Pedestrian Detection and Tracking for Counting Applications in Crowded Situations

Oliver Sidla, Yuriy Lypetskyy
JOANNEUM RESEARCH,
Graz, Austria
os@slr-engineering.at

Norbert Brändle, Stefan Seer
arsenal research,
Vienna,Austria
Norbert.Braendle@arsenal.ac.at

Abstract

This paper describes a vision based pedestrian detection and tracking system which is able to count people in very crowded situations like escalator entrances in underground stations. The proposed system uses motion to compute regions of interest and prediction of movements, extracts shape information from the video frames to detect individuals, and applies texture features to recognize people. A search strategy creates trajectories and new pedestrian hypotheses and then filters and combines those into accurate counting events. We show that counting accuracies up to 98 % can be achieved.

1. Introduction and Scope

We present a system to detect and track pedestrians in very crowded situations for the purpose of counting them. Applications range from railway transport security, pedestrian traffic management, detection of overcrowding situations in public buildings to tourist flow estimations.

Due to its vast number of applications, vision-based pedestrian detection and tracking is a very active research area in the computer vision community. Much progress has been made in the detection and tracking of individuals in groups, where the algorithms are often tested with small amounts of people in laboratory settings [6], [7]. The individuals’ trajectories can be used for counting passing people and be implemented by using virtual gates or tripwires: users can draw straight lines at any location in the field of view, and the algorithm continuously counts how many people are passing it (see Figure 1). Liu et al. [9] apply the human group segmentation algorithm presented in [7] and perform experiments with groups of 5 people. Sacchi et al. [15] present a real world outdoor counting application and report a mean error of 10%. Realistic scenarios, however, do not only contain loose groups of people but rather crowds of individuals like those shown in Figure 1a. For camera views with shallow angles, the mutual occlusions become so severe that no tracking algorithms can handle them effectively, even with a multi camera approach, [8].

One way to avoid severe occlusions is to use top-view cameras, like in [8] or [11]. Actually most of today’s commercially available video-based people counter solutions are based on those configurations. We consider people counting as an added value to security and safety applications and thus want to avoid top view cameras with limited sensing areas and unfamiliar perspectives for security personnel.

When dealing with oblique cameras, one solution to avoid group segmentation is to directly estimate the crowd density by extracting significant features and feed those into a classification framework to obtain an estimation of the number of people as in [12], [13]. The accuracy of such systems strongly depends on the training set and on the choice of the feature set. Lin et al. base their people counting on the recognition of head-like contours with on Haar wavelet features and SVM classification, [14]. While they provide quantitative results for model worlds with 125 person-like puppets, they do not provide quantitative results on real world data, due to the lack of ground truth. Our approach for

Figure 1. Subway platform scenario,
a) Camera 1  b) Camera 2
counting passing people in crowds is based on a novel integration of multiple hypotheses for the detection and tracking of human head-shoulder regions. The algorithm does not produce unique trajectories, but we show that after a one-time estimation of a systematic correction factor based on manually labelled ground truth data accuracies up to 98% can be achieved for real-world scenarios.

This paper is organized as follows. Section 2 first gives an system overview and describes the different algorithms and their relationship. Section 3 gives a detailed analysis of two real-world tests of the system which is finally discussed in Section 4.

2. System Description

Experience from previous work shows clearly that tracking algorithms which are based on background subtraction and blob analysis are severely limited in their performance as soon as pedestrians tend to accumulate. Upon this perception we decided that a background model should at most aid in detecting regions of interest (ROI) and that individuals must be located by other means. Our algorithm is based on the observation that pedestrians, regardless of their clothing and hairstyle, display a typical Ω-like shape, which is formed by their heads and shoulders. The detection of these shapes, masked by a ROI filter which is built from a simple background model and motion detection, is the core of our proposed people tracking system. Every pedestrian is characterized by a description which is calculated from a co-occurrence matrix feature vector. This description serves as a means to recognize individuals from frame to frame and also to coarsely detect occlusions or a disappearing pedestrians. In such cases the history of a known trajectory is extrapolated with a Kalman Filter for a short period of time to fill these gaps. The counting of passing pedestrians is finally achieved with a virtual gateway and a simple trajectory-based heuristic.

