

Human Head Detection and Tracking

Yisheng Chen

chenyis@cis.ohio-state.edu

1. Introduction

In this report, I present a novel algorithm to detect and track the head regions in a normal human walking sequence. The algorithm also classifies each head orientation for the detected head regions. The main assumption for the algorithm is: Head is always located at an extreme position of the body image and there is a noticeable difference between the head and shoulders in the input images. There are several systems used to detect the head: Ismail Haritaoglu et al. used the geometry features to segment the head region [3], Saad Ahmed Sirohey used an ellipse to represent the head region in an edge map [2], Sangho Park et al. used a partial ellipse to fit the head contour in the silhouette image[1]. Combined with these models, I first apply a bounding box for the head region, and then use an extended ellipse model to match the head region and perform the recursive convex hull algorithm to segment the head region. Kalman Filter is applied on the ellipse representation to track the head. Furthermore, I implement and improve the algorithm of Sangho Park [1] to classify the head orientation.

2. Background Subtraction

The first step to detect and track head is to extract the body from the original footage images. I do the head human head detection and tracking based on the logic silhouette images. So the first step is to do background subtraction. There are several methods to subtract the background: constant threshold RGB model, Gaussian-based RGB model, constant threshold HSV model and Gaussian-based HSV model.

In figure 1, we can see that the background is mainly composed of two parts: the upper region is black wall and the lower region is the light-colored floor. Using 68 frames of background images, I calculate the mean and standard deviation of the individual pixel



Figure 1: The mean of the background images

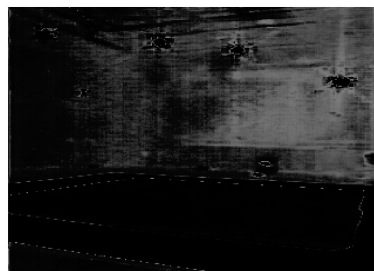


Figure 2: The standard deviation of the background on the hue channel, range in $[0, 1]$.

locations for the background model. The figure 2 shows the standard deviations, which are very noisy in the black region, so it is very hard to get a good foreground. Thus I gave up the HSV model and do the background subtraction on the RGB model. First I calculate the mean value of every pixel in the 68 frames. Then for every pixel in the test images, I check the sum of absolute difference in a 3×3 grids based on the local spatial correlation feature. If the sum is larger than a pre-defined threshold, then the pixel is marked as a foreground pixel, otherwise it is a background pixel. The method does not work very well for every foreground pixel. The first reason is the existence of the shadow area, and the second reason is that I cannot detect the hair region. Actually the hair is too dark and it is indistinguishable from the background. So



Figure 3: One frame of the footage.



Figure 4: A typical silhouette extracted from the footage

finally the foreground that I extracted from the footage is the human body (excluding the hair) and the shadow on the ground. Since this project focuses on the head region detection and track, the shadow on the ground is not a big issue, but due to the lack of hair, the head region in the following parts actually refers to the face region of human body. Figure 4 shows a typical silhouette extracted from the footage.

3. Head Region Detection

As the assumption mentioned before, in the footage, there is only one person and all the motions he makes are some standard walking sequences, which means that his head is always located at the top of his body. Based on the geometry feature of the head region, we know that the shape the head silhouette seems like an ellipse. So if we project the silhouette onto the vertical axis, we can find an increase and decrease when we scan down the project histogram. Figure 5 shows a typical human

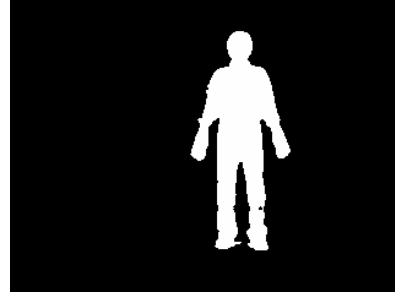


Figure 5: A typical human shape

shape in monochrome image. We will use 3 algorithms to detect the head region.

