Intruder Tracking with a Pan-Tilt-Zoom Camera

Miroslav Trajković
Philips Research USA, 345 Scarborough Rd., Briarcliff Manor, NY 10510, USA
Miroslav.Trajkovic@Philips.com

Abstract

In this paper, we describe an automated intruder tracking engine (ITE), a software module that provides automated control of a Pan-Tilt-Zoom camera to follow a person and keep his/her image centered in the camera view. The current application of the ITE is to simplify the ease of use of security surveillance equipment. Instead of having to manually follow an intruder with the system joystick, the security operator simply needs to identify the subject to the ITE module, and it will track that person, leaving the operator free to more closely observe the behavior of the subject.

1. Introduction

With the fast growth of sensing and video technologies, it is increasingly becoming more economical to use video-based systems in surveillance and monitoring. These systems usually have a large number of cameras and at the moment, the operator has to monitor the entire facility on several monitors and manually follow an intruder or suspect with the system joystick. This is a tedious and often boring task, and therefore prone to human errors. The goal of our research is to develop software for facility wide tracking that will be able to perform automatic tracking of the target of interest. Hence, the operator will only need to identify the subject to the ITE module, and it will track that person, leaving the operator free to more closely observe the behavior of the subject.

While there has been significant amount of work on person tracking from the single static camera (e.g [3], [9]) there has been much less work on people tracking using multiple pan-tilt-zoom cameras, one exception being [1] where PTZ camera is used in collaboration with a parabolic mirror camera. The task of person tracking with moving camera is much more complex than the person tracking from a single camera, not only because the fact that camera moves, but also because the person is being seen from different view angles, and is often partly occluded.

The overall architecture of the system presented in this paper is given in Figure 1.

The first step in the tracking process is target selection. In the current implementation, the operator selects the target manually, by placing a rectangle on the torso (required) and head and part of trousers (optional) of the person being tracked. The rectangle placed on the person will be referred to as the Target Rectangle (TR), and we will track the person by tracking his/her Target Rectangle. Once the TR is selected, the parameters of its color model are found by computing its color histogram.

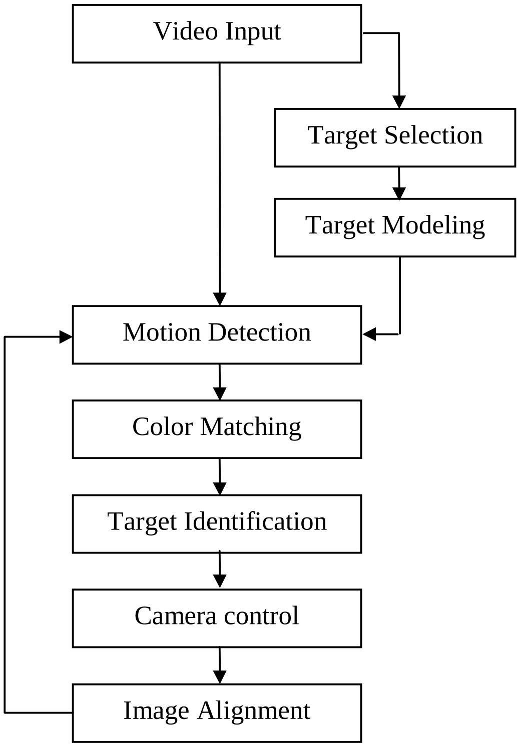

Figure 1: The overview of the tracker functionality.

We then use motion detection to find moving areas in the image, and use color matching to find the motion area that corresponds to the TR. Once the TR has been

identified, we issue velocity command so that camera moves towards the TR and acquires the next image.
Clearly, in order to perform independent motion detection, we have to identify the global motion (due to camera movement) and then align images before and after camera motion.

The paper is organized as follows. Section 2 provides details about target modeling. The motion detection algorithm is given in section 3, while the color matching algorithm is given in section 4. The target identification algorithm is described in section 5. The camera control is presented in section 6, and the finally, image alignment algorithm is given in section 7.