2.1 ROI Processing

For every new video frame a simple foreground mask is created which decides whether the underlying pixel is part of a pedestrian or not. All foreground pixels constitute the region of interest (ROI) on which all further computations take place. We implemented an algorithm which uses a histogram for each pixel on a reduced set of 64 grey values. The histogram is accumulated for every new frame and its highest gray value frequency is selected as background. At regular intervals the pixel wise histograms are divided by a constant factor to reduce the absolute numeric values of the histogram entries. New dominant grey values can so much easier gain control of the histogram in case of illumination changes or new stable background objects.

In addition to the background computation 1200 salient image points are constantly tracked frame to frame using the Kanade Lucas Tomasi (KLT) algorithm [2]. These points provide motion information for the whole image and they are used in two ways. Firstly, clusters of points which move in the same direction with similar speed are added to the ROI, they are used to project the location of detected pedestrian shapes into future video frames.

2.2 Shape Detection

Pedestrian detection is divided into two parts. A fast contour searching method detects candidates for Ω-like shapes within the ROI, it is based on principles described by Zhao and Nevatia [1]. A pedestrian contour is represented as model which consists of 23 points \( M_i = (x_1, y_1, ..., x_{23}, y_{23}) \). A local angle \( a_i \) of a model point is defined as the direction of the vertex on its right hand side when the contour is traversed in clockwise direction.

The detection process starts by applying a Canny edge detector to the input color image, masked by the ROI. We then spread out from the so computed edges (which are assigned a value of 1.0) an exponentially decaying eight pixel wide hull of gradient values \( E = e^{-0.25*D} \), \( D \) being the distance to the closest original edge pixel. The result is a gradient image map \( O \) in the range \([0 .. 1]\). An angle map \( R \) is computed directly from \( O \) by calculating local pixel wise orientations on it. Then, for every possible location \((x, y)\) of the reference shape in the input image, a cost function \( S \) is computed by

\[
S(x, y) = \frac{1}{23} \sum w_i O(M_i, O(x, x+y, y)) \cos(a_i - R(M_i, x+y, y))
\]

(1)

The weights \( w_1, ..., w_{23} \) are used to assign more importance to head points and reduce the influence of shoulder points of the model. Figure 2 displays the operation of the algorithm on an example. The threshold cost image \( S \) is filtered to close gaps and fill holes. Local maxima of \( S \) in the resulting blobs finally give the desired locations of head candidates. In order to improve the overall detection rate, we match not only one but with 5 different shape models and use all resulting candidates.

Perspective changes within the image are handled by scaling the reference shape points accordingly before they are used for the computation of \( S \). The necessary factors are chosen manually for a scene by defining 4 scales at the image corners and bilinear interpolation in between. The Zhao algorithm results are further analyzed by fitting active shape models (ASM). Only those candidates where the ASM can match well to the underlying image
gradients is the pedestrian hypothesis accepted. Our ASM implementation follows closely the method described by Cootes et al. in [3].

A set of 130 Ω-shapes has been extracted manually and used to set up an Eigenshape space. Each contour again consists of 23 vertex points, the whole set represents pedestrian head-shoulder contours from all possible viewing angles. Only the first 5 Eigenshapes are used for modeling variations within shape space. The deformation of newly created shapes additionally restricted in terms of rotation and scaling as well as to the distance from the training samples in the Eigenshape space (only contours close to the training set are valid). ASM matching is implemented as an iterative procedure:

2. For each contour model point \( A_k \) (k = 1…23) of the ASM find the best matching image point \( I_k \) using a cost function \( C_z \).
3. Compute an update of ASM parameters so that the model fits best to the image points
4. Coerce ASM parameters for shape, scale and rotation to within allowed range
5. Repeat the process \( n \) times