3.1 Head Bounding Box

Based on the increase and decrease of the histogram of silhouette projection on the vertical axis (y-axis), we can find a coarse bounding box for the head region. After removing the noises of the original silhouette images using the median filter, the first non-zero pixel from the up-to-down scanline is the top pixel of the head, which determines the up bounding of the head region. Continuing the scan on the y-axis projection histogram until we find a local maximum, i.e., $h(y+1) < h(y) \ \& \ h(y-1) < h(y)$, then the row y gives us an indicator of the center of the head region. Then continuing the scan until we find an increase trend, i.e., $h(y+1) > h(y) \ \& \ h(y+1) > h(y)$, and thus we know we are approaching the shoulders and that should be the bottom of the head region. Now we know the top and bottom bounds of the head region and then we should detect the left and right bounds. Along the local maximum axis we have detected, the leftmost non-zero-value pixel is the leftmost



Figure 6: The bounding box of the head region. I draw it a little larger than it should be to make it noticeable.

pixel of the head region, and its x-axis location determines the left bound. Similarly we can find the right bound of the head region.

3.2 Partial Ellipse

As I mentioned, the head shape looks like an ellipse from every view angle. Instead of normal ellipses, which are more applicable to edge maps [2], I use partial ellipses to represent the head region for the logical silhouette images [1]. The standard partial ellipse is

Given the contour points inside the head bounding box, we can get the optimal parameters for the partial ellipse.

$$\square \text{ Equation } \left(\frac{x-x0}{a}\right)^2 + \left(\frac{y-y0}{b}\right)^2 = 1$$

$$\begin{aligned} x &= a \cos \theta \\ y &= b \sin \theta \quad \theta \in \left[-\frac{\pi}{6}, \frac{7\pi}{6}\right] \end{aligned}$$

Since the idea is to fit the best possible partial ellipse to the contour image, the ellipse function need to be solved for all contour points. The parameter a and b can be obtained by the over-constrained system of equations, using the pseudo inverse matrix. The results are accurate as long as the contour points of the head region conform the ellipse functions.

$$\begin{aligned} x0 &= \frac{\sum_i^n xi}{n} \\ y0 &= \max(yi) * \frac{2}{3} + \min(yi) * \frac{1}{3} \end{aligned}$$

$$\begin{pmatrix} (x1-x0)^2 & (y1-y0)^2 \\ \dots & \dots \\ (xn-x0)^2 & (yn-x0)^2 \end{pmatrix} \begin{pmatrix} \frac{1}{a^2} \\ \frac{1}{b^2} \end{pmatrix} = \begin{pmatrix} 1 \\ \dots \\ 1 \end{pmatrix}$$

$$AX = C$$

$$X = (A^t A)^{-1} A^t C$$

Now for every frame in the footage, we know the axis aligned partial ellipse functions for the head region, i.e., we have the center of the ellipse, $x0$ and $y0$, and the major axis length

and minor axis length, a and b . Because the temporal relationship between the frames, we can apply the Kalman Filter to smooth the parameter sets [4]. The parameters obtained by the contour points are the observed value, and we want to obtain the optimal value using estimation and correction. Assume the partial ellipse takes a constant velocity motion, which means:

$$\begin{aligned} x0(t+1) &= x0(t) + vx \\ y0(t+1) &= y0(t) + vy \\ a(t+1) &= a(t) \\ b(t+1) &= b(t) \end{aligned}$$

Then we use two Kalman Filter for the center and axis lengths respectively. Figure 7 shows the parameters before (green) and after (red) the Kalman Filter.