2. Person Appearance Modeling

The purpose of person modeling is to describe the visual appearance of a person in a unique way in order to be able to distinguish between the target being tracked and other people/objects in the camera field of view. There are many features that can be used for robust person modeling including color, texture, shape, facial characteristics etc. Unfortunately, precise and robust person modeling would typically require high resolution and computational resources that would prohibit real time implementation. In order to achieve real time performance of our system, we have modeled the target appearance using its color distribution, more precisely, the color histogram of the target rectangle. The color histogram has the advantage that it is relatively invariant to rotation and scaling, and does not require particularly high resolution. In addition, an efficient algorithm can be developed for histogram matching. Another possibility is to model the color distribution of the target using the mixture of Gaussians ([6]), but the disadvantage of this approach is that it requires the use of Expectation-Maximization algorithm, which may have problems with the convergence.

2.1 Color model

The color images acquired from camera are usually given by their R, G and B components. The disadvantage of the RGB color model is that it does not achieve the separation of brightness and chromacity components of the color. This separation is important if it is desired that the color model is invariant to changes in light intensity such as shadows. For this purpose, we have tested several color models (RGB, normalized RGB, YUV, HSI) and concluded that the HSI color model yields best results both in color/intensity separation and separation between different colors. Hence, we used this model in our system, although other color models have been implemented as well, and can be readily included.

The components of the HSI color model are hue, saturation and intensity. Hue represents dominant color as perceived by observer. Saturation refers to the relative purity, or the amount of white mixed with the color. Intensity is subjective measurement that refers to the light intensity of the color.

The formulas for conversion from RGB to HSI are non-linear and are given below:

$$\begin{aligned} I &= \frac{1}{3}(R+G+B) \\ S &= 1-\frac{\min(R,G,B)}{I} \\ H &= \cos^{-1}\left\{\frac{3}{2}\frac{R-I}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\} \end{aligned} \tag{1}$$

For more information about the HSI model and derivation of above-mentioned equations, please refer to [2].

2.2 Color histogram

In our implementation, the target color is given by its color histogram. In order to achieve intensity invariance, and to reduce the amount of computation, we only use the chromacity (i.e. hue and saturation) components of the image. Therefore, our color histogram is two-dimensional, having *m*×*n* bins, where *m* and *n* denote hue and saturation levels respectively.

We have experimentally determined that the hue component of the image color is more important than the saturation, and therefore we have finer resolution in hue than in saturation (*i.e.* $m > n$). For practical purposes, the number of bins should not be too high and default values in our implementation are $m = 32$, $n = 4$. An example of a color histogram for typical target is shown in Figure 2.

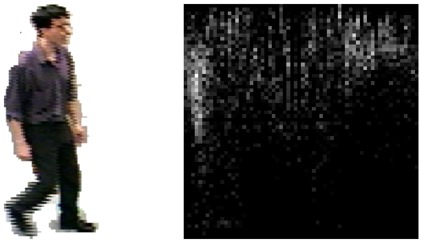

Figure 2: Typical target and its color histogram. Bright areas correspond to the high value of histogram bins.

2.3 Composite histogram

By examining the formula for hue in (1), it can be seen that for the color along the gray line (R=G=B) hue is not defined. Furthermore, small variations in color components will result in a large variations in hue. As an example for (R,G,B) = (101,100,100), (H,S,I) = (0, 0.0033, 100.333), while for (R,G,B) = (100,101,100), (H,S,I) = (2.0944, 0.0033, 100.333). From this example, we can conclude that for the colors close to gray, hue is extremely sensitive to noise. This is nicely illustrated by the color histogram given in Figure 2. One may notice that in addition to the peak corresponding to the person torso, there is almost uniform distribution across entire histogram corresponding to the black trousers.

In order to account for this behavior in hue component, we introduce a composite histogram representation in the following manner:

- Identify all the pixels that lie within the torus of radius *r* centered along the gray line and compute their gray level (*i.e.* intensity) histogram.
- Compute the color histogram of remaining pixels as described in section 2.2.

The color distribution of the target region is now represented by the composition of their color and gray histograms. The modified color histogram (for the pixels rich in chromacity) for the person shown in Figure 3 is shown bellow.

