# Object Tracking using Particle Filters

4005-758-01 - Dr. Gaborski

Due on Thursday, May 14, 2009

**Brian Rezenbrink and Jeffrey Robble**

# 1 Introduction

Particle filters are often employed to solve a variety of recursive state estimation problems in non-linear dynamic systems. Specifically, the Kalman filter is used in many control theory and control feedback systems where it is necessary to estimate the state of a system based on its previous state, such as in the areas of mobile robot localization and modeling of gyroscope motion. Such systems receive information about events on a periodic basis. They apply the Kalman filter to predict what happens between these information updates.

The extended Kalman filter (EKF) can approximate non-linear motion by approximating linear motion at each time step. The Condensation filter is a form of the EKF. It is used in the field of computer vision to track objects in video by predicting the position of particles is space. When people refer to a particle filter in computer vision they are most often referring to a version of the Condensation filter. The term particle filter is used in this paper.

The initial focus of our research was implementing a version of the particle filter presented by Cuevas, Zaldivar, and Rojas [2] with some modification and applying it to track students in a classroom video scene. Their filter uses a color histogram model to determine particle similarity. Because their filter is only intended to track a small distribution of pixels frame-by-frame throughout a video, we needed to devise a way to track multiple pixel distributions in order to account for the entire area of a multi-colored object, such as a student. To this end we considered applying spring forces between a particle system tracking the face of a student and a particle system tracking the torso of a student. This approach was met with minimal success so we began to investigate the face-dectection work of Viola and Jones using rectangular windows for feature extraction [3].

One of the major reasons we considered using rectangular windows is because they are able to divide an image into sub-regions and in turn calculate a feature for each sub-region. The benefit of using rectangular windows over a color histogram model is that the relative positions of color distributions is maintained. Thus, the flesh-tone of a student's face may be captured in the upper region of a rectangular window while the color of the student's shirt is captured in the lower regions. On the other hand, the color histogram model will not consider that the flesh-tone component of the color distribution should be located above the component capturing the color of the student's shirt. Thus, a particle system that employs the color histogram model is more likely to jump from tracking a student's face to a student's arm, for example.

In Section 2 we present the theory behind the Kalman filter and the EKF. Then we present the particle filter in Section 3. In Section 4 we present our approach to handling multiple particle systems using spring forces and the issues we encountered. Then in Section 5 we discuss a Viola and Jones inspired particle filter approach using rectangular windows for feature extraction based on the work of Yang, et al. [4]. Finally we consider areas for future research in Section 6 and conclude in Section 7.

# 2 Kalman Filter

The Kalman filter is used to solve recursive state estimation problems in non-linear dynamic systems where it is necessary to estimate the state of a system based on its previous state. Consider the following process equation that describes the behavior of the system in question (note that all equations in this section are taken from [1]):

$$\mathbf{x}_{k+1} = \mathbf{F}_{k+1,k}\mathbf{x}_k + \mathbf{w}_k \tag{1}$$

$\mathbf{F}_{k+1}$ is the transition matrix that describes the transition of state $\mathbf{x}$ at time $k$ to time $k+1$. $\mathbf{w}_k$ is additive, white, Gaussian process noise with zero mean and zero off-diagonal covariances. Note that the variances along the diagonal are represented by the matrix $\mathbf{Q}_k$.

Now consider the following measurement equation used to measure the change in behavior of the system under observation:

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{2}$$

$\mathbf{H}_k$ is the measurement matrix that maps the true space onto the observed space and $\mathbf{y}_k$ is the observed state of the true state $\mathbf{x}$ at time $k$. $\mathbf{v}_k$ is additive, white, Gaussian measurement noise with zero mean and zero off-diagonal covariances. Note that the variances along the diagonal are represented by the matrix $\mathbf{R}_k$.

We can represent a linear discrete-time dynamical system as a flow chart shown in Figure 1.

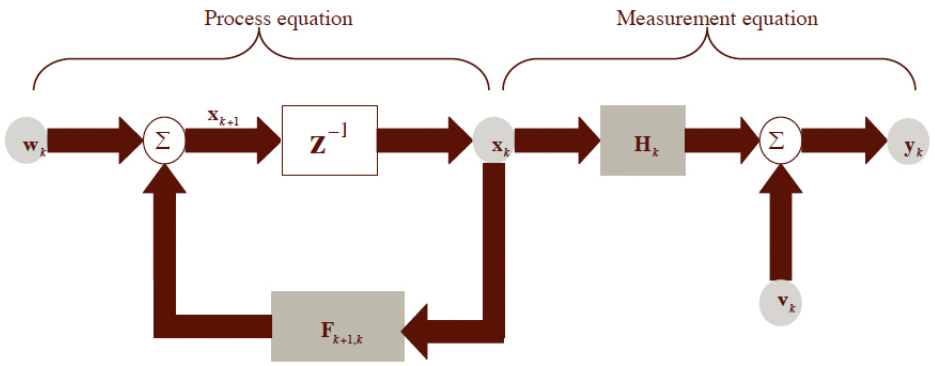


Figure 1: Flow chart showing the various components of a linear discrete-time dynamical system. Image taken from [1].

The Kalman filtering problem is defined as optimally solving the process equation and measurement equation for an unknown state. Using the observed data for all of the previous iterations, $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k$ for $k \geq 1$, we attempt to find the minimum mean-square error estimate of $\mathbf{x}_k$. Note that the state $\mathbf{x}$ at time $k = 0$ is known.

