Detection of Loitering Individuals in Public Transportation Areas

Nathaniel D. Bird, Osama Masoud, Nikolaos P. Papanikolopoulos and Aaron Isaacs

***Abstract*—This paper presents a vision-based method to automatically detect individuals loitering about inner-city bus stops. Using a stationary camera view of a bus stop, pedestrians are segmented and tracked throughout the scene. The system takes snapshots of individuals when a clean, nonobstructed view of a pedestrian is found. The snapshots are then used to classify the individual images into a database, using an appearance-based method. The features used to correlate individual images are based on short-term biometrics, which are changeable but stay valid for short periods of time; this system uses clothing color. A linear discriminant method is applied to the color information to enhance the differences and minimize similarities between the different individuals in the feature space. To determine if a given individual is loitering, time stamps collected with the snapshots in their corresponding database class can be used to judge how long an individual has been present. An experiment was performed using a 30-min video of a busy bus stop with six individuals loitering about it. Results show that the system successfully classifies images of all six individuals as loitering.**

***Index Terms*—Computer vision, human activities recognition, short-term biometrics, surveillance.**

I. Introduction

THIS paper describes the culmination of the work started in Gasser *et al.* [1]. Our partner in this endeavor was Metro Transit, Minneapolis, MN. Metro Transit is one of the country's largest transit systems, providing roughly 95% of the 73 million bus trips taken annually in Minneapolis and Saint Paul, MN. An average of 231,000 board Metro Transit buses each weekday.

Metro Transit operates 137 routes using a fleet of 913 buses: 70 local service routes, 51 express routes, and 16 contract service routes. Of these buses, 790 are standard 40-ft buses, 115 are articulated, "accordion" buses, 16 are small buses, and two are coach buses. All Metro Transit buses have wheelchair lifts or ramps.

The goal of the project is to create a vision-based system to monitor inner-city bus stops for suspected drug-dealing activity. This is a major problem for Metro Transit, because drug-dealing activity makes their bus stops dangerous and discourages citizens from using them. According to officials at the Metro Transit, drug dealers will typically loiter around bus stops while their customers come in on buses, purchase their material, and then leave on other buses. Thus, suspected drug-dealing behavior can be detected by monitoring a bus stop for people loitering about longer than would be necessary to catch a bus. If we were to monitor a bus stop for an extended period of time, we would notice that people either just walk by or wait for the bus for a period of time that rarely exceeds 0.5 h after which they board a bus. Again, one of the main characteristics of drug-dealing behavior is the presence of the dealer for extended periods of time without boarding any bus.

The system must then have the ability to match a new image of a pedestrian to a pedestrian that it has already seen, essentially a system that can pick someone it has seen before out of a police lineup. However, in this lineup, everyone is not necessarily facing the same direction, or in front of the same background, making automated recognition difficult.

The system is comprised of a single stationary color camera that is positioned to give a view of the entire bus stop. An example frame is shown in Fig. 1. Because it must monitor the entire scene, not much detail can be found about individual pedestrians. The video from this camera is to be processed remotely, with suspected loitering activity reported to a human operator who would otherwise not interact with the system. Using a vision system in this manner presents many difficulties including shadows, occlusions, nonrigid targets, and varying lighting conditions. Other problems lie in the classification of pedestrians seen such as discerning the volume of data required to discriminate between varying numbers of pedestrians, what

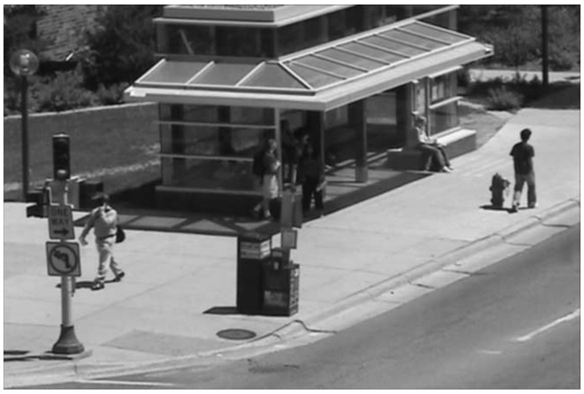
Fig. 1. Example bus stop surveillance frame.

