Tracking of Humans Using Masked Histograms and Mean Shift

Introductory Project in Computer Vision – Summary

Elad Ben-Israel (ben-israel.elad@idc.ac.il) · supervised by Dr. Yael Moses (yael@idc.ac.il)
Efi Arazi School of Computer Science · **The Interdisciplinary Center Herzliya** · March 2007

Abstract

In this project we have experimented with basic computer vision algorithms related to tracking, and used them to implement a simple human tracker based on OpenCV mean-shift tracker. In addition, we have also developed an open-source wrapper library for OpenCV to allow using it from a .NET hosted environment, such as C#.

Motivation

Tracking of humans is an important computer vision building block. It is needed for many applications, ranging from surveillance and military through computer-aided driving and advanced human-machine interfaces. The main challenges in human tracking are: (1) differentiating between background and foreground areas; (2) differentiating between the tracked object and other human objects on the same scene; (3) changes in lighting, which causes the appearance of the tracked object to change; (4) two-dimensional scaling of the tracked object, for scenes where the objects change their distance from the camera; and (5) occlusions of the tracked object by other objects.

The final tracker we devised consisted of a color-histogram mean-shift tracker [1] based on the OpenCV [7] implementation, where the initial histogram accuracy has been improved by using a masked histogram instead of a box histogram. The mask was calculated by interactively allowing the user to place a predefined template on the initial human being tracked, and separating foreground from background using a gray-scale region growing [2] technique on both background and foreground areas.

The tracker has been compared with the standard mean-shift tracker, where the initial histogram is based on calculating a box histogram for a selected region.

Background Subtraction

We have initially tried to use background subtraction combined with the approach suggested by [4] to determine the vector of movement in the scene. The method described in [4] suggests accumulating foreground pixels over time, where the pixel's age is represented by its grayscale value; then, calculating the vector of movement in the scene is done by calculating the gradient on the accumulated history image.

In order to segment foreground and background, we subtracted from every pixel the weighted average of its history, and labeled the pixel according to some threshold: if the subtraction yielded an absolute value higher than the threshold it was labeled foreground. Once foreground and background were segmented, we applied the method from [4] to

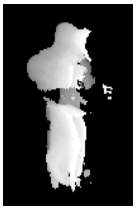

Figure 1: ***Output frame from history algorithm****. As the grayscale gradient suggests, object is moving to the right.*

$$F_{i,j}^{curr} = \begin{cases} label(i,j) = fg \Rightarrow white \\ label(i,j) = bg \Rightarrow F_{i,j}^{prev} - c \end{cases}$$

determine the vector of movement in the scene by preserving the short-term history of foreground pixels in any given frame: every foreground-labeled pixel was colored white (in the single-channel output frame) and for every pixel labeled background, we subtracted its previous grayscale value by a constant *c* as the equation in figure 1 suggests. Thus, every non-zero pixel in each single-channel output frame describes a foreground pixel at some point in "history". The gradient of the grayscale image is then equivalent to the vector of movement in the last *c* frames (see figure 1).

This method can yield an understanding of the vector of movement in the last *c* frames of the scene, but it lacked the ability to differentiate between multiple objects in the scene. As soon as we introduced more than one object (or moving background elements), there was no way of telling which one was the object being tracked. This method can be used as an auxiliary method to some other tracking mechanism (to determine vector of movement and facilitate in path initialization for example), or in applications where a single object is being viewed in the scene (e.g. the application from [4] – a music synthesis program). Therefore, we have decided not to use it for our current implementation.

Overview of Mean-Shift

We are using the scheme implemented in OpenCV [7], which consists of three steps: (1) calculation of an initial histogram which identifies the object being tracked; (2) applying the initial histogram onto every new frame from the input stream using a technique called back-projection, yielding a single channel (grayscale) image where each pixel contains the bin size of the initial histogram for the color of the corresponding pixel in the new frame (this is sometimes referred to as the "probability" of the pixel to be part of the tracked object represented by the initial histogram); and (3) searching the back-projection to find the region with the highest intensity which corresponds to the area where the tracked object most probably resides.

The mean-shift search is performed using a predefined number of iterations, where in every iteration, OpenCV uses spatial moments to calculate a new center of mass on the back-projection.

