Template Based Human Detection and Tracking

Shiwei Song (shiweis@andrew.cmu.edu)
Advised by Chris Geyer (cgeyer@cs.cmu.edu)
Carnegie Mellon University
Pittsburgh, PA 15213

May 1, 2007

Abstract

This paper explores implementation of template based human detection and tracking using techniques presented in [1] and [2]. The general technique involves matching templates of human outlines with distance transformed edge images to locate humans. This study explores several implementation variations with different pre and post processing. Results from these implementations for both moving and stationary scenes are evaluated and compared.

1. Introduction

Human detection and tracking is used in many areas including security monitoring, interactive toys, and pedestrian avoidance. Current work in human detection uses learning-based approaches [1]. Although these techniques generally perform well, they are often complex to implement and requires a large sample of good training data.

This study focuses on a generic shape-based human detection method [1] that searches edge features of an image for human-like shapes. A set of human-shaped templates of different sizes and poses are used. Local minimums in average distances between templates and edges are marked as points of interest [2]. This approach has the advantage of being simple and straightforward to implement and requires no training data. This allows the technique to be used in many different situations and in conjunction with other techniques such as [3].

The outline of this paper is as follows. Section 2 discusses the general technique for this approach. Section 3 goes into implementation details, including optimizations and extensions used to improve performance. Section 4 looks at results of the implementation for data sets from both stationary and moving scenes. Finally, Section 5 concludes the paper by discussing further possible extensions to this approach.

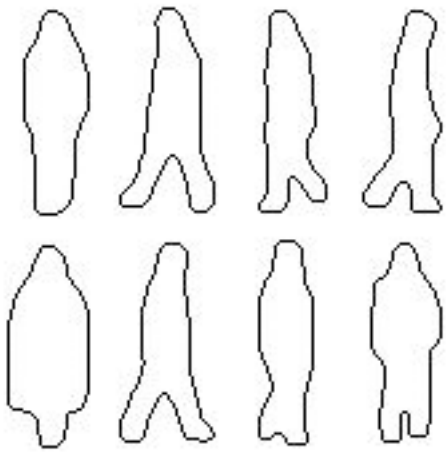

Figure 1: Templates used for detection

2. General Approach

The template based approach uses shape cues for human detection. The goal is to find points in the image with shapes closest to those of the template images. The input image is first passed through an edge detector to extract the shape data. The resulting binary edge image is the feature image of the input image. The non-zero pixels in the feature image are the feature pixels.

Distance transform is applied to the feature image to help search for closest matching points [2]. A distance transform (DT) converts a binary feature image into an image where the value of each pixel denotes the distance from that pixel to the nearest feature pixel. Matching is done by convolving template images with the DTed images to find average distances from template outline to image edges.

A smaller average distance from image edges to template outline means a better match. A detection occurs when the average distance is below a threshold. Convolution is done with templates of different sizes to find the best match (Figure 1). The average distances are scaled appropriately to prevent bias.

3. Implementation Details

age C

3.1 Preprocessing and Edge Detection

Since the algorithm uses edges for human detection, the quality of the edge image directly affects the accuracy of the results. To improve the quality of the edge image, histogram equalization and blurring are first applied to the input image. Histogram equalization increases image contrast and helps accenturate edges; it also make a sequence of input images closer in brightness, thus making thresholding easier. A gaussian mask is used to blur the image. Blurring removes some noise and unwanted texture detail, giving a better edge image.

Once the input image has been preprocessed as above, it is fed to the edge detector. There are a few choices for the edge detection algorithm, with the tradeoff between speed and accuracy. The Canny edge detector is used here for its accuracy (Figure 2).

Figure 2: Result of Sobel (left) and Canny (right)

3.2 Distance Transform and Convolution

Distance transform is applied to the feature image from the previous step and convolution is applied to find closest matching points to the templates. 8 different templates are used to detect humans of various poses (Figure 1). Since humans in images have different sizes depending on how far they are from the camera, 4 different sizes ("notes") of each of the template types are used. Furthermore, the DTed image is scaled down 3 times to get 4 different sized ("octave") DTed image. So for each input image, convolution is applied $8 \times 4 \times 4 = 128$ times to detect shapes of various sizes and poses.

