turns around (and has short hair) the tracker will



(a) (b)

(c)

Figure 1: (a) Using the ball as a pen; (b) detection of a ball thrown into view and subsequent tracking as it bounces and rolls; (c) a red ball being tracked as it and a blue ball are juggled. The blue ball is in front of the juggler's nose and the red ball is falling into his left hand

slip onto the smaller patch of skin visible on the back of their neck; this could be erroneously interpreted as the subject moving away from the camera.

4.1 Static Color Model

Despite its simplicity, the **SC** tracker is effective for a number of tasks, particularly when the object $\mathcal{O}_i$ is in front of a generally dissimilar background and does not move too far. It works best when $\mathcal{O}_i$ is roughly spherical, like a ball or a human head, so that shading does not change dramatically during rotations. We outline a few applications below.

By keeping a record of the $\mathcal{W}_i$'s consecutive image positions, the tracker gains additional information for use as input to another program. Figure 1(a) shows a red ball being used as a pen. The end of a word can be signalled by concealing the ball in the hand; when it is revealed to start the next word, the loss recovery procedure automatically detects the ball at its new location. We have varied this formula to create a simple painting program in which the user can create their own palette by sampling objects lying around—a green hat becomes a green paintbrush, and so on. An option is to have brush size vary according to object image size, allowing the user to move in three dimensions with varying effects on the canvas.

Dynamic, dextrous robots that use visual input typically assume a simplified visual environment in order to achieve robustness and speed. The juggling robot of [12] and the ping-pong playing robot of [2], for example, require a well-illuminated white ball and a black background for accurate segmentation. This limits the robots to performing only in specially-constructed workspaces with fixed camera views that minimize distraction. The **SC** tracking method alone is fast enough and resistant enough to distraction to offer a vision system for such tasks that can work in front of natural backgrounds. Figure 1(c) shows the tracking of one of two juggled balls. The gaps in the trajectory line are most likely due to the juggler's hands occluding the ball as it is caught, as well as occasional excessive speed; recovery allows tracking to resume at the first available opportunity.

The recovery procedure need not be regarded as stemming from failure; rather, if the disappearance of the tracked object is expected, searching for it while it is gone becomes surveillance. In Figure 1(b), tracking was initialized on a ball which was then removed from view. The ball was thrown into view some time later; the tracker reacquired it quickly and tracked it as it bounced.

4.2 Adaptive Color Model

The **AC** algorithm is good for tracking objects into and out of shadow, or non-spherical objects through rotations that would confuse the **SC** tracker.

The shortcomings of the **SC** algorithm are illustrated in Figure 2. In this demonstration, tracking was initialized on a piece of red paper with one sample click in a bright patch. The paper was kept relatively planar as it was then rotated nearly ninety degrees, changing the surface normal with respect to the viewer considerably and thus reducing the light reflected from it into the camera. The **AC** algorithm was able to track both the paper through image space and its color cluster through color space successfully. From examination of Figures 2(c) and 2(f), the distance travelled by the paper's cluster in color space is great compared to the size of the cluster itself. An **SC** tracker trying to perform the same task would be left behind in color space and lose track of the paper.

4.3 Adaptive, Distraction-Resistant Tracking

The **ADC** method is the best general purpose tracking algorithm of the three we have discussed. It adds the capability of dealing with similar colors in the background that might cause the **AC** method to slip onto another feature.

How distraction can occur is shown in Figure 3.