Manuscript received September 24, 2004; revised January 27, 2005. This work was supported in part by the National Science Foundation through grant #IIS-0219863, the Minnesota Department of Transportation, and the ITS Institute at the University of Minnesota.

N. D. Bird, O. Masoud, and N. P. Papanikolopoulos are with the Artificial Intelligence, Robotics, and Vision Laboratory, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: bird@cs.umn.edu; masoud@cs.umn.edu; npapas@cs.umn.edu).

A. Isaacs is with the Metro Transit, Minneapolis, MN 55411 USA (e-mail: aaron.isaacs@metc.state.mn.us).

Digital Object Identifier 10.1109/TITS.2005.848370

features to use to differentiate individuals, and determining when a given piece of data is no longer useful for classification. The proposed methods have a wide applicability to domains such as monitoring of retail spaces, threat evaluation, etc.

The rest of the paper is organized as follows. Section II reviews literature relevant to the loitering detection system. Section III gives a brief overview of how the system functions. Section IV describes in depth the segmentation module of the system, and Section V describes the correlation module. Section VI discusses experimental results, and Section VII presents the conclusions.

II. Literature Review

A great deal of work has been presented on low-level vision tasks such as segmenting foreground objects in a scene from background objects in the scene. One of the most popular background segmentation methods is the mixture of Gaussians method presented by Stauffer and Grimson [2]. This method, suitable for fixed cameras, represents the background as a set of weighted, adaptive Gaussian functions at each pixel. If the color value recorded at the pixel does not fall within the dominating Gaussians at the current time instance, it is marked as foreground. This method is very popular and has improved upon in the literature in places such as [3] and [4].

Tracking has been used with great success in the field of computer vision. Several methods are used in practice including tracking the overlap in extracted blobs from one frame to the next. Other methods track color information within the scene such as the active contour (snakes) method [5] and the kernel tracking method [6]. Snakes are useful if the desired region is very different from the background, while the kernel tracking method has shown itself to be very robust in tracking nonrigid targets in complex backgrounds.

A great deal of research has also been invested into automatic biometric identification of individuals from vision systems. Face recognition systems, such as the eigenfaces and fisherfaces demonstrated by Belhumeur *et al.* [7], recognize individuals based upon an analysis of facial features. Other biometric identification methods require much closer monitoring of the subjects including fingerprints and retina identification. These methods require high-resolution images of the individuals to be identified in constrained orientations: one cannot recognize a face without sufficient resolution, or if the individual is turned away from a camera. Fingerprints and retina identification simply cannot be used to monitor a scene from a far-away vantage point. Other methods, such as color-based models, have been used to identify inanimate objects in constrained settings such as the method by Nayar *et al.* [8].

A variety of statistical pattern recognition procedures are used to distinguish between different classes of data points including Fisher linear discriminants (FLD), principal component analysis (PCA), support vector machines (SVM), kernel FLD, and kernel PCA. FLD and PCA are presented as linear classifiers by Fukunaga [9]. A good survey of the kernel-based methods is presented by Muller *et al.* [10].

Some work has been performed on segmenting individuals from somewhat unconstrained scenes. Zhao and Nevatia [11] present a model fitting method to identify individuals in crowds even when they partially occlude each other. The foreground tracking portion of the method presented by Elgamal *et al.* [12] also demonstrates robustness tracking people through partial occlusions.

Some work has been presented on identifying specific activities performed by humans using vision-based systems. These methods can be generally thought to belong to two categories. The method presented by Ben-Arie *et al.* [13] demonstrates an activity recognition model based on tracking the motion vectors of the subject's hands, legs, and torso. In contrast, Masoud and Papanikolopoulos [14] present a method to distinguish between several motions within a constrained environment without specifically tracking any particular body parts.