Consider that we are concerned with finding $\mathbf{x}_i$ for the current time $k$. If $i = k$ then we are trying to draw a conclusion about $\mathbf{x}$ based on the current state of the world and all observed data. This is known as a filtering problem, such as object tracking. If $i > k$ then we are trying to predict a future state, so we have a prediction problem. If $1 \leq i \leq k$ then we are trying to determine the state based on some prior and subsequent information, so we have a smoothing problem. Figure 2 summarizes how the Kalman filter can be used for recursive state estimation to solve a filtering problem.

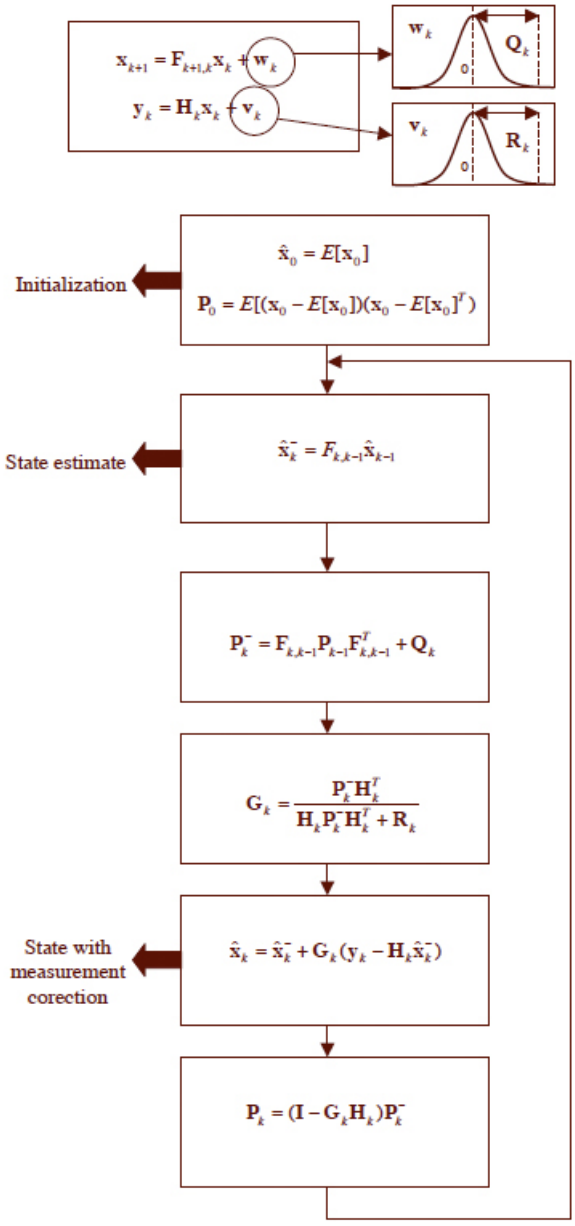


Figure 2: Flow chart showing how the Kalman filter can be used for recursive state estimation. Image taken from [1].

Although the details of the equations are beyond the scope of this paper, essentially recursive state estimation reduces to 6 steps:

1. In the absence of observed data at $k = 0$ the initial state estimate, $\hat{\mathbf{x}}_0$, is simply $E[\mathbf{x}_0]$, where $\mathbf{x}_0$ is the initial state and $E$ is the expectation operator that attempts to minimize mean-squared error. The posterior covariance matrix, $\mathbf{P}_0$, is generated based on the initial state and initial state estimate.

2. The next partial state estimate, $\hat{\mathbf{x}}_k^-$, at time $k$ is determined by simply applying the transition matrix, $\mathbf{F}_{k+1,k}$, to the previous state estimate, $\hat{\mathbf{x}}_{k-1}$.

3. The next partial posterior covariance matrix, $\mathbf{P}_k^-$, is calculated by transforming the previous posterior covariance matrix, $\mathbf{P}_{k-1}$, using the transition matrix and then adding in the Gaussian process noise component, $\mathbf{Q}_k$.

4. The Kalman gain factor, $\mathbf{G}_k$, is calculated by transforming the next posterior covariance matrix by the measurement matrix, $\mathbf{H}_k$, and dividing it by the transform of the next posterior covariance matrix with the measurement matrix summed with the Gaussian measurement noise component, $\mathbf{R}_k$.

5. The Kalman gain factor is then applied to the difference between the observed state, $\mathbf{y}_k$, and the estimated observed state, $(\mathbf{H}_k\mathbf{x}_k^-)$. This offset value is then added to the next state estimate to get the true next state estimate, $\hat{\mathbf{x}}_k$. The offset value is used for measurement correction.

6. The true posterior covariance matrix, $\mathbf{P}_k$, is then calculated based on the Kalman gain factor, measurement matrix, and partial posterior covariance matrix.

