Object Tracking using Particle Filters

4005-758-01 - Dr. Gaborski

Due on Friday, May 15, 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, especially in the areas of mobile robot localization and modeling of gyroscope motion.

2 Kalman Filter

2.1 Extended Kalman Filter

3 Particle Filter

The Extended Kalman Filter inspired ...

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

The color distribution $p_y = \{p_y^{(u)}\}_{u=1,2,3...m}$ at location y is calculated as

$$p_y^{(u)} = f \cdot \sum_{i=1}^{I} k\left(\frac{|\mathbf{y} - \mathbf{x}^i|}{r}\right) \delta[h(\mathbf{x^i}) - u] \tag{2}$$

Where the normalization factor f is

$$f = \frac{1}{\sum_{i=1}^{I} k\left(\frac{|\mathbf{y}-\mathbf{x}^i|}{r}\right)} \tag{3}$$

To measure the similarity between two color distributions we use the Bhattacharyya coefficient

$$\rho[p, q] = \sum_{u=1}^{m} \sqrt{(p^{(u)} q^{(u)})} \tag{4}$$

The larger ρ is, the more similar two distributions are. Given a particle distribution S_k^i, we calculate the probability, b, of it being the target model, q

$$b^i = \frac{1}{\sqrt{2\pi\rho}} \exp\left(-\frac{(1-\rho[p_{S_k^i}, q])}{2\sigma}\right) \tag{5}$$

3.1 Target Update

Use presentation images and talk about three target determination methods here.

Copy + paste and describe the particle system algorithm on page 6 here.

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 [1]. 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.

3.2 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.

4 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 3.2, 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 [2]. 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 4.1 and Section 4.3, 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.

4.1 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 1 are used to sub-divide each window.

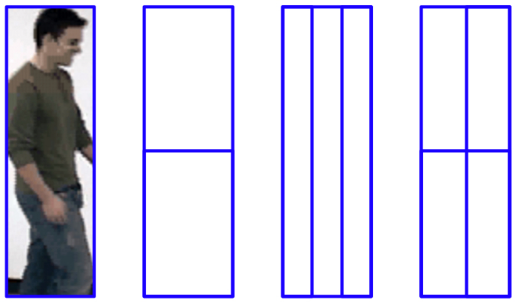

Figure 1: Rectangles used for calculating color features.

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 1 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 [1]. We simply choose the same rectangles as Yang et al. in [2] 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.

$$(r_i, g_i, b_i) = \sum_{(x,y)\in R_i} (r(x,y), g(x,y), b(x,y)))/A_i \tag{6}$$

If we let the target color model be $\mathbf{k}^* = \{(r_i^*, g_i*, b_i^*\}$ for all regions $i = 1 \dots n$ 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.

$$\rho(\mathbf{k}^*, \mathbf{k}(\mathbf{x}_t)) = \left[\sum_{j=1}^{n} (r_i^* - r_i)^2 + (g_i^* - g_i)^2 + (b_i^* - b_i)^2\right]^{1/2} \tag{7}$$

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

$$p \; \alpha \; \exp(-\rho^2(\mathbf{k}^*, \mathbf{k}(\mathbf{x}_t))/\sigma^2) \tag{8}$$

4.2 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 [1]. 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 2.

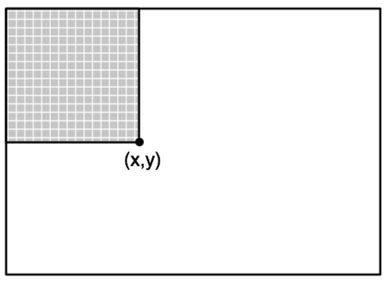

Figure 2: Integral image representation. Pixel (x,y) is equal to the summation of the pixel values in the grey region. Based on a figure from [1].

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 3.

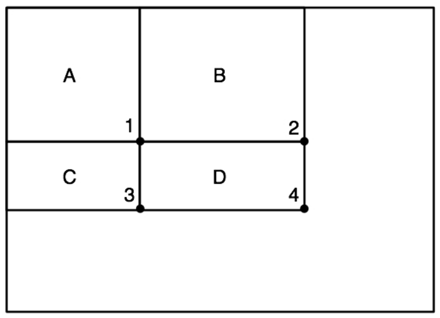

Figure 3: Example integral image use. Pixel 1 is equal to the summation of all of the pixels in region A. Pixel 2 is equal to the summation of all of the pixels in region $A + B$. Pixel 3 is equal to the summation of all of the pixels in region $A + C$. Pixel 4 is equal to the summation of all of the 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 [1].

4.3 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. [2] 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.

$$K_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \tag{9}$$

$$K_y = K_x' \tag{10}$$

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

$$G_x(x, y) = K_x * I(x, y) \tag{11}$$

$$G_y(x, y) = K_y * I(x, y) \tag{12}$$

The strength, S, and orientation, θ, of the edges can then be calculated.

$$S(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)} \tag{13}$$

$$\theta = \arctan\left(G_y(x, y)/G_x(x, y)\right) \tag{14}$$

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.

$$g_x(x, y) = G_x(x, y)/S(x, y) \tag{15}$$

$$g_y(x, y) = G_y(x, y)/S(x, y) \tag{16}$$

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 3.

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 7 and the calculate the probability of the two being a match using an equation similar to Equation 8. Yang et al. state that this technique is very similar to the Scale Invariant Feature Transform (SIFT) descriptor.

4.4 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 4.1. 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 4.3. 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 model. The most naive way would be to simply adopt the newly determined target's color model, ρ_{c_n}, and edge histogram model, ρ_{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, ρ_{c_t}, and edge histogram model, ρ_{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.

$$\rho_{c_{t+1}} = (p_m * \rho_{c_n}) + ((1 - p_m) * \rho_{c_t}) \tag{17}$$

$$\rho_{e_{t+1}} = (p_m * \rho_{e_n}) + ((1 - p_m) * \rho_{e_t}) \tag{18}$$

Equation 17 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 18 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.

4.5 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.

5 Conclusion

References

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

[2] 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.