A method to detect people and analyze their behavior in an outdoor environment is presented by Haritaoglu *et al.* [15]. This method segments blobs from grayscale video of outdoor scenes and then performs silhouette analysis on the blobs to try to identify the action currently being performed by a subject in the scene.

III. System Description

The system described here can be divided into two functional components. The first component is the pedestrian segmentation module. This is the part of the system that goes through the video and extracts pictures of different pedestrians as well as rough tracking data to pass on to be classified and analyzed. The second component is the correlation module. This module extracts features from the images passed from the segmentation module and classifies them as either one of the pedestrians seen before or a completely new pedestrian.

At first glance, this high-level look at the system may seem very close to the one described in [1], and it is. The high-level system design turned out to be very good. The low-level internal workings are a different matter and are very different to overcome significant shortcomings.

IV. Segmentation Module

The purpose of the segmentation module is to process the video of a bus stop and extract images of individual pedestrians from it, along with their most probable backgrounds, time stamp, and tracking number. It then presents this information to the correlation module. This portion of the software was implemented using the framework for processing video which was developed by Osama Masoud at the University of Minnesota.

A. Background Segmentation

Background segmentation is performed on the video stream using the mixture of Gaussians background segmentation method described by Atev *et al.* [3]. This is a refinement on the original method proposed by Stauffer and Grimson [2], which has been further modified to remove divisions for a significant increase of performance on Intel processors. For this system, the background is modeled by four Gaussians in RGB color space. A correlated covariance mode is used, sorted by the ratio of standard deviations. Using this continuously updated model for the background, a distance measure in RGB color space from each pixel to the background pixel can be used to classify that pixel as foreground or background. This comparison creates a background mask that identifies every pixel in the image as either foreground or background.

The structural noise reduction algorithm described by Bevilacqua [16] is then used on the background mask to emphasize blobs corresponding to possible pedestrians and eliminate as much noise as possible. For this system, a square 7×7 cell structure was used with a fitness threshold of 20. These settings will join foreground regions that are not too far separated because a pedestrian will often split into two or more separate blobs, but still remove most noise.

B. Blob Extraction and Tracking

Blobs are then extracted from the background mask by identifying regions of connected foreground pixels. These blobs are then associated with the blobs found in the last frame. This provides tracking information for where the blobs are moving about the scene. In [1], tracking was handled using the kernel-based tracking method described by Comaniciu *et al.* [6]. The kernel tracking method was dropped in favor of blob tracking for two reasons. The first is that kernel tracker must be initialized on a target, and this would have to be done through blob extraction, making the use of a nonblob-based tracker redundant. The other reason not to use the kernel tracker is that complete target occlusion causes erroneous results instead of failure, which is difficult to recover from. Blob tracking does not have this problem with complete occlusion: the blob merely ceases to be detected and tracked.

C. Calibration

The system is calibrated using the geometric calibration method described by Masoud and Papanikolopoulos [17]. Because of the angular layout of most bus stops and the sidewalks around them, identifying the required geometric primitives in the scene should not be a problem in most cases. Calibrating the scene is important because it allows the system to compute the real world height of a pedestrian based upon the shape and position of their corresponding blob within the scene. An example of a calibrated bus stop system can be seen in Fig. 2, which shows the data fit created by the calibration system. For the calibration depicted, the gridlike separations between sidewalk slabs were measured, giving good parallel and perpendicular lines on the ground plane. Vertical lines corresponding to street lamps and the newspaper stand were also used for calibration.