7. Steps 2-6 are then repeated to estimate the state for the next iteration.

### 2.1 Extended Kalman Filter

The extended Kalman filter (EKF) can approximate non-linear motion by approximating linear motion at each time step. The state space model is linearized at each time instant around the most recent state estimate. Standard Kalman filter equations are then applied to the linear model and the process shown in Figure 2 is performed [1]. The process equation now becomes:

$$\mathbf{x}_{k+1} = \mathbf{f}(k, \mathbf{x}_k) + \mathbf{w}_k \tag{3}$$

$\mathbf{f}(k, \mathbf{x}_k)$ is a function representing a non-linear transition matrix that may be time-variant. The measurement equation becomes:

$$\mathbf{y}_k = \mathbf{h}(k, \mathbf{x}_k) + \mathbf{v}_k \tag{4}$$

Similarly, $\mathbf{h}(k, \mathbf{x}_k)$ is a function representing a non-linear measurement matrix that may be time-variant. We construct the transition matrix from time $k$ to time $k+1$ as:

$$\mathbf{F}_{k,k+1} = \frac{\partial \mathbf{f}(k, \mathbf{x}_k)}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \tag{5}$$

The matrix is equal to the partial derivative of the transition function with respect to state $\mathbf{x}$ evaluated over the most recent state estimate, $\hat{\mathbf{x}}_k$. We construct the measurement matrix for time $k$ as:

$$\mathbf{H}_k = \frac{\partial \mathbf{h}(k, \mathbf{x}_k)}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k^-} \tag{6}$$

The matrix is equal to the partial derivative of the measurement function with respect to state $\mathbf{x}$ evaluated over the most recent partial state estimate, $\hat{\mathbf{x}}_k^-$. Once $\mathbf{F}_{k,k+1}$ and $\mathbf{H}_k$ are known, we can employ a first-order Taylor series approximation to calculate $\mathbf{x}_{k+1}$ and $\mathbf{y}_k$ (see [1] for details). Step 2 of the recursive state estimation process shown in Figure 2 is not update to account for Equations 5 and 6. The updated process is shown in Figure 3.

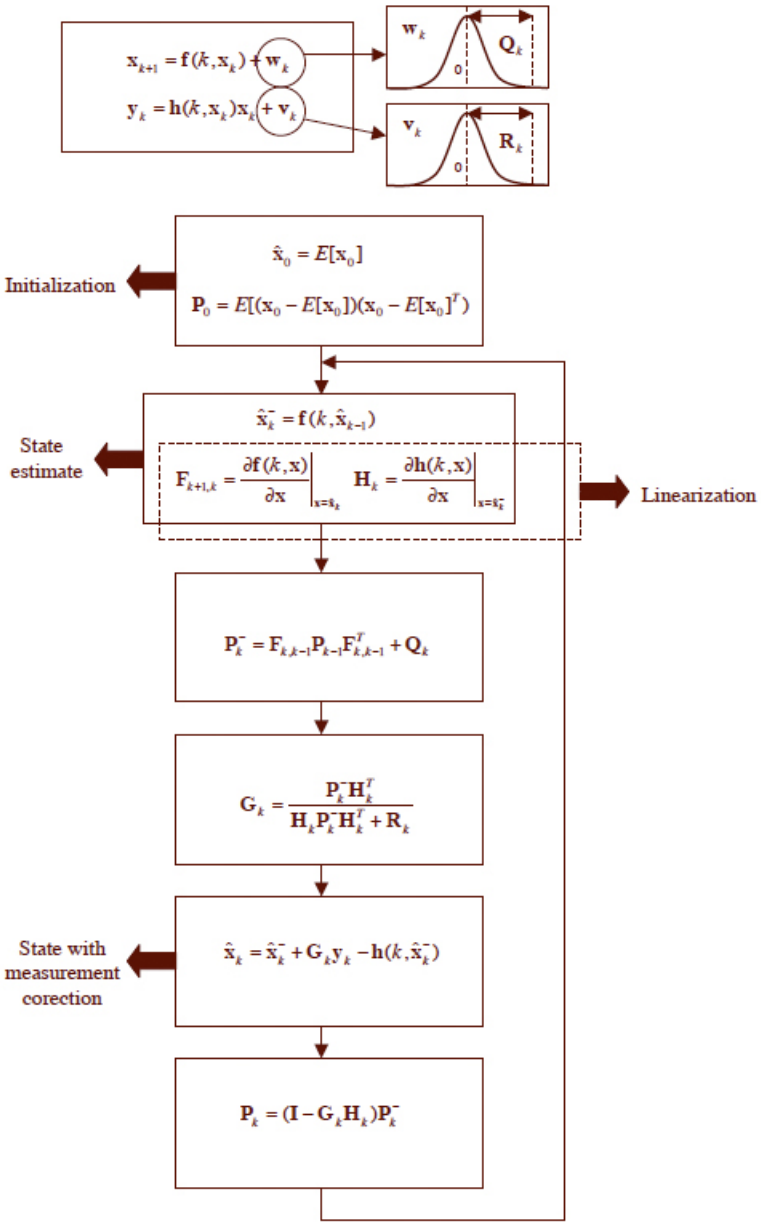


Figure 3: Flow chart showing how the extended Kalman filter can be used for recursive state estimation. Image taken from [1].

# 3 Particle Filter

The extended Kalman filter inspired the development of the particle filter for tracking objects in video. The particle filter presented by Cuevas, Zaldivar, and Rojas attempts to track a small color distribution within a circular window centered around a target pixel [2]. All of the equations in this section are taken from their paper.