Because the view angle of the head is not always the front view, the assumption about the ellipse model is not always held. We can improve the model by adding more parameters, and one possible better ellipse model is to add rotation to the ellipse coordinates.

$$\begin{cases} x' = x \cos \mathbf{a} + y \sin \mathbf{a} \\ y' = x \sin \mathbf{a} - y \cos \mathbf{a} \end{cases} \quad \begin{cases} x' = a \cos \mathbf{q} + x0 \\ y' = b \sin \mathbf{q} + y0 \end{cases} \quad \mathbf{q} \in \left[-\frac{\mathbf{p}}{6}, \frac{7\mathbf{p}}{6}\right]$$

There are two methods to find the new major axis: brute search and PCA on head contours. The first one is quite straightforward: in the predefined range, rotate the y axis certain angle and use the aforementioned method to find the parameters for the partial ellipse. By using PCA we avoid the exhaustive search and locate the major axis with only one step. However, if we apply the PCA on the entire head contour, the result is undesirable. Actually the top contour of the head tends to drag the major axis to the y axis, as shown in the Figure 9a. If we exclude the top several rows of the head contour and do the PCA on the 2 sides of the head contour respectively and then average the 2 axis, we can get an approximate major axis for the ellipse, as shown in the Figure 9b.

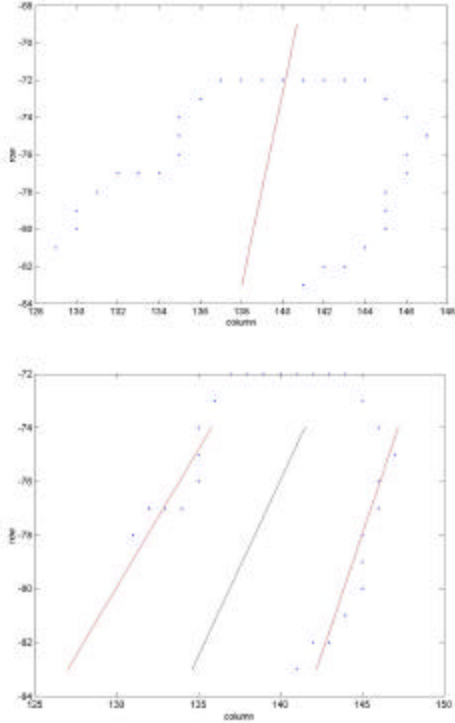


Figure 9a (top): Apply PCA on the entire head contour

Figure 9b (bottom): Apply 2 PCA on each side of the head contour (excluding the top lines)

Moreover, we can use different phases for the ellipse model, i.e., the new ellipse function can be written as:

When the phased is changed, we may

$$\begin{cases} x' = x \cos \mathbf{a} + y \sin \mathbf{a} \\ y' = x \sin \mathbf{a} - y \cos \mathbf{a} \end{cases}$$

$$\begin{cases} x' = a \cos \mathbf{q} + x_0 \\ y' = b \sin \mathbf{q} + y_0 \end{cases} \quad \mathbf{q} \in [\mathbf{b}, \mathbf{b} + \frac{4}{3}\mathbf{p}]$$

obtain a better fitting of the partial ellipse. Figure 10 shows the difference between a standard partial ellipse and a dynamic phase partial ellipse

The metric used to measure the fitness of a method is the sum of absolute difference between head contours and the ideal ellipse functions, and as expected Brute search + Dynamic phase gives the best result, while PCA + Static phase is the worst one. The search time for a frame is also measured, and it shows that Brute search is faster than PCA in static phase. A possible reason is Matlab uses more time to calculate the PCA than do a small range brute search. All the comparisons are shown in Table

1 and Table 2, and the ellipses are drawn in Figure 14.

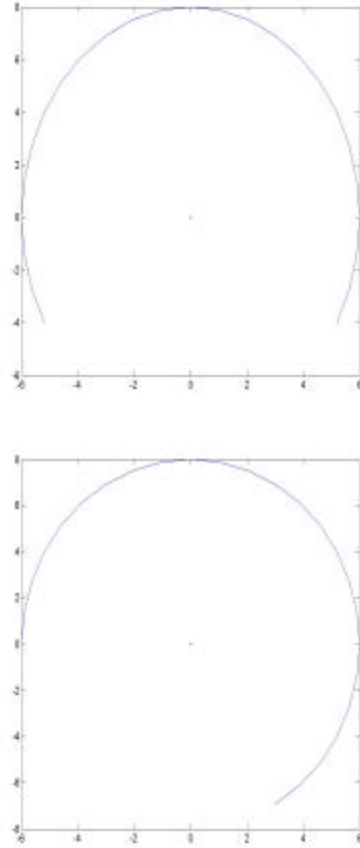


Figure 10: left figure shows a normal partial ellipse model, and right figure shows a partial ellipse with dynamic phase.