For a particular pixel, p_d, in the DTed image and its nearest feature pixel, p_f

$$p_d = \sqrt{(x_{p_d} - x_{p_f})^2 + (y_{p_d} - y_{p_f})^2}$$

For template image T with dimensions w_t and h_t, the DTed image D, and the resulting convolution image C

$$C(x,y) = \sum_{i=0}^{w_t} \sum_{j=0}^{h_t} \left(D\left(x + i - \frac{w_t}{2}, y + j - \frac{h_t}{2}\right) T(i,j) \right)$$

3.3 Scaling and Result

Convolving a particular template note with the DTed image at a particular octave gives the total distance from the template to features in the image. To find the best matching, we must scale the result to account for octave and note sizes and different template poses. For a particular pixel value $C(x,y)$ after convolution, the scaled value $C'(x,y)$ is

$$C'(x,y) = \frac{C(x,y)}{S^q t_y t_c}$$

Where S is the scale ratio for each octave, q is the current octave (0 to 3, with 0 being largest size), t_y is the height of the template, and t_c is the number of non-zero pixels in the template (i.e. template size). S^q takes care of scaling for different octaves, t_y scales different notes, and t_c scales different template poses.

The work is done after the previous step. Detection is simply iterating through C' and finding the location of the minimum scaled distance.

3.4 Search Window

Conducting the convolutions over the entire DTed image presents two problems. First, convolutions are slow and thus processing each image would take a long time. Second, this may lead to false positives as detections may occur in areas where humans are unlikely to be present, such as in the sky. To address both of these problems, a search windows along the y direction is imposed based on the geometry of the scene. Assuming a flat ground, a standing person further away will be higher in the image. Thus the y-coordinate of the person is linear to the scale of the person [4] (Figure 3). We define the search window to be

$$y = As + B \pm \epsilon$$

where A and B are constants defined by the location of the horizon and the position of the camera.

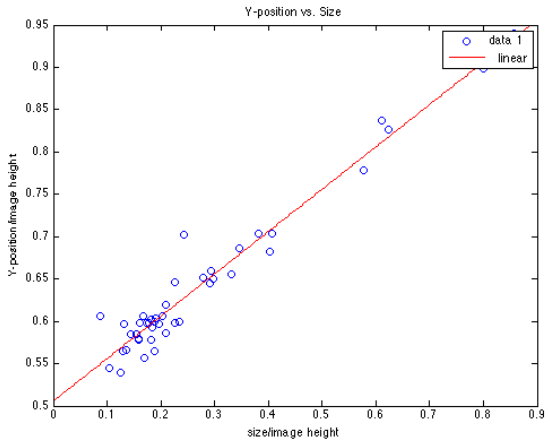

Figure 3: Relationship between size and position

When the program starts, A and B can be found given position of horizon (y_h), head position of any human in the scene (y_t), and foot position of that same person (y_b). If the image is y_i pixels high, then

$$A = \frac{y_h}{y_i}, B = \frac{y_b}{y_t - y_b}$$

Once A and B are derived, the search window can be found by plugging in the height of the template for s. This window decreases the number of pixels we need to convolve over and increases both speed and accuracy. Additionally, since pixel level accuracy is not needed, we can skip every other pixel when convoluving to further increase speed.

3.5 Extensions

For a given input image, the minimum scaled distance may correspond to the location of one of many humans in the image. Sometimes, however, we would like to track a single person's movement over a period of time. Gating is bounding the x direction search window to correspond to movement of a single person. A naive implementation, limiting the x direction search window based on results of previous frame, is used here. A problem with gating is that the gate may stick somewhere due to false positive results, so gating is turned off for the analysis of next section.

Background subtraction takes advantage of the fact that frames from a stationary camera with have a static background. By removing the background from input image before running the algorithm, most of the irrelevant information is removed from the scene, resulting in dramatic improvement in detection accuracy. The background image is generated by assigning the median value for each pixel over several (hundred) frames to be the value of the pixel in the background image. Background is subtracted from an input image by assigning pixels whose values are within a threshold of the corresponding background pixel to be white.