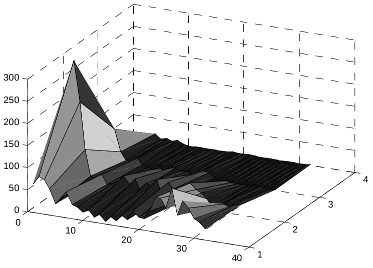

Figure 3: Modified color histogram obtained by removing gray pixels from the target shown in Figure 2. Compared to the histogram shown in Figure 2, this histogram has better separation of colors (fewer uniformly spread pixels) and two distinct peaks.

2.3 Histogram vector

The computational cost of histogram matching (see section 4) is directly proportional to the number of bins in the histogram. However, in typical histogram, only small number of bins are of importance (*i.e.* the bins whose colors are most frequent in the target), and therefore, significant speed-up may be obtained if only those bins are used.

As we are using MMX technology for histogram matching, the best choice for number of bins being used is of form $k = 2^l - 1$, and in most of our implementations $l = 4$ ($k = 15$ bins). It should be noted that sometimes more bins should be used. However, our experiments suggest that with 15 bins good trade-off between speed and performance can usually be achieved.

3. Motion Detection

To find moving areas, two consecutive gray level images are subtracted and the histogram of the difference image is computed. The difference image has elements in the range –255 to 255 and its histogram will usually have three modes as shown on Figure 4.

The largest mode is around zero differences and it corresponds to the static regions of the image. The difference in the image intensities from this mode is due to noise and changes in lighting, hence, the mean of this mode may not be at zero. Two other modes will occur, one to the left and one to the right of the central mode. These modes correspond to the pixels where change in image intensity is high and they correspond to the moving areas in the image.

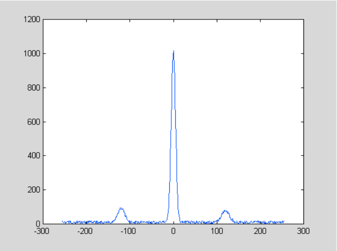

Figure 4: Three modes histogram model used for detection of moving areas.

The pixels in these modes can be considered as the outliers in the original Gaussian distribution. Since they will almost always constitute less than 50% of the total number of pixels, they can be detected using a robust estimation techniques such as RANSAC or the Least Median of Squares (LMedS, see *e.g.* [5]).

The complete algorithm for motion detection is given below:

1. Compute the difference image $D = I_2 - I_1$ and the histogram of the difference image h_D.
2. Compute $\mu = \operatorname*{median}_{i,j}(D_{ij})$.

Since the histogram of h_D of D is already computed, median of D can be computed very efficiently. Note that μ corresponds to the mean value of the main mode, and often this step can be skipped by assuming $\mu = 0$ (There is no change in lighting between two images).

3. Compute standard deviation of the main mode as
$$\sigma = 1.4826 \operatorname*{median}_{i,j}\left(D_{ij} - \mu\right)$$
4. Compute moving pixels as those that satisfy condition
$$| D_{ij} - \mu | > 2.5\sigma .$$

Filter the foreground image with a Box filter of a size identical to the target rectangle to obtain I_F. Ideally, the target we are looking for will correspond to the maximum of I_F. However, due to noise and the error of the method it may not always hold. Therefore, we threshold I_F and all the pixels whose value is higher than the threshold t ($t = c \times target\_size$, $0 < c < 1$), are considered as potential candidates. The best candidate is obtained by using a histogram matching algorithm described in the next section.

4. Histogram Matching

The problem that we are addressing in this section is the following: Find the area in a new image that has histogram most similar to the target histogram.

To do this, we first have to define a measure of similarity s of two histograms H_1 and H_2. In our work we adopted the similarity measure defined in [7] which is defined as the intersection of two histograms:

$$s(H_1, H_2) = \sum_{i,j} \min(H_1(i,j), H_2(i,j)) \qquad (2)$$

As the histograms are normalized, $0 \le s \le 1$. A typical similarity function between the target histogram and an image is given in Figure 5.

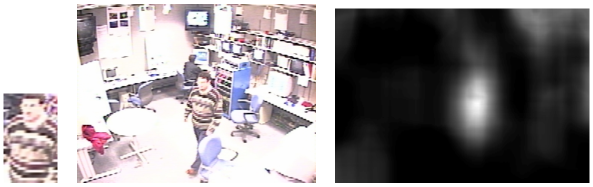

Figure 5: A template, an image and the histogram similarity function. Red areas correspond to high, while blue areas correspond to low similarity.

