Tracking Objects By Color Alone

Christopher Rasmussen, Kentaro Toyama, and Gregory D. Hager

Department of Computer Science
Yale University
51 Prospect Street
New Haven, CT 06520-8285
rasmuss@powered.cs.yale.edu, toyama@cs.yale.edu, hager@cs.yale.edu

Abstract

This paper discusses a simple approach to tracking objects based solely on their color. By limiting our attention to areas of approximately uniform color on nearly Lambertian surfaces, we can assume that pixels corresponding to surface patches on the object form a linear, tubular distribution in color space. Performing principal component analysis on user-selected sample pixels parametrizes an ellipsoidal model of this distribution. The shape thus derived defines a static membership function that can be used to quickly and robustly track many objects through a range of orientations and scales, with varying levels of illumination, and in front of complex backgrounds. In combination with a loss 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 hit; 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 in many practical applications.

In the past, blob tracking was largely implemented on gray-scale images using thresholding or simple functions of gray-scale value [1, 2, 4, 11]. These techniques usually relied on a structured environment (a black backdrop) so that the target (a white ping-pong ball) was 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 they use to determine where to move the tracking window, and all of their results are based only on synthetic, abstract image sequences. Pfinder [16] and Perseus [7] 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. 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.

In this paper, we describe a general-purpose technique for color blob tracking within the XVision real-time vision software package [6]. The primary attributes of our approach are that it does not rely on a static 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. We can easily track multiple objects on a Sun SPARC 20 at 60 Hz (almost twice as fast, actually, but we are limited to the rate at which the digitizer can grab interlaced pictures). 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

Clearly, 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. [8]. 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 finding the principal components [10] of the sample color distribution's covariance matrix. The results give the tracker a bounding ellipsoid for determining pixel membership in the object.

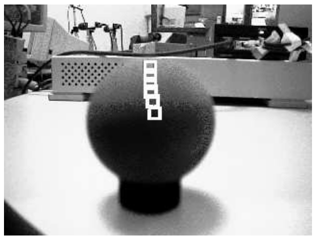
This approach can easily be extended to handle objects composed of multiple uniform color patches. Knowing which pixel belongs to which color, we can partition the object into its constituent colors and fit an ellipsoid to each one. Object membership is the union of the computed ellipsoids.

3 Methods

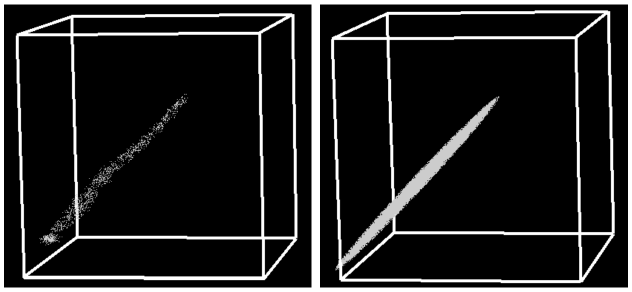
In our formulation, there is one *tracker* associated with each *object* to be tracked. An object is a three-dimensional entity that projects onto a spatially compact set, or region, of similarly colored pixels in the image. A tracker is a process that attempts to position a subwindow in the image at 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 [16]; the tracking window is repositioned to be centered on them.

3.1 Initialization

Initial information about the object 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



(a)



(b) (c)

Figure 1: Constructing $\mathcal{M}$ for a blue racquetball. (a) Six 16 x 16 color samples; (b) the distribution of sample pixels in RGB space; and (c) the ellipsoid fit to them. In (b) and (c), 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.

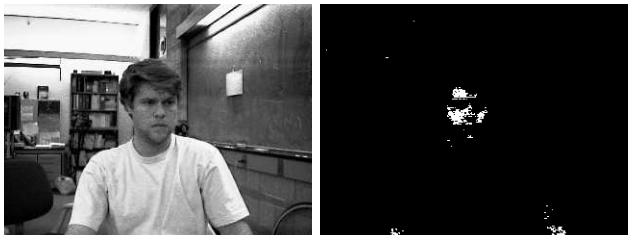
grabs a 16 x 16 pixel subimage centered at the pointer location. The sampled pixels' color values form a distribution in RGB space from which the tracker's color model of the object is derived; the mean image location of the sample clicks is used as the initial tracking window location. The steps of this process are illustrated for a racquetball as object in Figure 1.

The ellipsoid computed for an object can be described mathematically as a scaling S, rotation R, and translation T of the unit sphere. A particular point in color space $P = (r, g, b)^T$ is thus in the object's support if the following membership function $\mathcal{M}$ maps it to 1:

$$\mathcal{M}(P) = \begin{cases} 1 & \text{if } \|S^{-1}R^{-1}(P - T)\| \leq \rho \\ 0 & \text{otherwise} \end{cases}$$

where 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.

Two to four clicks capturing the extremes of a gradient of intensity in the object generally suffice. This gradient can either be spatial, as along a line of longitude on a top-lit sphere; or temporal, as with one spot on a book as it is being rotated. If possible, it is often useful to move the object through a range of locations in the workspace that it might visit later.