The search for best matching image points \( I_k \) (Step (1) above) is done along the bisector of the connections between two vertices of the ASM as shown in Figure 3. The best matching image point is calculated by maximizing a cost function \( C_z(P) \) along the line as a weighted sum of (i) distance from the ASM model point \( \omega_d C_d(P) \), (ii) the gradient magnitude \( \omega_\theta C_\theta(P) \) and (iii) gradient orientation \( \omega_\phi C_\phi(P) \):

\[
C_\sum (P) = \frac{C_d(P) + C_\theta(P) + C_\phi(P)}{\omega_d + \omega_\theta + \omega_\phi}, \quad (2)
\]

\( P \) represents a point on the search line, \( \omega_d \) and \( \omega_\theta \) and \( \omega_\phi \) are weighting factors which are used to shift the relative importance of the individual costs. The local edge direction is considered to be most important for a good match – an edge with the right direction is a good support for a correct shape, even if its magnitude is low.

![Figure 3: ASM contour and associated search lines.](image)

The point \( P \) on the search line with the maximum cost value \( C_z \) is selected as the best matching image point \( I_k \). We aim at adjusting the shape parameters to move points \( A_k \) from their current locations to be as close to the new locations \( I_k \), while still satisfying the shape constraints of the model. We first align the ASM points to the image coordinates \( I_k \) with respect the rotation and scale using a least square error distance measure [4]. The shape model is forced to stay within a certain pose envelope (it may not be to small or large and must more or less stay upright). Once we have computed a fitting transformation between the two point sets there remain residual adjustments \( dX = (dX_1, dX_2, ... dX_{23}) \) which can only be satisfied by deforming the shape of the model itself:

A shape \( X \) which consists of vertex points \((x_{1}, y_{1}, x_{2}, y_{2}, ... x_{23}, y_{23}) \) can be approximated by \( X = X_m + Eb \), where \( X_m \) is the mean shape from the training set, \( E = (e_{1}, e_{2}, e_{3}, e_{4}, e_{5}) \) is the matrix of the first 5 eigenvectors which our implementation uses, and \( b \) are the coordinates in Eigenshape space. We first project \( dX \) into the Eigenshape space, resulting in \( db \). This is the change required to adjust the model points as closely to \( dX \) as possible such that \( X + dX \approx X_m + Eb + db \). Since \( X = X_m + Eb \) and \( E^T = E^T \) (the columns of \( E \) are mutually orthogonal and of unit length) it follows that \( dX = Edb \) and \( db = E^T dX \).

We ensure that the model only deforms into shapes consistent with the set of training shapes by enforcing that the ASM co-ordinates in the Eigenshape space lie within a maximum Euclidean distance to the closest training sample point. If this is not the case, the model projection coordinates \( b \) are rescaled in order to make the model fit within that distance. After 40 iterations, only those ASMs which fit reasonably well to the image are accepted as pedestrian shapes and used for tracking.
2.3 Trajectory Generation

Once the pedestrian candidates have been computed, their motion is analyzed using the already computed KLT tracking points which lie within their boundaries. In case that no valid KLT information is available, it is approximated with a Kalman filter. Furthermore we verify the predicted position in the new frame using co-occurrence texture features. In case that the texture at the predicted position is not similar enough to the original one (due to bad motion prediction or occlusion), the new location is dismissed and the motion estimation used instead. After at most 5 dropouts the contour is removed from the list of actively tracked shapes. The operation of the tracking algorithm can be summarized as follows:

1. Create Zhao candidates within the foreground mask.
2. Remove false positives with ASM.
3. Merge very similar shapes computed in Steps 1, 2.
4. Compute Co-occurrence features of remaining shapes
5. For each ASM shape
   a. Project the shape into the next frame using either KLT motion information or a Kalman filter
   b. In an area around the prediction from Step 5a find the most similar co-occurrence feature vector for shape. Unmatched shapes keep their predicted location.
   c. Merge the results of Step 5b with the candidates from Step 3 to avoid redundantly tracked shapes.
   d. ASM iteration at the location from step 5c with the initial shape model from the previous frame – the new shape is only allowed to differ slightly from the previous frame.
6. Link new shape locations into their respective trajectories.
7. Head candidates from Step 3 start new trajectories.
8. Remove trajectories which have been predicted more than 5 times with the Kalman filter.