## 3.1 Particle Selection and Target Similarity

Since we are most concerned with particles concentrated near the center of the window region, we assign a lower priority to pixels further away from the center. To do this we employ following equation where $e$ is the radial distance between a current pixel under consideration and the window center:

$$k(e) = \begin{cases} 1 - e^2 & e < 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

We represent the color distribution within the circular region as $p_y = \{p_y^{(u)}\}_{u=1,2,3...m}$, where $u$ is a histogram bin assignment. Note that each bin corresponds to a combination of color components, for example 3 components in RGB color space. We calculate the value for each color distribution bin, $u$, within a circular region, $I$, centered at location $y$ as:

$$p_y^{(u)} = f \cdot \sum_{i=1}^{I} k\left(\frac{|\mathbf{y} - \mathbf{x}^i|}{r}\right) \delta[h(\mathbf{x^i}) - u] \tag{8}$$

$r$ is the radius of the circular region, $\mathbf{x}^i$ is a pixel within $I$, and $h$ is a function that assigns pixel $\mathbf{x}^i$ histogram bins according to the current target color model. We call this assignment the color profile for pixel $\mathbf{x}^i$. $\delta[\,\cdot\,]$ is the Kronecker delta function. It returns 1 when the color profile of pixel $\mathbf{x}^i$ matches that of $u$ and 0 otherwise.

The normalization factor $f$ is calculated as:

$$f = \frac{1}{\sum_{i=1}^{I} k\left(\frac{|\mathbf{y} - \mathbf{x}^i|}{r}\right)} \tag{9}$$

To measure, $\rho$, the similarity between two color distributions, $p$ and $q$, we use the Bhattacharyya coefficient:

$$\rho[p, q] = \sum_{u=1}^{m} \sqrt{(p^{(u)} q^{(u)})} \tag{10}$$

The larger $\rho$ is, the more similar two distributions are. Given a set of particles (i.e. pixels in the image), $S_k$, at time $k$, we calculate the probability, $b$, of each particle, $S_k^i$, being the target particle, $q$:

$$b^i = \frac{1}{\sqrt{2\pi\rho}} \exp\left(-\frac{(1-\rho[p_{S_k^i}, q])}{2\sigma}\right) \tag{11}$$

$\sigma$ is a variance determined by the user. The smaller the variance, the more selective the particle filter will be about choosing the particle that most closely matches the target pixel. Setting the variance too low will result in very small probabilities and will make it hard to track the target particle. Setting the variance too high will result in a greater probability of choosing a non-optimal particle as the target particle match.

The particle filter algorithm in [2] employs these equations and proceeds according to the following steps:

1. For each particle, calculate the cumulative probability as:

$$c_{k-1}^0 = 0 \ , \ c_{k-1}^i = c_{k-1}^{i-1} + b_{k-1}^i \ \Big| \ i = 1, 2, \ldots, M \tag{12}$$

   Note that the ordering of the $M$ particles is not important. Each particle $i$ in set $S$ now has three components: a state, $\mathbf{x}$, (i.e. location), probability of matching the target particle, $b$, and a cumulative probability, $c$. Thus, $S_{k-1}^i = \{\mathbf{x}_{k-1}^i, b_{k-1}^i, c_{k-1}^i\} \ \Big| \ \text{i} = 1, 2, \ldots, M$.

2. Resample particles using a Monte Carlo approach by selecting $M$ states with repetition from $S_{k-1}^i$ according to the following procedure:

   (a) Generate a uniformly distributed random number $r \in [0, 1]$.

   (b) Find the smallest $j$ for which $c_{k-1}^j \geq r$.

   (c) Select the particle with that cumulative probability.

3. Apply the transition function to the newly selected particle states such that $\mathbf{x}_k^i = \mathbf{f}(k, \mathbf{x}_{k-1}^i) + \mathbf{w}_k$. The transition function can take the form of an update rule, as discussed in Section 3.2.

4. For each new particle calculate the corresponding probability, $b$, of matching the target particle.

5. Normalize the calculated probabilities such that $\sum_{i=1}^M b_k^i = 1$ and build the new particle set $S_k^i = \{\mathbf{x}_k^i, b_k^i\} \ \Big| \ \text{i} = 1, 2, \ldots, M$.

6. Estimate the new target particle location at time $k$ and update the target particle, as explained in Section 3.3.

7. Repeat the previous steps for the next iteration.

### 3.2 Particle Update Rule

Each particle, $\mathbf{x}_k^i$, in set $S_k$ updates its location in the image at time $k$ according to an update rule, $\mathbf{f}(k, \mathbf{x}_k^i)$. In the simplest case, the update rule is static and can be used to account for straight-line linear particle motion:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \Delta x_{k+1} \\ \Delta y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \Delta x_k \\ y_k + \Delta y_k \\ x_k \\ y_k \end{bmatrix} + \mathbf{w}_k \tag{13}$$

We chose to implement our particle system using the Brownian motion update rule:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \Delta x_{k+1} \\ \Delta y_{k+1} \end{bmatrix} = \begin{bmatrix} \exp(-\frac{1}{4}(x_k + 1.5\Delta x_k)) \\ \exp(-\frac{1}{4}(y_k + 1.5\Delta y_k) \\ \exp(-\frac{1}{4}(\Delta x_k) \\ \exp(-\frac{1}{4}(\Delta y_k) \end{bmatrix} + \mathbf{w}_k \tag{14}$$

In practice, Brownian motion is able to track objects with unpredictable movement as long as those objects are not moving too quickly. Brownian motion is based on the work of Scottish botanist Robert Brown in 1827 and attempts to simulate the seemingly random movement of visible particles suspended in a liquid or gas caused by collisions with invisible atoms. It is also referred to as a "random walk" and is commonly used in Monte Carlo methods. Figure 4 is a plot showing the Brownian motion update rule applied to a single particle over time.