(a) (b)

Figure 2: A face as object. (a) Three samples were taken from the subject's forehead and right cheek; (b) white pixels indicate the face's support over the entire image.

One reason to use only a range of intensities is that at the lower, darker end of an object's ideal color ray, the thickness due to noise/color variation causes different color clusters to overlap because they are all emanating from the same point (blackness). In practice, this means that as an object becomes more and more shaded, it is increasingly difficult to discern from any other color, leading to distraction. This effect is especially problematic for similar colors because of the small angle between them. It is up to the user in their selection of sample patches to choose a range appropriate to the color of the object and the colors of the background.

Conversely, extremely bright pixels can be saturated, causing nonlinearity: the portion of an ideal ray that would normally jut out of the color cube (hardware limits colors to a volume of color space) because of brightness is instead projected back onto it. We have found that excluding saturated pixels altogether from shape calculation and membership testing (i.e., they are always out) is satisfactory, and faster than other alternatives.

Figure 2 illustrates the discrimination ability of $\mathcal{M}$ for flesh color.

3.2 Update

We can use $\mathcal{M}$ for tracking by initializing a small (128 x 128 pixels) tracking window centered on the mean image location of the color sample clicks. Each tracking cycle, every pixel in the tracking window is evaluated to compute the object's support. The mean image location of the pixels in the support is used as the center of the tracking window in the next cycle. For reasons detailed in the Applications section below, the size of the tracking window is not varied to account for object scaling.

Since we do not currently update the membership function after initialization, a significant speedup can be gained from precomputing $\mathcal{M}$ over the range of color space once and performing simple table lookups during tracking. For additional speed, we subsample the image within the tracking window at 4 x 4 resolution, yielding a 32 x 32 array of pixels to test for membership.

3.3 Recovery

Occasionally the tracker will lose its object by allowing it outside the tracking window due to excessive speed or occlusion. We define an object as "lost" when the size of its support falls below an aggregate image area threshold α. Distraction, or the mistracking of non-object pixels satisfying $\mathcal{M}$, may also occur, but is not readily detectable with no other discrimination criteria. When lost from object speed or occlusion, though, a tracker can often recover by resampling the entire image at a lower resolution and performing a global search [14]. 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 $\mathcal{M}$ 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 p_i that are in the object's support over the entire image, let $win(i)$ be the set of pixels both in the support and also lying within a hypothetical tracking window centered at p_i; let $win_j(i)$ be the image location of the jth pixel in this set. A heuristic for the best candidate location for the lost object is given by:

$$k = \operatorname*{argmax}_i \sum_j \frac{1}{1 + ||win_j(i) - L||} \qquad (1)$$

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

4 Applications

This kind of tracking is more qualitative than precise. No explicit information about object scale or orientation is generated as a by-product of the update cycle, as is the case with many correlation-based methods [12]. An approximation to such information can in principle be calculated by fitting an ellipse to the object's support, but it is not guaranteed to be meaningful because of the object's three-dimensionality. The size and shape of its support may change as it 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 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. As input to higher level processes, a geometry-less color tracker is probably best suited to being an attentional mechanism [15] that suggests where to perform more sophisticated analyses. There are many task categories for which such a seemingly simple service is quite valuable; we outline some below.

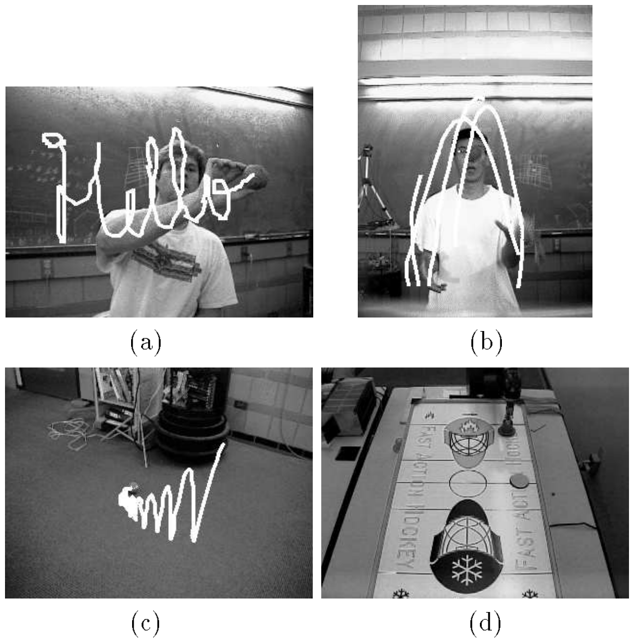


Figure 3: (a) Using the ball as a pen; (b) a red ball being tracked as it and a blue ball are juggled. The blue ball is in front of the juggler's nose and the red ball is falling into his left hand; (c) detection of a ball thrown into view and subsequent tracking as it bounces and rolls; and (d) the air hockey set-up.

4.1 Input Device

By keeping a record of the tracked object's consecutive positions, the tracker gains additional information for use as input to another program. Figure 3(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.

