Robust pedestrian detection and tracking in crowded scenes

Yuriy Lypetskyy
JOANNEUM RESEARCH,
Graz, Austria
yuriy.lypetskyy@joanneum.at

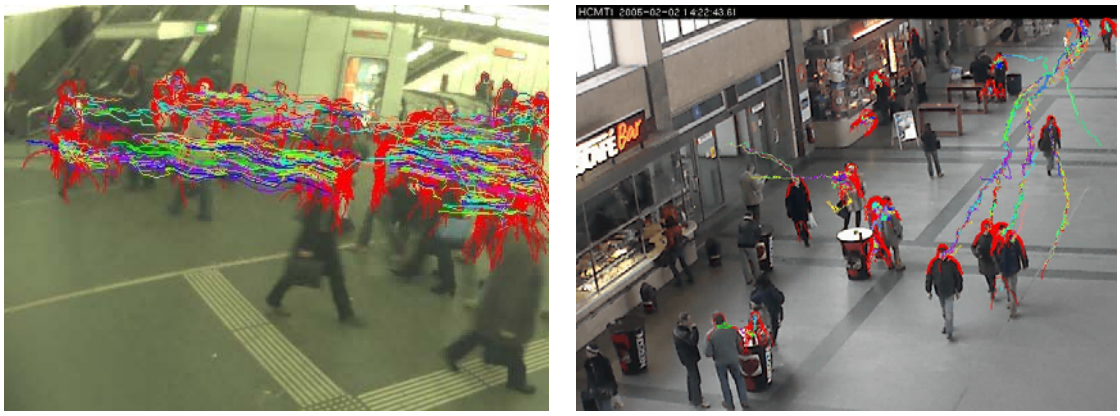
ABSTRACT

This paper presents a vision based tracking system developed for very crowded situations like underground or railway stations. Our system consists on two main parts – searching of people candidates in single frames, and tracking them frame to frame over the scene. This paper concentrates mostly on the tracking part and describes its core components in detail. These are trajectories predictions using KLT vectors or Kalman filter, adaptive active shape model adjusting and texture matching. We show that combination of presented algorithms leads to robust people tracking even in complex scenes with permanent occlusions.

Keywords: Tracking, crowded scenes, Kalman filter, active shape model, texture matching

1. INTRODUCTION

The presented system is developed in order to detect and track people in crowded situations (Figure 1). It can be used both in indoor (like in underground stations) and outdoor scenes (like in public transport junctions). The main goal of the system is people counting, but it can be used for other purposes related to the trajectory analysis, as well (e.g. detection of overcrowding situations or tourist flow estimations).



(a) Underground Station (b) Railway Station

Fig 1. Typical real scenes for people tracking task.

In recent years pedestrian detection and tracking have become a very active area due to the increasing number of possible applications. Tracking of isolated individuals seems to be realizable task, but to cope with crowded situations there are more complicated algorithms needed. The main problem of tracking in crowds is a permanent strong occlusion of pedestrians. To avoid them, one could use top-view camera positions, like in [7] or [8]. However, this approach has

severe restrictions: camera installation is often not possible in real-world situations due to architectural constraints, the observing area is quite small, and the perspective is unusual and non-informative for a human.

Another solution would be estimation of crowd density by using of some classification framework, as it is done in [12]. But it is quite difficult to learn classifiers so that they give robust results in different crowded scenes. People are often partially occluded and this makes classification task even more complicated. To avoid these problems, we use another approach. The main idea is to detect and track human head-shoulder regions. We use an active shape model algorithm to detect people candidates. Each shape will be tracked through the scene using one of two prediction strategies. To cope with occlusions we use adaptive shapes adjusting and texture matching during the tracking process. We show that our tracking implementation works stable even in strongly occluded scenes. The detailed description of these algorithms is given in the following sections.

2. SEARCHING OF PEOPLE CANDIDATES

In order to track people, first we have to find them. The choice of the pedestrian detection strategy is made due to the expected scene and image conditions. It is very hard to extract blobs or to do some classification in crowded scenes with strong occlusions (which is the normal situation e.g. for underground stations). Instead, we search for so called "Ω-shapes", which represent the contours of the heads and shoulders of humans.