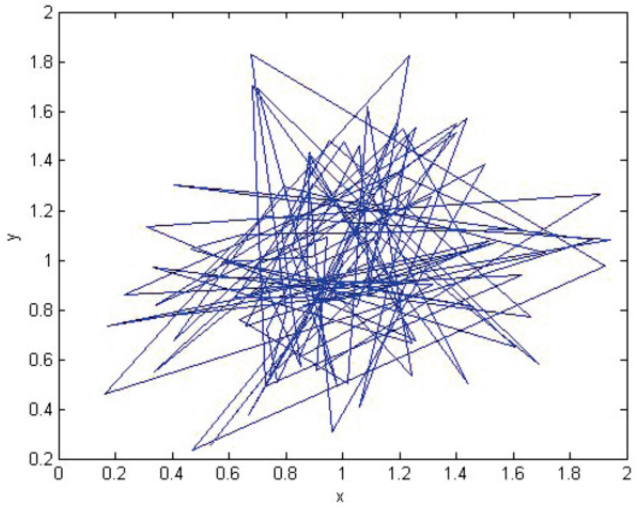


Figure 4: Example plot of the Brownian motion update rule applied to a single particle over time. Vertices in the plot denote positions of the particle at specific instances in time.

### 3.3 Target Particle Estimation

Once all of the particles in the set $S_k$ have been selected and updated, we must determine the new target particle location and update the target particle according to one of three methods: weighted mean, best particle, or robust mean.

The weighted mean method averages all of the $M$ particle locations in set $S_k$ according to their probability of matching the target, $b$, and takes the mean as the new target location, $\hat{\mathbf{x}}_k$:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{M} b_k^i \mathbf{x}_k^i \tag{15}$$

This is illustrated in Figure 5.

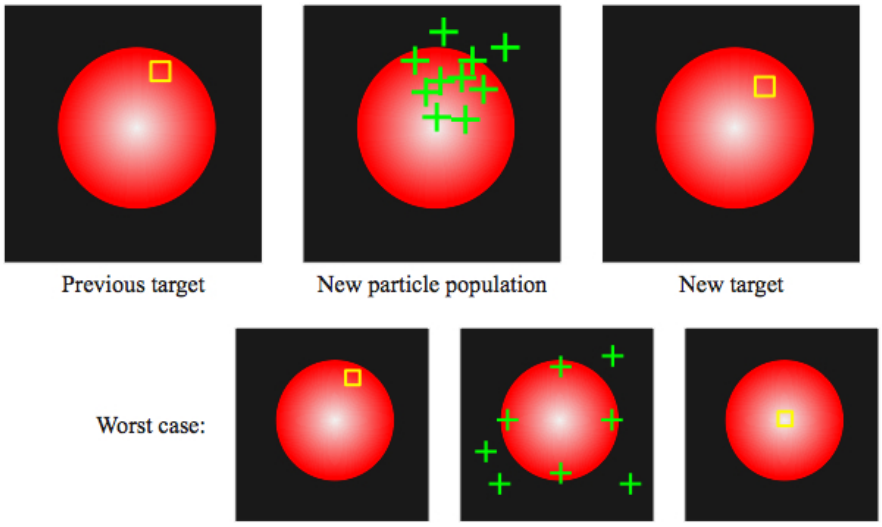


Figure 5: Example of the weight mean target particle estimation method applied to tracking a red orb. The yellow box represents the target particle. The green crosses represent the current particles in the particle set. In the worst case scenario is also shown.

The best particle method simply chooses the particle with the highest probability of matching the target as the new target. This is illustrated in Figure 6.

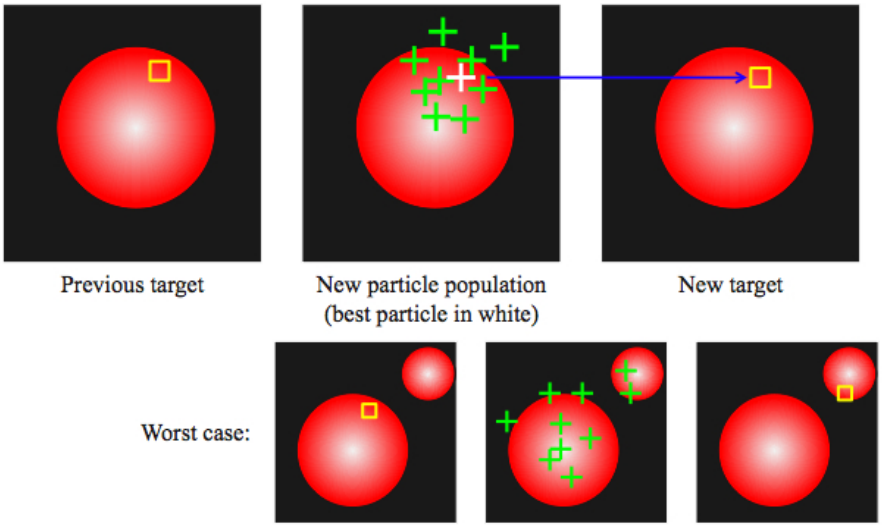


Figure 6: Example of the best particle target particle estimation method applied to tracking a red orb. In the worst case scenario we erroneously start tracking another red orb in the scene.