Masked Histogram Calculation

Along our experiments with mean-shift, it is apparent that one of the main challenges is to calculate a quality initial histogram. If this initial histogram will correspond to colors not related to our object (e.g. background colors or colors from other objects), it will not identify the tracked object accurately, and tracking it will be harder and less efficient.

Figure 2: ***External box histogram****: calculation of initial box histogram (box depicted as a blue rectangle on the top image) and corresponding back projection (bottom image) on the same frame. Shows low accuracy of histogram, as background pixels are contained within the bounding rectangle.*

A common trivial approach to identifying the initial histogram is to place a bounding rectangle on the target object, and calculate the histogram of this entire region. As mentioned before, this potentially leads to inclusion of many pixels not directly related to the tracked object – mainly background. Unfortunately, background pixels are most hazardous to mean-shift, since they increase the probability of

Figure 3: ***Internal box histogram****: calculating the initial histogram based on a region within the object being tracked rather than bounding the entire object yields better back-projects (right image).*

negative correspondence during the tracking phase (figure 2). A possible way to improve the accuracy of such a histogram is to place the bounding box inside the object thus excluding background pixels from the initial histogram.

This yields to a more accurate back-projection (figure 3), but does not allow bounding of the entire human being tracked, possibly minimizing the number of applications such a tracker may be used for. We compare these two "box" histogram approaches (namely internal and external box histogram mean-shift) to our approach below.

The "masked histogram" approach tries to improve the accuracy of the initial histogram by using a simple segmentation technique to try and include only foreground pixels in the initial calculation, and exclude background pixels.

The initial step for devising the masked histogram is to place a human "template" on the target object (see figure 5). The template consists of a grid of foreground and background markers, devised to accommodate a common human form, as we are tracking humans in this application. A possible improvement of this would be to accommodate to different types of human templates (e.g. using some computer learning method). Figure 4 shows the template we used in our tests; however any template may be devised to accommodate different types of objects.

Figure 4: ***Calculation of the histogram mask****. Leftmost image shows placement of template markers on target object. Notice the placement of foreground (blue) and background (red) markers. Middle image shows resulting mask after region growing was applied on foreground and background markers (foreground label is white, background is black and neutral label is gray). Rightmost image shows an overlay of the mask on the original image, where foreground pixels kept their original color and background and neutral pixels are green.*

Then, for each marker, we apply a single-channel region growing [2] algorithm (flood fill), labeling foreground and background regions with two different labels. Foreground regions are calculated first to ensure foreground areas are accommodated and background markers second. The result contains three labels: foreground areas, background areas and neutral areas (areas not included in foreground or background regions).

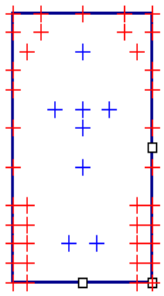

Figure 5: ***Template for human markers****. Blue crosses refer to foreground areas and red to background.*

Following the segmentation stage, we devise the final histogram mask. The mask is a binary matrix of size corresponding to the bounding rectangle size, where each cell indicates whether the corresponding pixel should or shouldn't be part of the histogram calculation. The mask matrix is devised by scanning the segmentation image (with three labels) and for each pixel, marking the corresponding pixel in the mask accordingly. A foreground label denotes inclusion in the mask, a background label denotes exclusion. Neutral labels may then be regarded as foreground or background (see figure 5). In our implementation, the decision of whether to include or exclude neutral pixels in the foreground was left for the user. However, a more automatic approach may be used for

neutral pixels, such as comparison to its surroundings.

Method Parameters

Our method consists of several steps, each using various parameters. We list below the parameters and their effect on our method.

The **number of histogram bins** used for calculating the initial and intermediate histograms: more histogram bins allows differentiating our object from other objects with similar (but not exactly similar) colors. Less histogram bins increase the sensitivity of our tracker to changes in the target's color histogram (e.g. due to lighting changes), but allows it to accommodate to changes in target colors during the scene. A common value of 30 histogram bins empirically yielded satisfactory results.

A second parameter that affects our method is the **flood-fill threshold** used to determine whether a pixel should be included in the region being grown or not. The larger this value is, the less sensitive the flood fill algorithm is, forming larger foreground or background regions. As the majority of humans wear single color garments (shirts and pants are usually single-colored), we used a low threshold on the flood fill (usually 2). We do take into account some typical characteristics of human clothing using our template markers. The shirt area contains more markers, since it is more likely for shirts to have multiple colors, and the pants area contained two markers (see figure 4 above). Thus, the flood fill will "catch" both typical shirts' and pants' colors.