Given that we do not deal directly with histograms, but with the histogram vectors, equation (2) can not be directly used and we use the following, two-step algorithm:

Using a histogram vector **h** and the newly acquired image *I* compute the "palette image" *P*, whose elements are defined as

$$p_{ij} = \begin{cases} 0, & if\ I_{ij}\ is\ \ outside\ \ the\, vector\, \mathbf{h} \\ m, & if\ I_{ij}\ is\ \ inside\ \ bin\ \ m\ \ of\ \mathbf{h} \end{cases}$$

As the vector **h** has $k = 2^l - 1$ elements, the palette image *P* will have at most 2^l different values. Therefore, histogram similarity between the given target rectangle and the same size rectangle of the image *P* whose upper left corner is given by position (*i*, *j*) may be computed using the following formula

$$s_{ij} = \sum_{k=1}^{n} \min(\mathbf{h}, \mathbf{h}_p(i,j)) \qquad (3)$$

where $\mathbf{h}_\mathrm{p}$ denotes the histogram of the particular rectangle of the palette image, which has to be computed at every pixel location, which, for an entire image, is a major computational issue.

In order to speed-up the histogram computation we propose a two-pass, incremental algorithm for the histogram computation.

In the first pass, given a palette image *P*, we compute a matrix of vectors ($\boldsymbol{H}_h$) each element of this matrix being defined as a histogram of one line of the matrix *P*.

In the second pass, given a matrix $\boldsymbol{H}_h$, we compute a final palette histogram matrix ($\boldsymbol{H}_p$) by simply adding elements of $\boldsymbol{H}_h$ along the vertical line.

Additional improvement in speed is obtained by noticing that there is a recursive relation between the elements of the histogram matrices. Note that histogram vectors $\mathbf{h}_h(i,j)$ and $\mathbf{h}_h(i,j+1)$ may differ in at most two elements, (due to inclusion of *p*(i, *j*+*m*) and removal of p(*i*,j) into the $\mathbf{h}_h(i,j+1)$) and this difference is at most 1. In other words, if $\mathbf{h}_h(i,j)$ is computed, than $\mathbf{h}_h(i,j+1)$ may be computed in the following manner:

$$\begin{aligned} &\mathbf{h}_h(i,j+1) = \mathbf{h}_h(i,j) \\ &\mathbf{h}_h(i,j+1)[p(i,j+m)]{+}{+}; \\ &\mathbf{h}_h(i,j+1)[p(i,j)]{-}{-}; \end{aligned} \qquad (4)$$

requiring only one increment and one decrement !!

Similarly, for a known $\mathbf{h}_p(i,j)$, it holds:

$$\mathbf{h}_p(i,j+1) = \mathbf{h}_p(i,j) + \mathbf{h}_h(i,j+n) - \mathbf{h}_h(i,j) \qquad (5)$$

requiring only two vector additions which can be efficiently implemented in MMX technology.

The introduction of the palette image and above-mentioned histogram matching technique is a very powerful tool, which makes possible real time implementation of the histogram matching. The computational time does not depend on the histogram size, but only on the image dimensions. Typically, for the image of dimensions 320×240, histogram matching takes less than 100 ms.

5. Image Alignment

The purpose of image alignment is to find the mapping between two images obtained by the same camera at different positions, more precisely different pan, tilt and zoom settings. It is well known that in this case the mapping between the two images I_1 and I_2 can be expressed in the following form:

$$\mathbf{p}' = sQRQ^{-1}\mathbf{p} = M\mathbf{p} \qquad (6)$$

where **p** and **p’** denote the homogeneous image coordinates of the same world point in the first and the second image, *s* denotes the scale change, *Q* is the internal camera calibration matrix, and *R* is the rotation matrix between two cameras/camera locations.