The robust mean method takes the weighted mean of all particles in the set within a circular region around the best particle. This is illustrated in Figure 7.

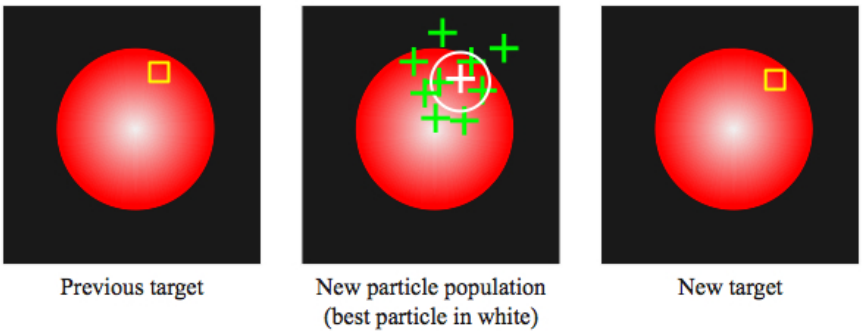


Figure 7: Example of the robust mean target particle estimation method applied to tracking a red orb. The new target was determined by taking the mean of all of the particles within the circular region (shown by the white circle) around the best particle.

In general the robust mean method works better than the weighted mean and best particle methods. The major reason why it is better than the weight mean method is because the resulting target will be located near another particle in the system. This reduces the probability that the target location will be in an undesirable region where the color distribution does not match that of the original target, as shown by the worst case scenario in Figure 5 where the new target is estimated to be at the specular highlight in the center of the red orb. The robust mean method often works better than the best particle method because the probability that the resulting target will jump large distances from its previous location due to the impact of outlying particles is reduced by averaging. The worst case scenario in Figure 6 shows the undesirable effect of outlying particles when using the best particle method. We chose to implement our particle system using the robust

mean method.

### 3.4 Analysis

A single particle system is capable of tracking a single point. A point consists of a target pixel surrounded by a small circular window of pixels. All of the pixels within that window are used to construct a color profile histogram for that point. These points can also be referred to as distributions because they capture the average color intensity within a small area.

We have had decent success tracking these points frame-by-frame in videos consisting of moving colored orbs and a video consisting of a moving remote-controlled helicopter. In general, the objects being tracked in these videos consisted of a single color with slight variations intensity. We also used the same approach to video of a moving soccer ball with great success. This is because both the black and white portions of the soccer ball contribute to the color histogram profile captured by the circular window.

After these simple tests we used the same approach to track faces in the student video. Following some trial-and-error experimentation we determined that the best results were obtained when we selected the target point within the space between a person's eyes. This makes sense because the window around the target point will capture a lot of the variation in a person's face, including eye color, eyebrow color, and skin tone, and incorporate all of it into the target point color histogram profile. If we the made the window around the target point large enough we would also capture part of the person's head hair and incorporate that into the color histogram profile.

This rationale is supported by the face-detection research of Viola and Jones in [3]. Viola and Jones discovered that the majority of faces in images could be detected by placing a rectangular window around a person's eyes and upper cheeks. This rectangle is then horizontally subdivided into two parts to capture the person's eyes and upper cheeks, respectively. The average color intensity of the pixels captures by the upper sub-rectangle would almost always be less than the average color intensity of the pixels in the lower sub-rectangle.

## 4 Multiple Particle Systems

The next goal was to track an entire person. Although we could simply track a person's face using a single representative point between the eyes, we decided to go for a more robust approach that would allow use to track more of a person's body. The most intuitive way to do this is to use multiple particle systems to track multiple points over a person's body. In the simplest case, one particle system would track a person's face and another particle system would be positioned below the first to track a person's torso. The primary issue with this approach is linking these particle systems together. Heuristically, we know that the center of a person's torso is located directly below a person's face by about 1.5 - 2 lengths of that person's head. To take advantage of this relative positional relationship between particle systems, we applied the use of spring forces between target points.

## 5 Hierarchical Particle Filter

As discussed in Section 3, Viola and Jones had great success detecting faces using a rectangular window to capture the average color intensity around a person's eyes and the average color intensity

of that person's upper cheeks. The major difference between Viola and Jones' approach and our single particle system approach is that their target window is capable of determining the relative position of average color intensities. Our original thought was to use multiple particle systems to track targets of varying color intensities and to constrain the relative positions of those systems using spring forces, however, as mentioned in Section **??**, doing so became quite a challenge.

In order to leverage the findings of Viola and Jones, we decided to take our current particle system approach and modify it according to some of the concepts presented by Yang et al. in [4]. Yang et al. present a three stage hierarchical particle filter based in part on the work of Viola and Jones. In general their system tracks objects by performing a coarse to fine-grained search over a portion of an image using rectangular windows. Each stage reduces the search space even further until they are left with the window which has the highest probably of matching the target window. The first stage or their approach applies a two-region rectangular window over a greyscale image of the frame being tracked. If the average color intensities of the pixels in the two regions do not match those of the target within a predetermined threshold, then that window is discarded and not used in the next stage of the search. The primary purpose of this stage is to increase performance by reducing the number of windows and therefore search time of later stages. We do not implement this stage in our approach since the videos we are using are of such a low resolution that performance is not a major concern. The second and third stage presented by Yang et al. are discussed in Section 5.1 and Section 5.3, respectively.