4. Results

4.1 Introduction

Four data sets of 250 images (640 by 480 pixels) each are used to measure the performance of the implementation above. The data sets include: 1. outdoor scene with a moving camera following a person in a park; 2. outdoor scene with a moving camera following a person walking in an urban area; 3. indoor stationary camera in a lobby; 4. outdoor stationary camera at entrance of a building. For each set, detection is done with 2, 4, and all 8 different templates. For the static scenes, detection is also done with and without background subtraction. We measure performance by looking at the algorithm's runtime, median distance from ground truths, and percentage of true positives.

4.2 Set 1 results

The first set of data was taken by a moving camera in a park, following a walking person (Figure 4). The person starts off in an open area and then walks onto a sidewalk.

Figure 4: Scene from data set 1

The table below shows the result of running the first data set. Column one indicates number of template poses used. Column two shows the run time per image processed. Column three gives the median distance from detected human position to ground truth position. Column four is the percentage of true positives.

#	t (seconds)	m (pixels)	%
2	0.41	7	91.63
4	0.47	7	89.64
8	0.68	7	90.04

4.3 Set 2 results

The second set of data was taken by a moving camera in an urban environment (Figure 5).

Figure 5: Scene from data set 2

#	t (seconds)	m (pixels)	%
2	0.40	26	62.34
4	0.46	29	61.09
8	0.66	27	63.18

4.4 Set 3 results

The third set of data was taken by a stationary camera in a lobby (Figure 6).

Figure 6: Scene from data set 3

Table below shows the result of running the third data set. The last column indicates whether background subtraction is used.

#	t (seconds)	m (pixels)	%	subtract
2	0.41	36	60.61	no
4	0.48	34	64.94	no
8	0.68	38	62.34	no
2	0.39	11	96.10	yes
4	0.46	10	96.97	yes
8	0.66	12	96.97	yes

4.5 Set 4 results

The fourth set of data was taken by a stationary camera outside the main entrance of a building (Figure 7).

Figure 7: Scene from data set 4

#	t (seconds)	m (pixels)	%	subtract
2	0.41	24	63.75	no
4	0.48	23	64.14	no
8	0.68	27	58.57	no
2	0.38	6	93.23	yes
4	0.45	6	93.63	yes
8	0.66	6	93.23	yes

4.6 Conclusion

The results for moving camera in open space is very good. This is partly due to the fact that there are no large objects near our search area to create strong vertical edges. The results for sets 2, 3, and 4, on the other hand, suffers from having many edges in our search window obscuring our view. Background subtraction takes care of this problem for stationary camera.

An interesting result is that using more templates does not improve search accuracy as we had hoped. This is because at a distance, a standing human is roughly the same shape. Using more templates, however, may provide hints as to what the detect human is doing or provide cues to movement direction.

Finally, doing background subtraction resulted in faster running time than doing no background subtraction. This unintuitive result may be due to processor optimizations in operating on large number of white pixels in the image.

5. Conclusion

Template based human detection is a simple and versatile approach to human detection. It gives good results for stationary cameras with background subtraction or tracking in large outdoor areas. However, it does not perform well in crowded areas. The simplistic nature of this approach gives room for improvement by extending the technique with other, more sophisticated, techniques. One possible extension is to use a hierarchical template tree to search with many more templates [1]. Another possibility is to use this technique to generate interest points (hypothesis) for techniques such as [3] to verify. Finally, template information may be used to gain information on human behaviors or movement prediction.

6. References

[1] D.M. Gavrila and J. Giebel, "Shape-Based Pedestrian Detection and Tracking"

[2] D.M. Gavrila and V. Philomin, "Real-Time Object Detection Using Distance Transforms"

[3] D. Ramanan, "Using Segmentation to Verify Object Hypotheses."

[4] C. Geyer and B. Grocholsky, "Robust Person Tracking for Following from a Robot"