Modifying an idea presented by Zhao *et al.* [18], the maxima of a potential pedestrian blob is taken to be the head point of the blob in the image, h_{i1}. This head point corresponds roughly to

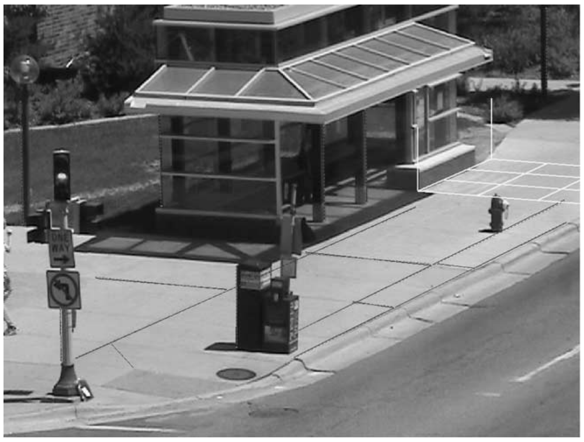

Fig. 2. Ground plane calibration of the bus stop scene using the method described in [17].

the top of the pedestrian's head. By assuming that the pedestrian is standing on the ground in the scene, subtracting the height of the blob from the y-axis component of the head point gives the foot point of the blob in the image, f_i. Using the homography matrix computed using the method in [17], the world coordinate point, f_w, of the image foot point can be found. A function, $h_{i2}(z)$, can be derived for the image coordinate location of a point some variable heights z above f_w in the world coordinate system. The point closest to the head point in the image plane is the solution to (1) for $h_{i2}(z)$. Thus, the height of the pedestrian can then be found by solving $h_{i2}(z)$ for z, which results in a long but closed form solution.

$$\frac{\partial}{\partial z} \left\| h_{i2}(z) - h_{i1} \right\|_2^2 = 0. \qquad (1)$$

D. Pedestrian Identification

Once the blobs have been identified, shape-based recognition can be performed to identify which blobs correspond to pedestrians and which do not. Because the goal is to identify loitering individuals by correlating their images, it is important to capture images that have just one pedestrian in them. The heuristics used by the system to identify pedestrian blobs are very conservative, because it is more important to get only good pictures of pedestrians than to get every possible picture of every pedestrian. If a drug dealer is loitering about a bus stop, there will be many opportunities to capture useful images of him or her, so passing up some opportunities is not a major concern.

The shape of each blob in the scene is examined at every frame to determine if it is a pedestrian. The first check of a potential pedestrian blob ensures that the blob's height is greater than its width, that the blob is not within 10 pixels of the edge of the image, and that the blob has not merged within the past five frames. If a blob passes these tests, its real world height is then computed. If the blob's real world height is within a threshold of human heights based on statistics from the National Center for Health Statistics [19], it is classified as a person. For this system, the blob's real world height must have between 60- and 80-in height to be classified as a person.

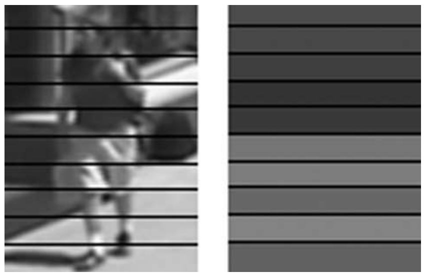
Fig. 3. Pedestrian stripe color example.

If the blob is identified as a person, a pedestrian tracking number is assigned to it unless it already has one. The pedestrian tracking number is removed if the blob merges with another blob, but propagates with the blob through tracking otherwise. The pedestrian tracking number is passed with the image of the person and the image of the most probable background to the correlation module for further processing.

V. Correlation Module

The correlation module implements an on-line classification technique that clusters incoming pedestrian images into classes comprised of images of a single pedestrian. The time stamps of the data points within these classes can then be analyzed to determine if a given pedestrian has been loitering about the scene or not. The correlation module was implemented using Matlab.

A. Features

The features analyzed by this method correspond to short-term biometrics as described in [1]. As opposed to long-term biometrics such as fingerprints, short-term biometrics can only describe a particular individual for a relatively short period of time. In this system, the short-term biometric chosen to discriminate between people is clothing color, which does not change throughout the day for most people.

The image of the pedestrian and the image of the most probable background are both converted from red–green–blue (RGB) to hue, saturation, lightness (HSL) color space. The lightness dimension of both images is then scaled by 0.5 so that slight lighting variations will not unduly skew the results. A new background mask is then created by applying a threshold to the distance between the two images at every pixel.