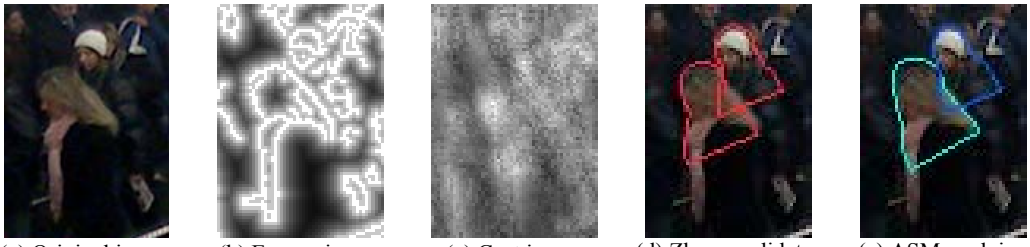
First, we use a simple foreground mask, based on histograms for each image pixel on a reduced set of 64 grey values. The histogram is accumulated for every new frame and its current highest grey value frequency is selected as background. We use individual background models for each image colour channel. After subtraction of background models from the corresponding colour channels of the current image we do thresholding and apply a logical OR operation to obtain the masking image. Masking is implemented to speed-up further algorithms – all calculations will be done only in the foreground regions.

The searching itself is then done in two stages – fast searching of all possible candidates and a following deeper analysis of each candidate position. The first stage is based on the algorithm described by Zhao in [2]. The goal is to find shapes similar to a model (which has a form of Ω-shape) represented by 23 points. This is done by analyzing of the gradient image, obtained by applying of Canny edge detector on an input RGB image. There will be a cost image created then, based on a convolution of a model over the gradient image. After cost image thresholding and some morphologic operations we have initial people candidates (Figure 2). A more detailed description can be found in [1].

These candidates are inputs for fine tuning and selection with an active shape model (ASM) algorithm. The implementation of ASM is close to the method described by Cootes in [3]. A set of 130 Ω-shapes has been extracted manually and used to set up an Eigenshape space. The whole set represents pedestrian head-shoulder contours from all possible viewing angles. Only the first 5 Eigenshapes are used for modelling variations within shape space. ASM matching is implemented as an iterative procedure:

1. For each contour model point of the ASM find the best matching image point using a cost function;
2. Compute an update of ASM parameters so that the model fits best to the image points;
3. Coerce ASM parameters for shape, scale and rotation to within allowed range;
4. Repeat the process n times.

After 40 iterations only shapes with high enough cost values are accepted and used for tracking.



(a) Original image (b) Energy image (c) Cost image (d) Zhao candidates (e) ASM applying

Fig 2. Searching of people candidates.

3. TRACKING

Robust tracking in crowded scenes becomes difficult task due to the numerous occlusions. This section gives detailed description of our tracking implementation. It includes prediction with Kanade Lucas Tomasi (KLT) tracking points or Kalman filter, ASM transformations over the trajectories history, texture matching, and the merging of shapes.

3.1 Shape projection to the next frame

To project detected people candidates (Ω-shapes) into the next frame, we use tracking points from the KLT algorithm [4]. There will be 1200 best trackable points used (for the image size 640x480 pixels), which are created in a foreground region. Also we enforce that created points are lying at least 4 pixels away from each other. This is done to avoid the aggregation of points in one region with no points in another. Points are constantly tracked over the frames; lost points are replaced with new ones (Figure 3). To predict position of each shape in the next frame, we use all tracking points lying within this shape and calculate average displacement of these points into the next frame.

Fig 3. Tracking with KLT vectors.

It is possible that some shapes do not contain any tracking points (e.g. too low contrast or there were no more trackable points left for this image region). In this case we use prediction with a Kalman filter. The implementation is very close to the one described in [5]. Given the state vector $X(k|k)$, the covariance matrix $P(k|k)$ (respective to the iteration k), state transition matrix $F(k)$ and 4x4 process noise covariance Q, at the iteration $k+1$ there will be estimated the new state vector $X(k+1|k)$ and the new covariance matrix $P(k+1|k)$. As soon as the next value of the new position is obtained, the real position of the Ω-shapes and its velocity will be estimated.