Another possible trade-off that we experimented with, which is also mentioned above, is whether to **include or exclude neutral pixels** in the mask. From our experiments, we have concluded that *exclusion* of those pixels is the correct behavior for the tracking algorithm. In most situations excluding neutral region-growing pixels yielded a good segmentation between the tracked object and the background. In situations where the resulting segmentation *seemed* to be less satisfying (see figure 6, center image), it was actually "good" segmentation in terms of mean-shift performance, as areas of the target object that "merged" with the background were not included at all into the histogram, thus reducing the sensitivity of our tracker to the background.

The forth parameter we experimented with was the **number of mean-shift iterations** to perform during the tracking phase. Increasing the number of iterations allows tracking of faster moving objects (where the center of mass changes more dramatically between frames). As the purpose of our work has been to track humans, a single iteration was sufficient, and yielded quicker results.

Experiments

In order to compare the two methods, we have decided to use OpenCV's mean-shift tracker implementation (*cvMeanShift*). We have also used OpenCV's histogram calculation capabilities.

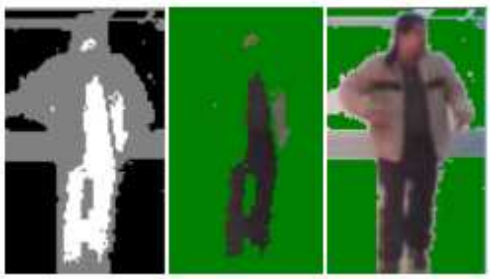

Figure 6: ***Inclusion vs. exclusion of neutral pixels****. Leftmost image shows region growing results (black = bg, white = fg, gray = neutral). Middle image shows mask overlaid on target image while* ***excluding*** *neutral pixels and rightmost image shows the overlay with neutral pixels* ***included****.*

In order to effectively use OpenCV to create a more elaborate user experience (which allows manipulating the various parameters in real-time), we created a wrapping library that allows accessing OpenCV's functionality from a .NET hosted environment (such as C#). This library

has been released as open source [8].

Comparison to Box Histograms Trackers

We compared our method to the box histograms mean-shift method (where the histogram calculated is based on calculating the histogram of a rectangle). We used the box-histogram calculation in two ways: (1) by calculating the histogram for the entire object, using a rectangle that bounded the object from the outside (external box); and (2) by calculating the histogram of a small, single-colored region placed *inside* the object (internal box).

The right-hand side sequence in figure 10 below shows the performance of the **external box** histogram mean-shift tracker. The top images in the figure show the initialization state of both trackers, where the tracking boxes bound the entire tracked object. The following images show a divergence of the external box tracker at a very quick stage of the sequence, as oppose to a successful track of the masked histogram tracker. This is caused due to the fact that the histogram calculated on the external box area contains many background pixels, which divert the tracker during the tracking phase.

The performance comparisons to the **internal box** histogram mean-shift are shown in figure 9. As expected, since the internal box does not contain any background pixels, the initial histogram is more accurate, thus delivering better results of the tracker, rather similar to the results of the masked histogram tracker.

A more insightful comparison can be viewed when observing the back projection images calculated from the three sequences. Figure 7c shows the back projection from the external histogram tracker, which supports our thesis regarding the divergence of this tracker at a very early stage (the mean-shift tries to find the center of mass on the back projection but fails to differentiate between the tracked object and its background). Images (a) and (b) of figure 7 show the results of the masked histogram tracker and of the internal box histogram tracker. Even through the basic tracking performance of those trackers seem similar, the back-projection results show that the masked histogram tracker is more robust than the internal box tracker. The masked histogram tracker uses a more accurate and complete initial histogram of the tracked object, resulting in a brighter back-projection, which may be "easier" for the tracker to follow in more complex situations (the calculation of the center of mass of the object results in a higher area value).