Note that by taking the point corresponding to the upper left corner of a rectangular window oriented along the x,y plane, each window can be thought of as a particle. A particle system then becomes a collection of overlapping windows slightly offset from each other. The same frame-by-frame movement of these particles used by the particle filter discussed in Section 3 can be applied to these particles. The major difference concerns the equation used to calculate the probability of a particle distribution matching the target.

### 5.1 Color Rectangle Features

The window corresponding to each particle in the system is sub-divided in three different ways to calculate color feature. These features are in turn used to determine the particle's similarity to the target particle. The three rectangles shown in Figure 8 are used to sub-divide each window.

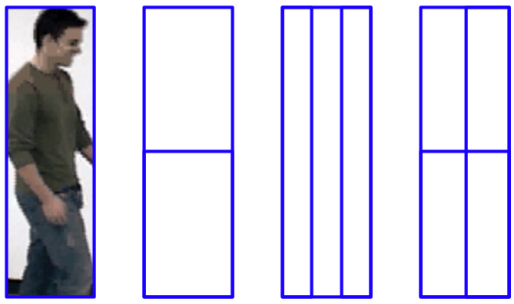

Figure 8: Rectangles used for calculating color features.

The average color intensity for each region of a given rectangle is calculated and constitutes an element of the feature vector for that rectangle. Thus, from left to right, the rectangles in Figure

8 have 2, 3, and 4 elements in their feature vectors, respectively. Note that Viola and Jones used a neural network with about a week of training to determine which rectangles worked best for face detection [3]. We simply choose the same rectangles as Yang et al. in [4] and resize them to capture the entire area of the person we wish to track.

For a given region $R_i$, with area $A_i$, we can calculate the average color intensity.

$$(r_i, g_i, b_i) = \sum_{(x,y)\in R_i} (r(x,y), g(x,y), b(x,y)))/A_i \tag{16}$$

If we let the target color model be $\mathbf{k}^* = \{(r_i^*, g_i*, b_i^*\}$ for all regions $i = 1 \ldots n$ in the rectangular window, then we can calculate the color similarity between the target color model and a candidate color model $\mathbf{k}(\mathbf{x}_t)$ for particle $\mathbf{x}$ at time $t$.

$$\rho(\mathbf{k}^*, \mathbf{k}(\mathbf{x}_t)) = \left[\sum_{j=1}^{n} (r_i^* - r_i)^2 + (g_i^* - g_i)^2 + (b_i^* - b_i)^2\right]^{1/2} \tag{17}$$

Then we can calculate the probability of the candidate matching the target using the following proportionality.

$$p \;\alpha\; \exp(-\rho^2(\mathbf{k}^*, \mathbf{k}(\mathbf{x}_t))/\sigma^2) \tag{18}$$

### 5.2 Integral Image

Depending on the number and distribution of particles in the system, the same image pixels intensities may be summed multiple times. In order to increase performance, each we convert each frame of the video into an "integral image", as discussed in [3]. An integral image is the of the same dimensions as the original image, and maintains the same number of color components, but the value of each pixel takes on the value of the summation of all of the pixel intensities to the top and left of it. This is shown in Figure 9.

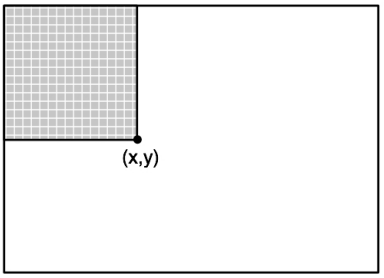


Figure 9: Integral image representation. Pixel (x,y) is equal to the summation of the pixel values in the grey region. Based on a figure from [3].

The integral image for each frame is calculated once and then used to calculate the color features for each of the three rectangles for each particle in the system. Using the integral image increases performance because it drastically reduces the total number of particle summations that need to be performed. One can use the integral image to determine the summation of pixel intensities for a

given region by adding and subtracting the values of other regions. An example is shown in Figure 10.

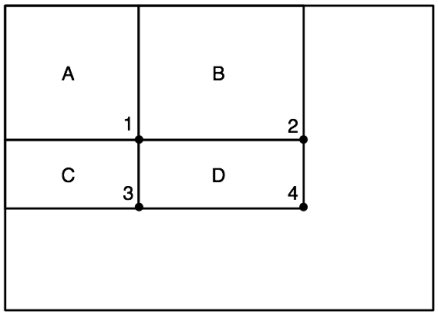


Figure 10: Example integral image use. Pixel 1 is equal to the summation of all of the pixels in region $A$. Pixel 2 is equal to the summation of all of the pixels in region $A + B$. Pixel 3 is equal to the summation of all of the pixels in region $A + C$. Pixel 4 is equal to the summation of all of the pixels in region $A + B + C + D$. The summation of pixel intensities for region $D$ can be calculated as $4 + 1 - (2 + 3)$ Based on a figure from [3].

### 5.3 Edge Orientation Histogram Features

In order to address situations where color information alone is not enough to properly track an object through a scene, we consider edge information. For example, if a person wearing similar colored clothing walks in front of the person being tracked then the current approach may begin to track the new person. To prevent this phenomenon we capture the edge information of the person being tracked. If the person being tracked is standing still, then the edge information for that person will be more vertically oriented than the edge information for the person moving in front of the person being tracked. Comparing edge information is also useful if the color of the person being tracked matches colors in the background of the scene.