X(k|k) has the form *[x vx y vy]'*, where *x*, *vx*, *y* and *vy* respectively are the horizontal position, the horizontal velocity, the vertical position and the vertical velocity the object at the iteration *k*. *F(k)* must be a 4x4 matrix and represents the model used to predict the new state vector. For example, if *F(k)* is the identity matrix, the predicted new state vector *X(k+1|k)* is exactly the same as *X(k|k)*. In our case,

$$F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

X (k+1|k) is the state vector predicted according to the state prediction equation:

$$X(k+1 \mid k) = F(k) \cdot X(k \mid k) \tag{2}$$

P(k+1|k) is the covariance matrix of X(k) predicted as follows:

$$P(k+1 \mid k) = F(k) \cdot P(k \mid k) \cdot F'(k) + Q \tag{3}$$

After that, covariance matrix will be updated:

$$P(k+1 \mid k+1) = P(k+1 \mid k) - W(k+1) \cdot S(k+1) \cdot W'(k+1) \tag{4}$$

Here *W(k+1)* defines the filter gain:

$$W(k+1) = P(k+1 \mid k) \cdot H'(k+1) \cdot S'(k+1) \tag{5}$$

Where:

$$S(k+1) = H(k+1) \cdot P(k+1 \mid k) \cdot H'(k+1) + R \tag{6}$$

R is the observation noise covariance matrix. It is a 2 by 2 diagonal matrix.

The output state vector *X(k+1|k+1)* is than determined by equation:

$$X(k+1 \mid k+1) = X(k+1 \mid k) + W(k+1) \cdot (z(k+1) - zp(k+1 \mid k)) \tag{7}$$

Where:

$$zp(k+1 \mid k) = H(k+1) \cdot X(k+1 \mid k) \tag{8}$$

z(k+1) is an array with the real data. It refers to the *(x,y)* coordinates, so that it is an array with 2 entries (it is not related to the velocity).

H(k+1) is the observation matrix. For the motion tracking it is the following constant matrix:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{9}$$

As the state vector *X(k+1|k+1)* is not an output, it must be built in advance for the next iteration *(k+1)*.

Shape predictions using KLT vectors and the proposed Kalman filter are shown in Figure 4.

(a) Prediction with KLT

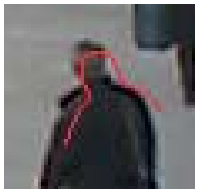
(b) Prediction with Kalman

Fig 4. Trajectories predictions.

3.2 Shapes adjusting

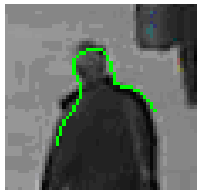
After the prediction of shapes positions using either KLT points or Kalman filter (depending on the available information) from the previous frame to the next frame we apply ASM matching again on those predicted positions. The idea here is that people change the geometry of their head-shoulder contour only very slightly in two successive frames. Therefore we use only one initial shape model – the one which was calculated in the previous frame. The number of iterations is also limited to 7 in the current implementation. The new shape will differ only a little bit from the previous frame, and the possible errors in prediction estimation will be corrected with the ASM process (Figure 5).

Finally, we check if cost value of each obtained shape is high enough (as it described in Section 2), and if it lies at least 75% in foreground. If this is not the case, we discard changes, and only keep the predicted position. We don't delete that shape (it can be just temporary occluded), but try to use the Kalman prediction further, but at most for 5 frames.



(a)

(b)



(c)

Fig 5. Shape adjusting. (a) Shape position on the previous frame. (b) Predicted shape position for the current frame. (c) Shape position after ASM applying.

3.3. Texture matching

To be accepted as successfully tracked, each shape must pass a texture matching check too. It is possible that it is occluded, but still has a big enough ASM cost value. The further tracking of that shape would be wrong. To do texture matching between the same candidates in two successive frames we apply a cooccurrence-matrix feature approach.

The cooccurence-matrix is a square matrix whose size represents number of grey levels taken (we reduce grey levels by factor 16). These matrixes can be used to describe the relation between each pixel and some its neighbour, displaced from the original pixel by a constant vector *D*. In general, if an image has low contrast, neighbour pixels are similar, so bins will be assigned mostly on the diagonal. And on an image with high contrast cooccurrence-matrix will be empty along the diagonal and more concentrated near the top-left and bottom-right corners.