So far we have seen better performance of the masked histogram tracker that resulted in a better initialization phase (more accurate initial histogram of the target object). However, during the tracking phase, the masked histogram tracker regards the entire tracking rectangle when it searches for the center of mass. This exposes the tracker to potentially more interfering objects than that of the internal box tracker (in that sense the masked histogram

(a) *(b)* *(c)*

*Figure 7: **Back projections**. Shows the back projection images resulted during the tracking sequences captured in figure 9 and 10 below. (a) from masked histogram tracker; (b) from internal box histogram tracker; and (c) from the external box histogram tracker.*

tracker is similar to the external box tracker). The internal box tracker suffers less from this problem because the tracking rectangle is smaller, and tends to stay "inside" the object, while the masked histogram tracker bounds a larger region that includes non-target pixels. When the tracking rectangle moves over areas (or objects) with similar histograms, it may be diverted more easily, as the center of mass search may contain those objects (see figure 8). A possible solution to this problem may be to "shrink" the tracking rectangle during the tracking phase, such that it will contain a smaller amount of background noise. This may be done automatically using the segmentation data we obtained during the initialization stage (e.g. form a rectangle devised using a region growing technique from the center of the mask). However, this is not in the context of our current work.

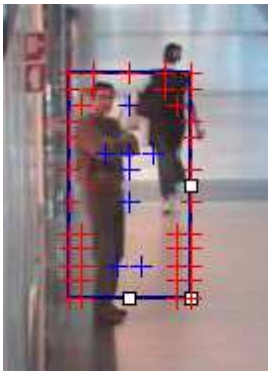

Figure 8: tracking rectangle is larger for our masked histogram tracker, exposing the tracker to other objects, with potentially similar histograms.

Conclusion

The masked histogram mean-shift tracking method we presented in this paper improves upon the initialization stage of the mean-shift method as oppose to using a simple rectangle bounding the tracked object. We used pre-existing knowledge of the human form to create a template of markers used with region growing to perform segmentation of the tracked object. Using the segmentation data, we created a mask to be used when calculating the initial histogram of the object. This histogram is then used to calculate the back-projection for every incoming frame, yielding a more accurate alignment between the back-projection and the actual object being tracked. The more robust the back-projection image is, the "easier" it is for the mean-shift algorithm to locate the center of mass of the actual object.

The external box mean-shift tracker could not track most of the objects we experimented with, as the back projection calculated usually contained high correspondence with background elements. The internal box tracker yielded overall tracking abilities similar to the masked histogram tracker, however it has one major drawback which is that the tracking rectangle does not represent the actual human being tracked, but rather a small box inside the object. This may not be useful in applications where the bounds of the entire human are important, and not just its estimated position based on a small rectangle.

As it was out of scope for this work, no improvements have been made to the tracking stage, in which case the large bounding rectangle used for the masked histogram tracker resulted in a more sensitive tracking stage, as oppose, for example, to those of the internal box tracker.

Advances from this work can be done in two major aspects: (1) use of automatic generation of the human template used for the mask generation, using for example, machine learning methods to generate a more generic template; and (2) improvement of the tracking phase by choosing a smaller tracking region, possibly based on the segmentation data from the initialization phase.

References

[1] D. Comaniciu and V. Ramesh, ***Real Time Tracking of Non-Rigid Objects using Mean Shift****, Computer Vision and Pattern*

Recognition, 2000 Proceedings IEEE Conference

[2] *D. Marshall,* ***Region Growing****, Vision Systems, 1994*

[3] *Forsyth and Ponce,* ***Background Subtraction****, Computer Vision – a Modern Approach, pp. 309-310, 2002*

[4] *G. R. Bradski and J. Davis,* ***Motion Segmentation and Pose Recognition with Motion History Gradients****, Machine Vision and Applications, 2002*

[5] *M. Piccardi,* ***Background Subtraction Techniques: a Review****, Systems, Man and Cybernetics, 2004 IEEE International Conference*