Alternatively, this equation may be written as:

$$x' = \frac{m_{11}x + m_{12}y + m_{13}}{m_{31}x + m_{32}y + m_{33}}$$
$$y' = \frac{m_{21}x + m_{22}y + m_{23}}{m_{31}x + m_{32}y + m_{33}} \quad (7)$$

where (x, y) and (x', y') are pixel locations in the first and second image respectively, and $M = [m_{\text{ij}}]_{3\times3}$ is the homography matrix that maps (aligns) first image to the second.

The main problem of image alignment, therefore, is to determine the matrix *M*. From equation (6), it is clear that given *s*, *Q* and *R* it is straightforward to determine matrix M. While it is true in the *ideal* case, it is not always applicable in practical situation, the main reasons being that:

- we do not know exact values of *s*, *Q* and *R*;
- strictly speaking, equation (6) holds only if camera center and the rotation center are identical, which, in most PTZ cameras, is only approximately true;
- in order to retrieve precise values of camera settings, i.e. pan, tilt and zoom values, the camera has to stop which will create unnatural motion; and
- in some PTZ camera systems, retrieving camera settings can take considerable time.

Hence, in our work, we compute the alignment matrix *M* directly from the images, therefore requiring no information about camera position and calibration. Our algorithm requires point matches between two images and is performed in following steps:

1. Scale-down images I_1 and I_2.
2. Find corners at low resolution using any corner detector. In particular, we used MIC corner detector ([8]) since it has very good trade-off between speed and performance. Other corner detectors, such as Harris ([4]) may be used as well, but the real time issue should be taken into account.
3. Using robust methods (RANSAC algorithm) find the alignment matrix M_l at low resolution.
4. Find corners at a finer scale images (half the image size in our case).
5. Perform corner matching at a finer scale image by finding the best corners around positions predicted by M_l.
6. Using robust methods (RANSAC algorithm) find the alignment matrix M_l at finer resolution.

It should be noted that in our implementation zoom was fixed and therefore at low resolution, we made assumption that the image motion can be approximated by translation and rotation in the image space. In other words, at low resolution, we used the following approximation of equation (7):

$$\begin{aligned} x' &= x\cos\alpha - y\sin\alpha + t_x \\ y' &= y\sin\alpha + x\cos\alpha + t_y \end{aligned} . \quad (8)$$

This approximation has several advantages:

- Due to lower number of parameters, RANSAC is exponentially faster.
- It is more stable than full modeling, as at low resolution, there are many parameters that can be neglected, and by using them no real improvement is obtained.

If the zoom has not been fixed, then instead of formula (8) we should use

$$\begin{aligned} x' &= s(x\cos\alpha - y\sin\alpha) + t_x \\ y' &= s(y\sin\alpha + x\cos\alpha) + t_y \end{aligned} . \quad (9)$$

where *s* denotes the scaling (zooming) factor. By introducing new independent variables $a_1 = s\cos\alpha$, $a_2 = s\sin\alpha$, equation (9) becomes:

$$\begin{aligned} x' &= a_1x - a_2y + t_x \\ y' &= a_2x + a_1y + t_y \end{aligned} , \quad (10)$$

and RANSAC can still be executed very quickly.

6. Finding the Best Candidate

The drawback of histogram matching is that, in general, it does not provide for a unique solution. Namely, it may happen that more than one area in the image has a histogram similar to the target histogram, and our task is to: (1) find all these areas (target candidates); and (2) determine which one of them is most likely to correspond to the target location.

6.1 Finding target candidates

Target candidates are determined as local maxima of the histogram similarity function, providing their similarity function is higher than a certain operator adjustable threshold. The local maxima are found using a 5×5 window, and after that all the maxima whose mutual distance is less than *d* (default 20) pixels are merged into a unique local maximum.

Note that we consider only those maxima that belong to the foreground region (only moving objects), and additional one which is in the vicinity of the latest target position (to allow for the stationary target). This greatly reduces the number of false positives, but unfortunately, they can still occur.

6.2 Selecting the best candidate

There are two approaches we considered for selecting the best candidate:

- Temporal approach, *i.e.* select a candidate which is closest to the predicted target position;
- Template matching approach, *i.e.* select a candidate that is most similar to the target using a template matching as the measure of similarity. For the time being this is not implemented.