Multiple shapes are usually generated by the algorithm described above and only the most similar ones are merged in Step 3. The advantage of this is that the overall tracking robustness per individual is increased, the disadvantage is that it makes counting single pedestrians much harder. Figure 4 visualizes the tracks of pedestrians in the underground station scenario. Note that many tracks are generated for each person.

2.4 People Counting

There exist trajectory clustering approaches in the literature which try to make trajectories unique [5]. Ground truth for crowded scenes generally does not reveal salient clusters, which makes automatic approaches difficult. For counting purposes, we apply instead a heuristic for a virtual gate $G$ by defining the following conditions for a video frame $n$:

1. A point $p$ of trajectory $T$ is in virtual gate $G$
2. No other point of $T$ has been in $G$ for frames $i$, for all $i < n$
3. No other trajectory points have been in $G$ for $k$ frames

If all three conditions hold, the pedestrian counter associated to $G$ is incremented. The third condition prevents multiple counts for an individual which has associated multiple trajectories. $k$ depends on the frame rate and is usually set to 5 for output data. The counting direction is based on the position of the starting point of $T$ relative to $G$.

![Figure 4. Tracks generated in a densely crowded situation. Multiple shapes are created and followed on each pedestrian.](image)

3. Experimental Results

Extended practical tests of the people tracking implementation have been undertaken in two different scenarios with vastly different environmental conditions. The setups and the resulting counting statistics are described in the next two sections.

3.1 Indoor Scenario

The indoor scenario comprises a subway platform area in front of a three-way escalator. The scene has been captured with two synchronized IP-based surveillance cameras at a frame rate of approximately 20 fps at 640 x 480 pixels. Figure 1a shows the snapshot, superimposed on the frame are trajectories, the virtual gate, and a directed counter at the top left corner. In this field of view counting is performed for the two ascending escalator...
lanes. Figure 1b shows a snapshot from Camera 2. Superimposed is a virtual gate of an annotation tool which is used to generate ground truth data.

Figure 5. Counting results for subway camera

Figure 6. Linear regression of automatic/manual counts for Camera 1 and validation of the linear model for Camera 1.

Figure 7. Counting results for subway Camera 2.

3.2 Outdoor Scenario

The outdoor scenario was setup at a junction of public transport in the city of Graz, Austria. Figure 8 shows a snapshot of the scene with trajectories and counting cate superimposed. Images have been recorded with 5 machine vision cameras at a resolution of 800 by 600 pixels with 15 Hz frame rate. Figure 9 gives the uncorrected and corrected manual and automatic counts and the result of a linear regression model. Despite the completely different environmental conditions, the counting statistics are comparable to the results in the previous section, although the variance in the scatterplots is usually higher. This is due to lower pedestrian frequency as compared to the indoor subway scenario.
Acknowledgements

This work was supported by the Austrian Ministry of Traffic, Innovation, and Technology (BMVIT) under program II Intelligent Infrastructure. We appreciate the support of Wiener Linien (Vienna, Austria) and Grazer Verkehrsbetriebe (Graz, Austria) during development and test of the prototype system. We thank Dietmar Bauer for his support in the statistical analysis.

References


4. Discussion

A combination of motion estimation, texture analysis and shape matching is sufficiently robust in order to track people even in adverse situations. The accuracy of counting based on the trajectories obviously depends on the accumulation interval: the smaller the time window for accumulation, the fewer people will pass and the observed accuracies will decrease due to the smaller number of samples. For most commercially available pedestrian counting systems claiming high counting accuracies, the data basis and counting interval for the stated figures remains unclear and can therefore not be compared with our approach. Our proposed system demonstrates that it is possible to count pedestrians i) in very crowded situations and ii) with conventional camera perspectives (no top view) with accuracies up to 98%. It is unclear whether the performance of human operators would be any better for crowded situations like those examined in our study.

Our implementation has currently a computation time of around 1 second per input video frame on a state-of-the-art PC. Future work will concentrate mainly on a speedup of the code in order to make the system close to real-time.

Acknowledgements

Figure 9. Counting results in outdoor scenario for people crossing the gate from right to left and linear regression for scatter plot of automatic/manual counts for the outdoor scenario.