[6] ***CAVIAR Data Set****, http://homepages.inf.ed.ac.uk/rbf/CAVIAR*

[7] ***OpenCV Library****, Intel Corporation*

[8] ***OpenCVDotNet****, http://code.google.com/p/opencvdotnet*

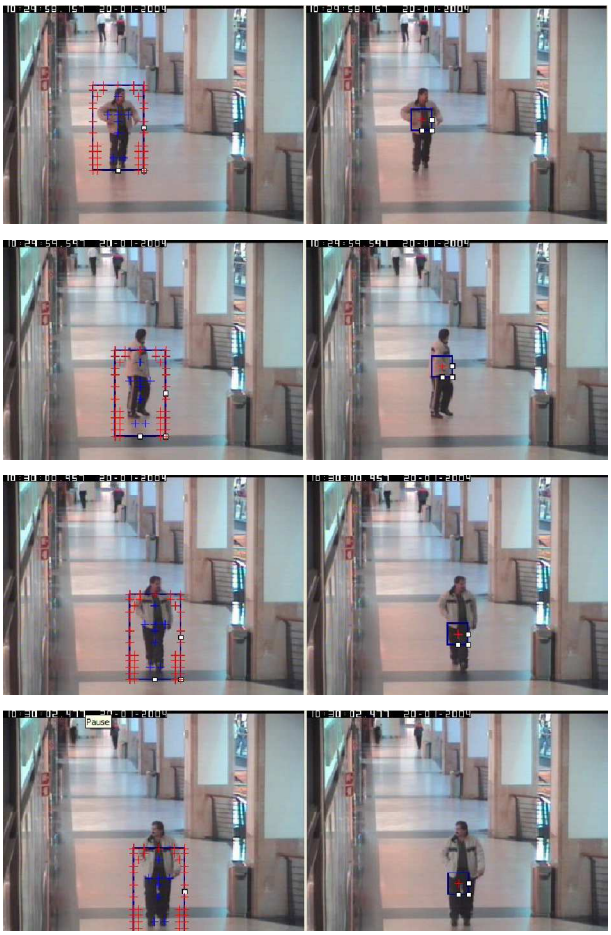

Figure 9: sequence shows behavior of box histogram mean-shift when box is inside the object (left-hand side sequence) against the masked histogram (right-hand side sequence). As box does not contain "junk" pixels (background in this base), performance of the trackers are generally similar.

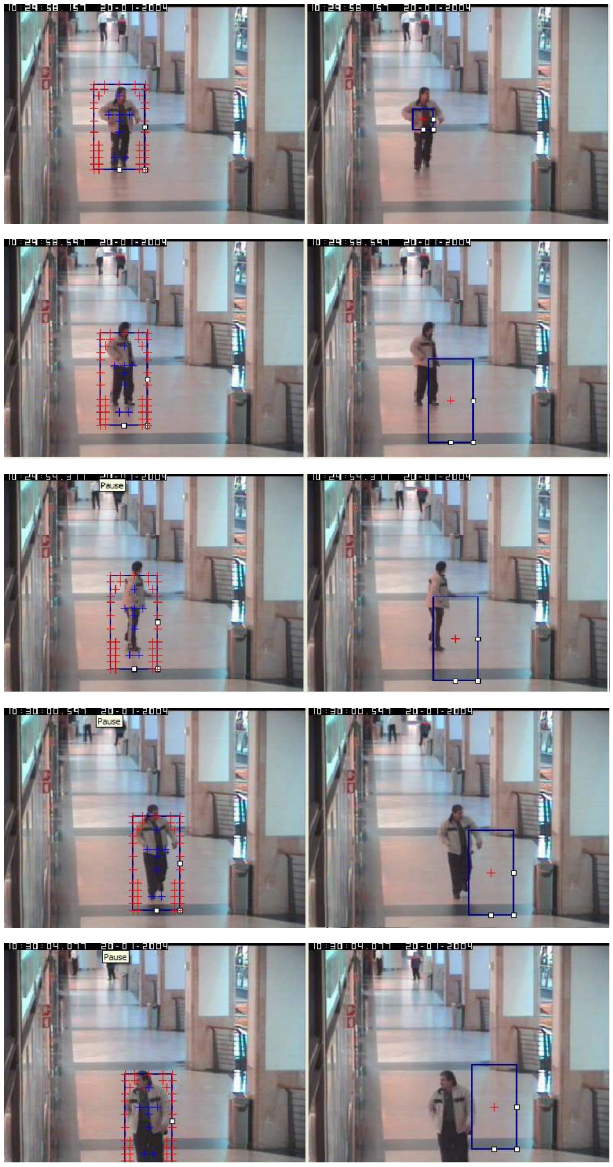

Figure 10: sequence shows behavior of box histogram mean-shift when box bounds the entire object (right-hand side sequence) versus masked histogram mean-shift (lefthand side sequence).