Cooccurrence-matrix *O* can be calculated by performing of following operations on each pixel of input image *I*. For each pixel:

$$i_{next} = I\,[\, I + D_i, j + D_j\,];$$
$$j_{next} = I\,[\, i, j\,];$$
$$O\,[\, i_{next}, j_{next}\,]\;{+}{+};$$
(10)

Here:

I is extracted from the whole input image as the bounding box of the current shape;

i and j are row and column indices ($i = [0..Nrows, j = [0..Ncols]$);

D is a constant displacement vector (we use $D = [2, 2]$).

These calculations will be done using masking image – only content of Ω-shape will be used.

Then we extract features from the obtained matrix as the following statistical parameters:

$$\text{Max} = \max_{i,j}(c(i,j)) \tag{11a}$$

$$\text{Energy} = \sum_{i,j} c(i,j)^2 \tag{11b}$$

$$\text{Entropy} = -\sum_{i,j} c(i,j)\log c(i,j) \tag{11c}$$

$$\text{1st Order Moment} = \sum_{i,j} (i-j)\cdot c(i,j) \tag{11d}$$

$$\text{Inertia} = \sum_{i,j} (i-j)^2\cdot c(i,j) \tag{11e}$$

$$\text{Local Homogeneity} = \sum_{i,j} \frac{1}{1+(i-j)^2}\cdot c(i,j) \tag{11f}$$

$$\text{Cluster Shade} = \sum_{i,j} ((i-\mu_i)+(j-\mu_j))^3\cdot c(i,j) \tag{11g}$$

$$\text{Correlation} = \sum_{i,j} \frac{(i-\mu_i)(j-\mu_j)}{\sigma_i\sigma_j}\cdot c(i,j) \tag{11h}$$

$$\text{Haralick's Correlation} = \frac{\left(\sum_{i,j} i\cdot j\cdot c(i,j)\right) - \mu_i\mu_j}{\sigma_i\sigma_j} \tag{11i}$$

Here:

c is the cooccurence-matrix;

$$\mu_i = \sum_i i\sum_j c(i,j),\ \mu_j = \sum_j j\sum_i c(i,j),\ \sigma_i = \sum_i (i-\mu_i)^2\sum_j c(i,j),\ \sigma_j = \sum_j (j-\mu_j)^2\sum_i c(i,j) \tag{12}$$

Altogether we have 9 features. As we analyze each image colour channel separately, we have 3 cooccurrence-matrixes and 27 features. This feature vector is compared with the vector from the previous image. If the distance between vectors is less than some threshold value, shape is considered as successfully tracked, and its cooccurence-matrix is replaced with the new one.

3.4 Shapes merging

The merging of shapes is necessary because searching of new shapes and tracking of existing ones are independent processes. Merging has to be done very carefully since there is a danger to loose some good trajectories. Our strategy is

to track all non-identical trajectories that appear in a scene. The reason is that some trajectories can easily get lost in difficult scenes with strong occlusions, but others may survive for a longer time.

To do the merging, we compare similarity of all new shapes to all shapes that are currently being tracked. We calculate mean distance of all 23 points of the two shapes, and if that distance is fewer than some thresholding value (e.g. 10 pixels), we merge them. Merging means that we select the shape with a bigger ASM value (i.e., it fits better on a person), and delete the other.

3.5 Trajectory generation algorithm

The operation of the tracking algorithm can be summarized as follows:

1. Update background model after new frame has come;
2. Track KLT vectors from the previous frame;
3. Project all existing shapes into the next frame, using either KLT vectors or Kalman filter;
4. Check for each shape if it fulfils the following conditions:

   a) ASM cost value is high enough after 7 iterations of matching with the initial shape model from the previous frame;

   b) Shape lies at least 75% on foreground;

   c) Feature vector, based on cooccurrence-matrixes, is similar enough to the obtained on a previous frame.
5. Accept tracking result, if the shape confirms all these conditions, or use prediction from the step 2. If the shape could not be tracked 5 successive frames, delete the trajectory;
6. Calculate new candidates, using successive executing of Zhao and ASM algorithms;
7. Do merging of new candidates with the existing ones.