	Brute Search	PCA
Static Phase	0.3897	0.4724
Dynamic Phase	0.3847	0.4664

Table 1: The pixel distance SAD of different ellipse models

	Brute Search	PCA
Static Phase	1.5930	1.6250
Dynamic Phase	2.7350	1.7180

Table 2: The time for different search methods Second/frame

3.3 Head Segment

Because the structure and shape of the human body, the head is mostly possibly located at the top of the body, i.e., we can find the head according to the geometry features of the silhouette boundary [3]. We employ the recursive convex hull algorithm to find those extreme points on the contour images. Actually, because my goal is to locate the head region, I only apply the convex hull algorithm inside the bounding box. In the first round of the convex hull algorithm, I get some extreme points including the top of the head, the ears (in the front view) and the neck vertices. In the second round, the algorithm is used to find the concave vertices between two convex hull vertices detected in the first iteration. The two end

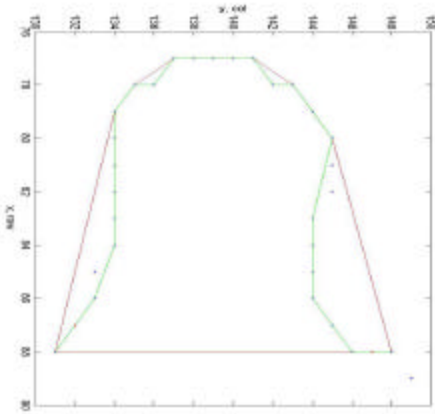


Figure 11: convex (red) and concave (green) hulls of the head region. The extreme head contour points are marked as two red points. All silhouette pixels above the two points are head region pixels.



Figure 12: the segmented head region

vertices of the head contour are the extreme points in the concave point set. Then all silhouette above the line that is decided by the two head corners are marked as the head region pixels. An example is shown in Figure 11 and its result is in Figure 12..

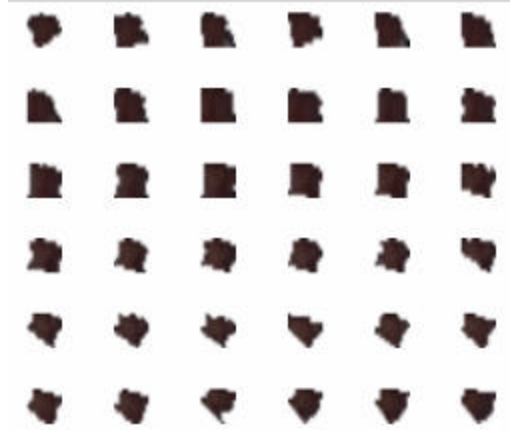


Figure 13: Face template matrix gives the head appearances for a 180-degree turnaround sequence.

4. Head Orientation

Given a sequence of 180-degree head rotation, I extract the colored head regions of these frames and consider them as the head templates for the typical view angles. All of the templates are resized into 16*16 images (Figure 13). Then for the input test sample images, I try to compare the head region of the input image (also be resized to 16*16) with the templates, and get an approximation of the input head orientation. The template matrix is divided into 6 classes according to the row index, and I use two methods to do the classification: the first one is to use the mean of 6 templates in a same class as the feature of this class; the second one is to use the 7 nearest neighbors out of the 36 template samples, and a test sample is classified as class i if the feature has the largest number of nearest neighbors belonging to class i among the 7 nearest neighbors [5]. Two kinds of distance functions are used in the classification. The first one is pixel-wise comparison. For every pixel inside the head bounding box, it is compared with the corresponding pixel in the template, and the difference of the RGB channels is taken as the distance. The second function is a

moments-based feature vector. Use a 4*4 grid mask on every template, and the feature vector X is composed of 16 moment values [1]:

$$X = (W_1, W_2, \dots, W_{16})$$

Where each W_i is computed as follows:

$$W_i = \frac{\frac{1}{N} \sum_{j,k} I_i(j,k) - M}{SD}$$

Where $I_i(j,k)$ is the pixel intensity at location (j,k) in window W_i of the grid and N_i is the number of non-zero pixels in W_i . M and SD are the mean and the standard deviation of the overall non-zero pixels in the template image.