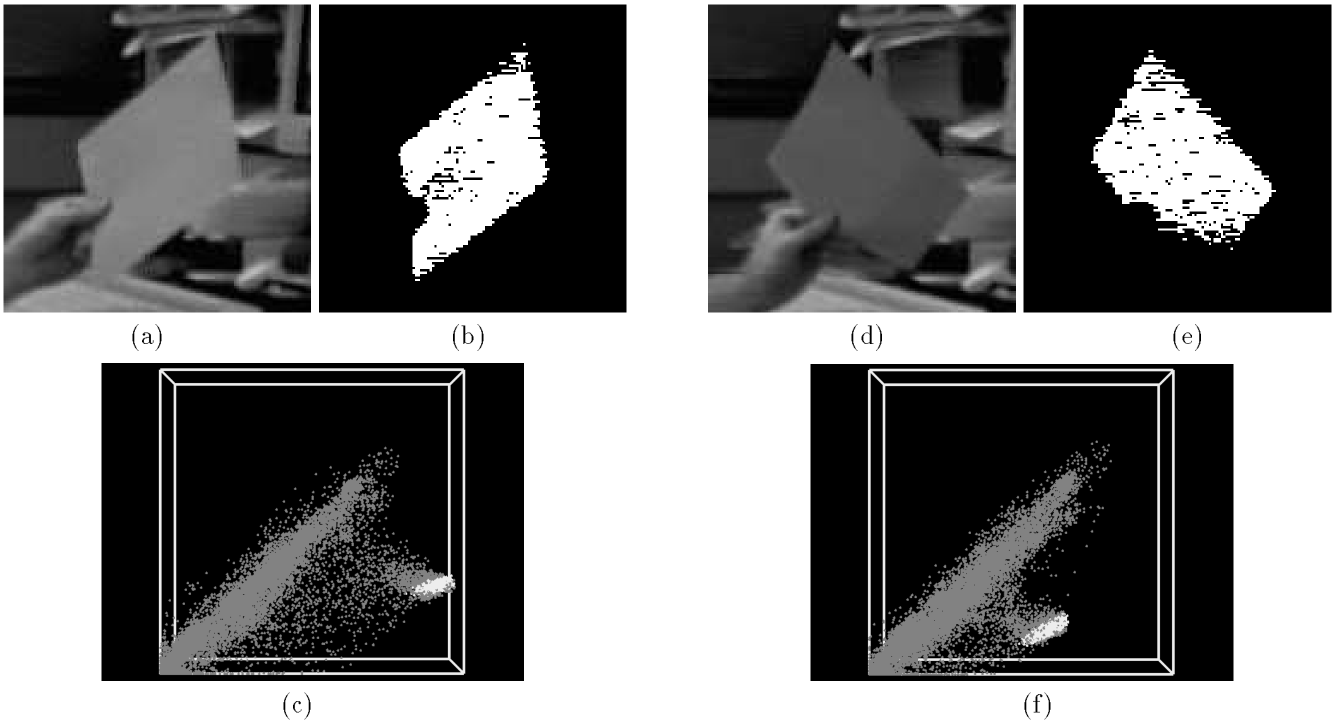


Figure 2: Tracking a piece of red paper as it rotates with **AC**. (a) Initial, brightly lit orientation; (b) Color support; (c) RGB representation of pixels in tracking window. White cluster at lower right is pixels inside ellipsoid; (d) Final, dimly lit orientation; (e) Color support—note that whole paper is still included; (f) Cluster associated with paper has migrated toward the origin, but the ellipsoid has followed it. In (c) and (f), the origin (black) is at the lower left front corner of the cube; the red axis is horizontal, green is vertical, and blue is into the page.

Using the **AC** algorithm to track a face can be problematic because the relative unsaturation of flesh color leaves it close to a number of other common colors in the scene, including hair and some tan brick. When enough distracting pixels enter the tracking window, the ellipsoid is prone to switching to the higher density cluster that they belong to in color space and losing track of the face. This has happened in Figure 3(c). By using **ADC**, the relatively sparsity and disconnectedness of the distractors in image space is counted against them, preventing their proximity in color space from dislodging the tracker from the actual object.

ADC works well for tracking faces in many indoor lighting situations, both from fixed and mobile camera platforms (Pfinder [17] does not allow camera movement). We have had good results with mounting the camera on a pan-tilt unit, allowing the tracker to follow the subject's face over a wide range of poses, positions, and distances in a room. This is accomplished by layering on top of tracking as usual a controller that moves the camera incrementally in the direction that will bring the tracking window back to the center of the 640 x 480 video signal. As long as there are no other human faces that the tracked face goes directly in front of or behind, performance is good. Dealing with this kind of distraction requires a much more sophisticated geometric analysis of the image.

5 Discussion

The ellipsoidal color tracking schemes that we have presented have wide applicability. They are especially notable for their high speed and insensitivity to gross scaling, rotations, or other geometric distortions of the tracked object, as well as changes in perceived object color as it moves about in the presence of non-trivial background distractors. Our purpose in this paper has been to devise a simple, general model and push it as far as possible. We believe that in situations for which they are suited, the techniques outlined demonstrate an ability to perform tracking tasks beyond the reach of most other approaches.

As input to higher level processes, any of the weak-geometry color trackers that we have presented is well suited to being an attentional mechanism [16] that suggests where to perform more sophisticated analyses. The **ADC** algorithm, with its use of connected components, points toward what form such analyses might take. For instance, we might employ a correlation method to choose between multiple same-color patches in the tracking window or as a final test of a candidate location during loss recovery. These situations occur infrequently enough that such hybrid