We have also implemented a six degree of freedom (DOF) "spaceball" input device. The set-up is a ball with two nearby, differently-colored dots on it, held close to the camera. By tracking the ball and dots as three separate objects, the ball's three-dimensional position and orientation can be estimated from its support size and shape, tracked location, and the tracked locations of the dots. The difference between the instantaneous measurements of these variables and their values in some canonical ball position provides a 6 DOF velocity controller. To date, the spaceball has been successfully used to position the gripper of a Zebra-ZERO robot arm in real time and to guide walk-throughs and fly-throughs of simulated graphical environments.

4.2 Dextrous Robot Vision

Dynamic, dextrous robots that use visual input typically assume a simplified visual environment in order to achieve robustness and speed. The juggling robot of [11] 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. Our color tracking method 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 3(b) 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 3(c), 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.

The speed and recovery ability of our tracking technique have been put to use by two students in the lab in a system that plays defensive air hockey. The camera's view is shown in Figure 3(d): a green puck is in the middle right of the playing area, and a Zebra-ZERO arm grasping a paddle guards the goal at the far end against a human opponent. Our method tracks the puck, providing information to a Kalman filter from which predictions about the puck's destination are used by the robot to plan an interception point. Despite a colorful background, sharp hits by the human player, occasional occlusions of the puck by his or her arm, and the puck's tendency to slide out of view briefly at the bottom of the playing area, tracking is robust and accurate.

4.3 Face Tracking

Our technique also works well for tracking faces in many indoor lighting situations, both from fixed and mobile camera platforms (Pfinder [16] does not allow camera movement). We have had good results with mounting the camera on a pan-tilt unit, allowing a tracker to follow the subject's face over a wide range of positions, as shown in Figure 4. 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. Note the variation in image size of the subject's head as they move and the changing pose of their face. As long as there are no other human faces that the tracked face goes directly in front of or behind, performance is good. We hope to soon attach the camera to our Nomad 200 mobile robot and investigate following, homing, and avoidance behaviors after an onboard color digitizer is installed.

5 Discussion

The speed gained from subsampling during tracking is beneficial, but object scaling can pose problems. When the image dimensions of an object's support shrink enough to be severely affected by aliasing resulting from subsampling, track is lost before it would be at full resolution. A fovea-like multiple resolution model [13] offers a compromise between efficient computation and accurate object localization. By decreasing sampling resolution away from the center of the tracking window, many fewer pixels can be tested for color membership per update while maintaining tracking accuracy. Foveation also suggests a link between the window-based tracking and full-screen search procedures that we now use, and might obviate the need to switch between them.

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 correspond-

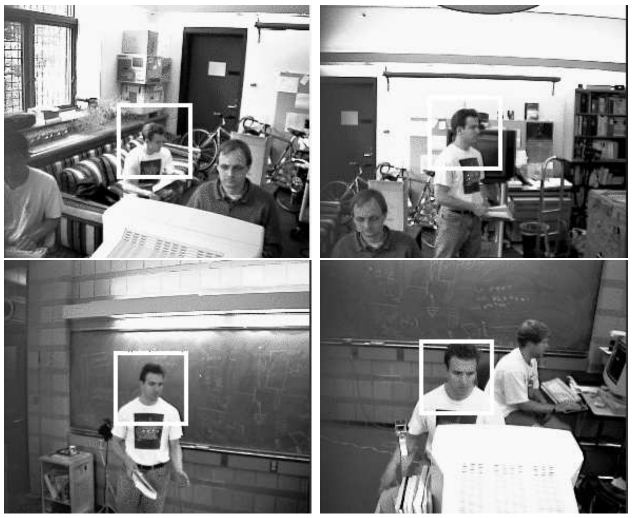


Figure 4: Selected images from a sequence of a person's face being tracked as they walk around a room. Four samples were taken of the subject's face beforehand at spots spanning the light-dark transition in the room.

ing to non-Lambertian objects may change according to the spectral composition of the light incident upon them. This mixture of light coming from radiant surfaces in the environment changes as an object moves about. At any given instant the object's color distribution may still be accurately modeled with an ellipsoid, but over time the original model ceases to be the best fit. It may be possible, however, to adapt the color model, or "track" the ellipsoid in color space, to maintain a best fit at all times [16]. This would be difficult because shadows can cause discontinuities in the incident light mixture.

Finally, it is important to remember that although we are calling the things being tracked "objects," they are really nothing more than patches of color. Differentiating between the flesh tone of someone's face and a distracting tan wall, for example, happens to be equivalent to finding a face object, but this is not always the case. When a person's hand or other people's faces enter the tracking window, multiple patches of what are legitimately the same color contend for the tracker's attention and can cause mistracking. If we wish for the tracker to deal correctly with such a contingency, further information, such as geometry, is necessary. 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 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, 9].

Despite its limitations, the ellipsoidal color tracking scheme that we have presented has wide applicability. It is especially notable for its high speed and insensitivity to gross scaling, rotations, or other geometric distortions of the tracked object. 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 it is suited, the technique demonstrates an ability to perform tracking tasks beyond the reach of most other approaches.

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

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

[8] 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.

[9] 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.

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

[11] 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.

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

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

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

[15] 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.

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