Using the pixel-wise comparison and motion vector respectively, I test the two classification methods on a 13-frame test sequence, and the results are shown in Table 3. Due to the small sizes of the templates, the motion vectors do not work very well; meanwhile, the pixel-wise comparison is quite robust, and the clear profiles of the templates and test samples is a very important reason for it. Some examples of the matching images are shown in Figure 15.

	Pixel-wise Comparison	Motion Feature Vector
6 classes	12/13	11/13
7-nearest neighbors out of 36 templates	13/13	9/13

Table 3: Experimental results (success number / total numbers)

5. Conclusion

Based on the observation of the geometry features of the head region and some existing algorithms, I implement a system that can detect the bounding box of the head region. Moreover, the system can use a good partial ellipse model to match the head contour and then based on the concave corners of the head contour to successfully segment the head region. Kalman Filter is applied on the ellipse representation to track head region in a motion sequence.

The detected head regions can be used to decide the head's orientation. A head image

matrix is constructed as the templates for training the head orientation. Given a frame or a sequence of frames, the system can classify the head orientation into the pre-defined head orientation classes.

6. Future Work

There are several factors that restrict the performance of the current algorithms. First, the head region is quite small now, usually no larger than 20*20, so we cannot get a very good result from the motion features. Also the assumption that head is located at the top of the body decides that the current system can only be run on some simple motions. The unclear background makes the extracted body silhouette do not have a smooth contour. So our future work will try to remove these restrictions. Furthermore, not only for the head region, we also try to detect other body parts, such as the arms, legs and torso. More works will continue on the 3D position retrieval for the head.

References

- [1] Head Segmentation and Head Orientation in 3D Space for Pose Estimation of Multiple People by Sangho Park, J.K. Aggarwal, 2000
- [2] Human Face Segmentation and Identification by Saad Ahmed Sirohey, 1993
- [3] Hydra: Multiple People Detection and Tracking Using Silhouettes by I. Haritaoglu, D. Harwood, L. Davis, 1999
- [4] An Introduction to the Kalman Filter by Greg Welch and Gary Bishop, 2003
- [5] Pattern Recognition and Image Preprocessing by Sing-Tze Bow, 1992