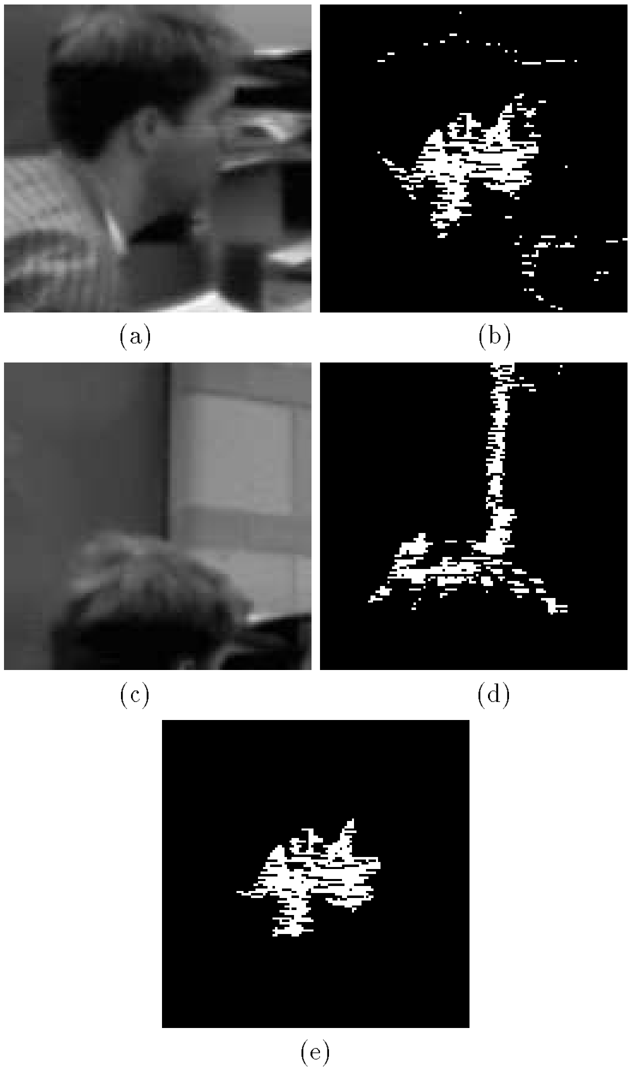


Figure 3: Avoiding distraction when tracking a face. (a) A face in front of a cluttered background; (b) Color support for the face—note the inclusion of hair highlights and some edges from the desk in the lower right; (c) Under **AC** tracking, the cluster in RGB space corresponding to hair has pulled the tracker off the face; (d) Color support for the wrong object; the window will continue up the wall before coming to rest; (e) **ADC**: The object support after keeping only the largest connected component in (b); no mistracking occurs.

tracking would still be much faster than geometry alone while adding robustness. Geometry is prone to the problems (multiple object orientations and scales) that we are avoiding by using just color, of course, but some work has been done on this [3, 10].

References

[1] P. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated Tracking and Grasping of a Moving Object with a Robotic Hand-Eye System. In *IEEE Trans. Robotics and Automation*, Vol. 9, No. 2, pp. 152-165, 1993.

[2] R. Andersson. Understanding and Applying a Robot Ping-Pong Player's Expert Controller. In *ICRA*, pp. 1284-1289, 1989.

[3] M. Black and A. Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. In *ECCV*, pp. 329-342, 1996.

[4] P. Corke. Visual Servoing. In *Visual Control of Robot Manipulators – A Review*, K. Hashimoto, ed., World Scientific, 1993.

[5] Y. Du and J. Crisman. A Color Projection for Fast Generic Target Tracking. In *IROS*, pp. 360-365, 1995.

[6] G. Finlayson, B. Funt, and K. Barnard. Color Constancy Under Varying Illumination. In ICCV, pp. 720-725, 1995.

[7] G. Hager. The 'X-Vision' System: A General-Purpose Substrate for Vision-Based Robotics. In *Workshop on Vision for Robotics*, 1995.

[8] R. Kahn, M. Swain, P. Prokopowicz, and R. Firby. Gesture Recognition Using the Perseus Architecture. To appear in *CVPR*, 1996.

[9] G. Klinker, S. Shafer, and T. Kanade. A Physical Approach to Color Image Understanding. *Int. Journal of Computer Vision*, Vol. 4, pp. 7-38, 1990.

[10] H. Murase and S. Nayar. Visual Learning and Recognition of 3-D Objects from Appearance. *Int. Journal of Computer Vision*, Vol. 14, pp. 5-24, 1995.

[11] E. Oja. *Subspace Methods of Pattern Recognition*, Research Studies Press, 1983.

[12] A. Rizzi, L. Whitcomb, and D. Koditschek. Distributed Real-Time Control of a Spatial Robot Juggler. *IEEE Computer*, Vol. 25, No. 5, pp. 12-23, May, 1992.

[13] J. Shi and C. Tomasi. Good Features to Track. In *CVPR*, pp. 593-600, 1994.

[14] D. Terzopoulos and T. Rabie. Animat Vision: Active Vision in Artificial Animals. In *ICCV*, pp. 801-808, 1995.

[15] K. Toyama and G. Hager. Incremental Focus of Attention for Robust Visual Tracking. To appear in *CVPR*, 1996.

[16] J. Tsotsos, S. Culhane, W. Wai, Y. Lai, N. Davis, F. Nuflo. Modeling Visual-Attention Via Selective Tuning. *AI*, Vol. 78, No. 1-2, pp. 507-545, October, 1995.

[17] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-Time Tracking of the Human Body. In *SPIE*, Vol. 2615, 1995.