Yang et al. [4] perform edge detection by converting the color image to a greyscale image and then applying the horizontal and vertical Sobel kernels, $K_x$ and $K_y$, respectively.

$$K_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \tag{19}$$

$$K_y = K_x' \tag{20}$$

The horizontal and vertical edge images, $G_x$ and $G_y$, are then computed by convolving ($*$) the Sobel kernels with the greyscale image $I$.

$$G_x(x, y) = K_x * I(x, y) \tag{21}$$

$$G_y(x, y) = K_y * I(x, y) \tag{22}$$

The strength, $S$, and orientation, $\theta$, of the edges can then be calculated.

$$S(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)} \tag{23}$$

$$\theta = \arctan\left(G_y(x,y)/G_x(x,y)\right) \tag{24}$$

After removing any values such that $S > 100$ to remove noise, we calculate the normalized horizontal and vertical edge strengths, $g_x$ and $g_y$, for each pixel in the image and use them to partition the edges into 10 bins.

$$g_x(x,y) = G_x(x,y)/S(x,y) \tag{25}$$

$$g_y(x,y) = G_y(x,y)/S(x,y) \tag{26}$$

We calculate the $g_x$ and $g_y$ image once per frame and separate each into 10 channels where each channel can be thought of as a binary image. For example, if a pixel in $g_x$ channel 5 is assigned a value of 1 it means that pixel was placed in the fifth bin after calculating the histogram for the normalized edge strengths. Similarly to how we calculated average color intensity values for regions within a rectangular window, we can calculate the average number of pixels within a region that have been placed within a certain normalized edge strength histogram bin. To do this we can calculate the integral image for each $g_x$ and $g_y$ channel and then use the arithmetic technique illustrated in Figure 10.

What we end up with is a 10 element feature vector for horizontally oriented edges and a 10 element feature vector for vertically oriented edges. We can then calculate the similarity between a candidate window and the target window using an equation similar to Equation 17 and the calculate the probability of the two being a match using an equation similar to Equation 18. Yang et al. state that this technique is very similar to the Scale Invariant Feature Transform (SIFT) descriptor.

### 5.4 Target Update

The first stage of our approach calculates the probability of each candidate particle matching the target using the color rectangle features, $p_c$, as described in Section 5.1. The particles within the top 5% $p_c$ are then passed onto the second stage which calculates the probability of each candidate particle matching the target using the edge orientation histogram features, $p_e$, as described in Section 5.3. The particles within the top 10% $p_e$ are considered and the rest are ignored. We multiply the probabilities calculated in these two stages together for each of these particles and choose the particle with the highest resulting combined probability, $p_m = p_c * p_e$, as the new target.

In order to handle slight variations in color (such as changes in lighting conditions) and edge orientation over time, we must update the current target color and edge orientation models. The most naive way would be to simply adopt the newly determined target's color model, $\rho_{c_n}$, and edge histogram model, $\rho_{e_n}$. This is undesirable because when the person being tracked becomes occluded the current particle may erroneously jump to an undesirable position near the person and capture the background. From that point forward the particles will then attempt to match the background and not the person. To address this issue we maintain the previous color model, $\rho_{c_t}$, and edge histogram model, $\rho_{e_t}$, and use them to calculate the color model, $\rho_{c_{t+1}}$, and edge histogram model, $\rho_{e_{t+1}}$, that will be matched in the next frame.

$$\rho_{c_{t+1}} = (p_m * \rho_{c_n}) + ((1 - p_m) * \rho_{c_t}) \tag{27}$$

$$\rho_{e_{t+1}} = (p_m * \rho_{e_n}) + ((1 - p_m) * \rho_{e_t}) \tag{28}$$

Equation 27 simply calculates the next frame's color model by multiplying the newly determined target's color model times the probability that the new target matches the old target. This value is then added to the product of multiplying the old target's color model times the probability that the new target does not match the old target. Equation 28 does the same for the edge histogram model. In practice this update scheme works well because when the person being tracked becomes occluded $p_m$ is very low, thus almost no alteration is made to the color model or edge model and the particle position does not change. Hence, the tracking window maintains its position over the person being tracked during the occlusion.

# 6 Future Considerations

Currently the size of the tracking window is constant over the period of time in which a person is being tracked. It would desirable for the window to expand and shrink as the person being tracked moves towards and away from the camera and takes up more and less of the frame area, respectively. We attempted to randomly alter the window width and height for each particle in the particle set slightly to address this. We then observed that for a person walking horizontally across the screen the window would often expand to capture the person's previous position as well as old positions, often increasing in two to three times its original width. Clearly this is undesirable, so the modification was removed from our implementation.

# 7 Conclusion

# References


[1] E. Cuevas, D. Zaldivar, and R. Rojas. Kalman Filter for Vision Tracking. Technical Report B 05-12, Freie Universitt Berlin, August 2005.

[2] E. Cuevas, D. Zaldivar, and R. Rojas. Particle Filter in Vision Tracking. Technical Report B 05-13, Freie Universitt Berlin, August 2005.

[3] P. Viola and M. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[4] C. Yang, R. Duraiswami, and L. Davis. Fast Multiple Object Tracking via a Hierarchical Particle Filter. In *Tenth IEEE International Conference on Computer Vision*, volume 1, pages 212–219, October 2005.