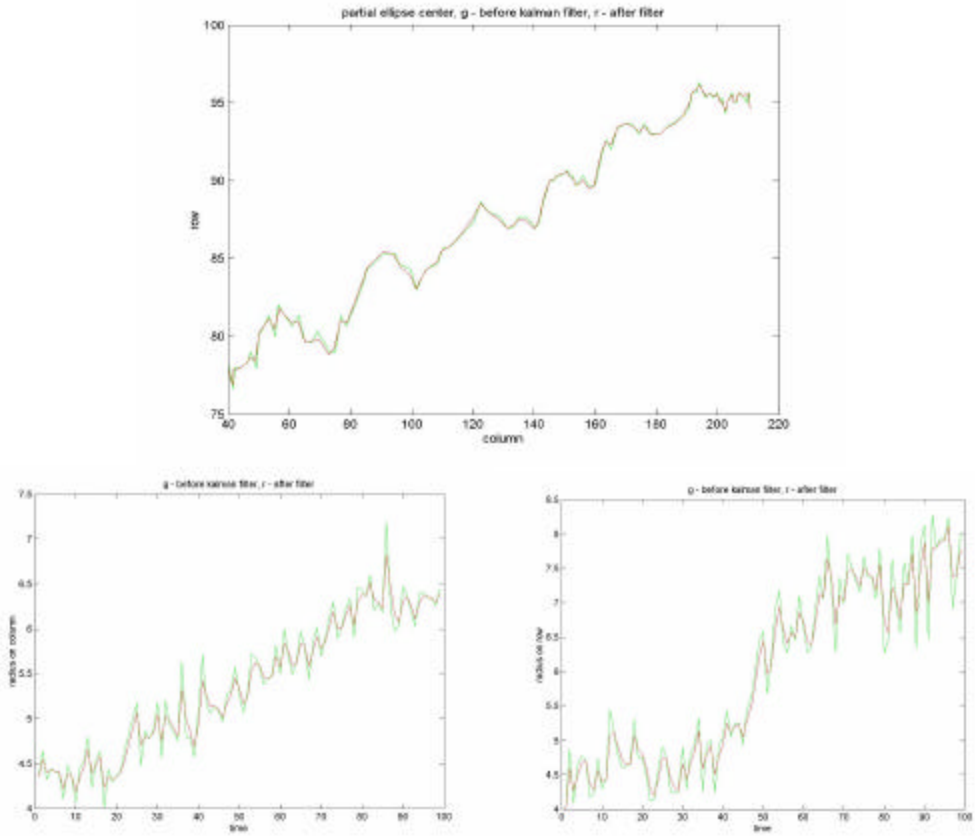


Figure 7: The top one shows the results of Kalman Filter on the center of the partial ellipse. The bottoms two show the results of Kalman Filter on the major axis and minor axis respectively.

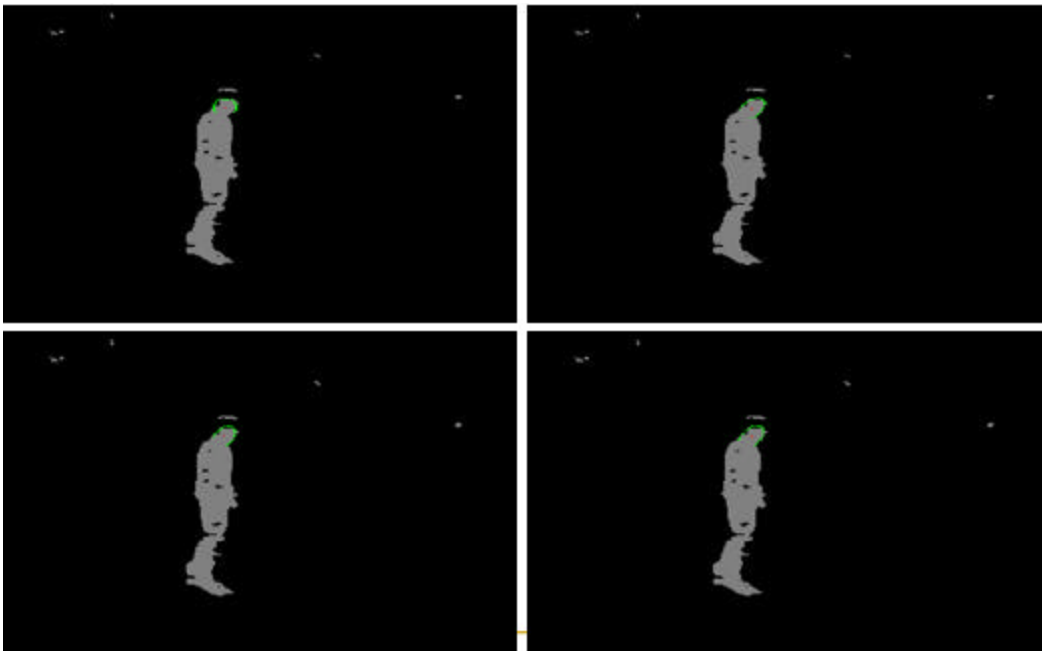


Figure 14: a. left top, using the standard ellipse model
 b. right top, using the PCA to decide the rotation angle
 c. left bottom, using brute search to decide the rotation angle
 d. right bottom, using brute search and dynamic phase to find the ellipse parameters



Figure 15: Each row shows the result of classification. The left images are from the training samples and the right images are from the test data.