By using of verifications, described at the step 4, we want to ensure that a shape is being correctly tracked. Its parameters, like Ω-shape model and texture feature vector, are being continually updating from frame to frame. Such strong conditions are needed because in crowded scenes shapes disappear often due to occlusions. We try to find immediately when this is happening, and use predictions with the hope that we can restore trajectory later. After a trajectory could not be restored 5 frames one after another, we close it. Finished trajectories come to the final, post-processing stage.

The tracking system at work can be seen on Figure 6. It shows an underground station in Vienna with the people coming out from the train. The longest trajectories have a history of about 3 seconds (45 frames).

Fig 6. Tracking example.

3.6 Post-processing stage

After a trajectory has died, it comes to the post-processing stage. Post-processing helps as to remove noise and to smooth trajectories. First, we accept only long enough trajectories. If the trajectory has a length less then e.g. 10 frames, we reject it. Also, we do trajectories smoothing using simple averaging filter with a fixed kernel size. We compute a sliding window average over the trajectory history.

4. DISCUSSION

Our experimental results (details presented in [1]) show that the tracking system works robust even in very difficult situations. The use of ASM Ω-shape models helps us to find close to 100% of all people in a scene. There will be a certain number of false alarms found too, but part of them will be rejected during the tracking – we track only shapes that keep their form over the video sequence. Tracking in crowded scenes is not easy due to permanent occlusions, that is why we keep as many trajectories as possible and merge only very similar of them. Normally one person "collects" several trajectories – part of them will be successfully tracked over the scene. It brings us robustness, but from the other side it affects performance strongly. To speed-up the tracker, we use a background mask model – all calculations will be done in the foreground regions only. Performance remains our main problem – the computation time is now around 2-3 fps in average on a state-of-the-art PC. Our present goal is to optimize algorithms implementations in order to make the tracking close to real-time. Also we want to try to combine our system with some people classifier.

ACKNOWLEDGEMENTS

This work was supported by the Austrian Ministry of Traffic, Innovation, and Technology (BMVIT) under program I2 Intelligent Infrastructure. We appreciate the support of *Wiener Linien* (Vienna, Austria) and *Grazer Verkehrsbetriebe* (Graz, Austria) during development and test of the prototype system.

REFERENCES

1. O. Sidla, Y. Lypetskyy, N. Braendle, and S. Seer, “Pedestrian Detection and Tracking for Counting Application in Crowded Situations”, *Proc. AVSS2006*, pp. 70-70.
2. T. Zhao, and R. Nevatia, “Bayesian Human Segmentation in Crowded Situations”. *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2003, vol.2. pp.459-466.
3. T. Cootes, C. Taylor, D. Cooper, and J. Graham, “Active shape models – their training and application”, *Computer Vision Image Understanding*, vol. 61(1), 1995, pp.38-59.
4. C. Tomasi and T. Kanade, “Detection and Tracking of Point Features”, *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991.
5. V.Chang, N.Kehtarnavas, “A Smart Camera Application: DSP-based People Detection and Tracking”, *Journal of Electronic Imaging 9(3)*, pp. 336-346, July 2000.
6. S. Lee, J.H. Kim and F.C.A. Groen, “A fast computational method for minimum square error transform”, *Proceedings 9th IAPR International Conference on Pattern Recognition* Rome, Italy, Nov. 1988, pp. 392-394.
7. L. Snidaro, C. Micheloni, and C. Chiavedale, “Video Security for Ambient Intelligence”, *IEEE Trans. Systems, Man and Cybernetics A* 35(1), 2005.
8. A. Albiol, V. Naranjo and I. Mora, “Real-Time High Density People Counter using Morphological Tools”, *Proc. ICPR’00*.
9. J. Rittscher, P.H. Tu and N. Krahnstoever, “Simultaneuous Estimation of Segmentation and Shape”, *Proc. CVPR’05, San Diego*, 2005.
10. X. Lui, J. Rittscher. A. Perera, and N. Krahnstoever, “Detecting and Counting People in Surveillance Applications”, *Proc. AVSS’05*
11. C. Sacchi, G. Gera, L. Marcenaro and C.S. Ragazzoni. “Advanced image-processing tools for counting people in tourist site-monitoring applications”, *Signal Processing 81*, 2001, pp. 1017-1040.
12. D.B. Yang, H.H. González-Banos and L.J. Guibas. “Counting People in Crowds with a Real-Time Network of Simple Image Sensors”. *Proc. ICCV’2003*.