6.3 Predicting target position

Due to the highly unconstrained nature of the human motion, we are only using a small number of frames (five) to predict the position of the target in the following frame. The target motion is modeled using a constant velocity model:

$$\begin{aligned} x_t &= x(t) = a_0 + a_1 t \\ y_t &= y(t) = b_0 + b_1 t \end{aligned} \qquad (11)$$

where x_t and y_t denote the current target position (*i.e.* position of the center of the target), and a_0, a_1, b_0, b_1 denote the parameters of the constant velocity model. Given the velocity parameters a_0, $a_{1,}$ the expected coordinates of the point in the next frame are given by:

$$\begin{aligned} \tilde{x}_{k+1} &= a_0 + a_1(k+1) = x_k + a_1 \\ \tilde{y}_{k+1} &= b_0 + b_1(k+1) = y_k + b_1 \end{aligned} \qquad (12)$$

However, since the target can slow down rapidly, we assume that the predicted position may be anywhere in the line connecting points $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$, which may be represented as

$$\mathbf{p} = \mathbf{p}_k + \mathbf{n} \cdot t,\ \ 0 \le t \le 1 \qquad (13)$$

and $\mathbf{n} = [a_1\ b_1]^T$.

The best candidate is determined as the one closest to the line given by (13).

7. Experimental Results

The tracker engine described in this paper performs tracking of a person with moving camera in the real time, with frame rate of 5 -10 Hz. When a person is being tracked in an area with no other moving objects, the tracker achieves excellent performance, being able to track the person over a wide range of velocities (from very slow, to very fast, almost running). It is also able to track the person in very different camera positions, and generally, has very low dropout rate. Our experiments show that the most difficult tracking scenario occurs for the camera set at low heights and the person walking quickly straight below the camera. In this case, the angular velocity of the person is very high, and due to physical restrictions on the camera speed, the person can be lost. We are currently working on the camera control mechanisms to overcome this problem.

When there are multiple moving people in the field of view, the matching rate varies from sequence to sequence depending how different is the color distribution of the target being tracked and the other moving objects. If the tracked target meets with another moving object, there is no guarantee that the tracker will remain on the desired target. This problem can be somewhat overcome by including more elements in the histogram vector, by including some spatial information into the histogram, and by including some additional information about the trajectory. Even then, this is a very difficult task that can present a challenge even to the human operators.

When the tracked target is sufficiently different from other objects being tracked, our system is able to track it successfully over a period of time and is not particularly affected by occlusions and people crossing.

8. Conclusion

In this paper we have presented a system for intruder tracking and described the basic building components of the system. The system operates in real time (5-10 frames per second) and allows for the partial or temporary complete occlusion of the target being tracked. It achieves very good performance when there is only one target in the field of view, or the target is sufficiently different from the other moving objects that occlude it. The main novelties of the system are a real time feature points based registration module, the composite color-gray histogram description of the target, and a fast recursive algorithm for histogram matching.

9. References

[1] Cui Y. *et al*,"Indoor Monitoring via the Collaboration Between a Peripheral Sensor and a Foveal Sensor", *in Workshop on Visual surveillance*, Bombay, India, 1998.

[2] Gonzalez R., Woods R., "Digital Image Processing", 1992.

[3] Haritaoglu I., Harwood D. and Davis L., "W4 - Real time detection and tracking of people and their parts". *FGR 98*.

[4] Harris C. and Stephens M., “A combined corner and edge detectior”, *4th Alvey Vision Conference.*, pp. 147-151, 1988.

[5] Meer P., Mintz A. Rosenfeld A. and Kim D.Y., “Robust regression methods for computer vision: A review”, *Intl. Journal of Computer Vision*, vol. 6, pp. 59-70, 1991.

[6] McKenna S., Raja Y. and Gong S., “Tracking colour objects using adaptive mixture models”, *Image and Vision Computing*, vol.17, pp. 225-231, 1999.

[7] Swain M. and Ballard D., “Color Indexing”, *International Journal of Computer Vision*, (1991), 21-32.

[8] Trajković M. and Hedley M., "Fast Corner Detection", *Image and Vision Computing*, vol. 16, pp. 75-87,1998.

[9] Wren C. *et al*, "Pfinder: real-time tracking of the human body", TR 353, MIT Media Lab, 1995.