The image of the pedestrian is then segmented into ten equally spaced horizontal segments as shown in Fig. 3. The median HSL color of the foreground pixels of each of these ten segments is then used as the set of features defining the image of the pedestrian. Thus, the pedestrian image is distilled down to 30 data points (ten segments, three data points each). The setup of the feature vector ($\vec{v}$) is shown in (2). Note that all lightness components (L_i) are scaled to lower their importance when comparing features.

$$\vec{v} = [H_1, S_1, 0.45L_1, \ldots, H_{10}, S_{10}, 0.45L_{10}]^{\mathrm{T}}. \tag{2}$$

B. Fisher Linear Discriminants

Fisher Linear Discriminants [20] is a powerful method to reduce the dimensionality of data. It does this by finding an m-dimensional nonorthogonal basis for an n-dimensional set of data where $m < n$. The basis is found using a set of training data points by finding the basis that minimizes within-class scattering of the data while maximizing between-class scattering of the data. This emphasizes places where individuals differ greatly from each other while simultaneously deemphasizing the ways in which they are similar. Euclidean distances are used for this purpose.

Use of kernel FLDs [21] was tried briefly, but numerous problems preclude them from being applicable to this problem including the inability to find a kernel that sufficiently modeled the data and the method's inability to reduce the dimensionality of the data.

Because of the need to determine the basis prior to classification, an on-line system, such as the one described here, must be trained on a set of example data before it can be used to distinguish between classes of pedestrians. Training the system results in a transformation from the original space to the new Fisher space as well as a distance cutoff, above which two points correspond to different classes and below which two points belong to the same class. The training method used in this particular application is detailed below. It is assumed that all feature vectors $\vec{v}_{k,j}$ (jth vector in class k) have been sorted by hand into N classes. Every class k contains M_k feature vectors.

To train the FLD system, the mean feature vector for all classes and the mean feature vector for each class are needed. Equation (3) shows how to find the mean feature vector for all classes, and (4) shows how to find the mean feature vector for each individual class.

$$\vec{\mu} = \frac{\sum_{k=1}^{N}\left(\sum_{j=1}^{M_k} \vec{v}_{k,j}\right)}{\left(\sum_{k=1}^{N} M_k\right)} \tag{3}$$

$$\vec{\mu}_k = \frac{1}{M_k}\sum_{j=1}^{M_k} \vec{v}_{k,j}. \tag{4}$$

Once the mean vectors have been found, the matrices describing the between-class scatter and the within-class scatter must be found. Equation (5) shows how to compute the between-class matrix $\mathbf{S}_{\mathrm{B}}$. Equation (6) shows how to compute the within-class scatter matrix $\mathbf{S}_{\mathrm{W}}$. Because the vectors used in this method contain 30 elements, both of these matrices have 30×30 dimensionality.

$$\mathbf{S}_{\mathrm{B}} = \frac{1}{N}\sum_{k=1}^{N}(\vec{\mu}_k - \vec{\mu}) \cdot (\vec{\mu}_k - \vec{\mu})^{\mathrm{T}} \tag{5}$$

$$\mathbf{S}_{\mathrm{W}} = \sum_{k=1}^{N}\left[\sum_{j=1}^{M_k}(\vec{v}_{k,j} - \vec{\mu}_k) \cdot (\vec{v}_{k,j} - \vec{\mu}_k)^{\mathrm{T}}\right]. \tag{6}$$

The transformation matrix $\mathbf{D}$ to convert from the original space to the Fisher space is found by solving the eigenvector problem involving $\mathbf{S}_\mathrm{B}$ and $\mathbf{S}_\mathrm{W}$ as shown in (7), where $\mathbf{D}$ is the matrix of eigenvectors and $\mathbf{V}$ is the diagonal matrix of corresponding eigenvalues, sorted in descending order.

$$\mathbf{S}_\mathrm{B} \cdot \mathbf{D} = \mathbf{S}_\mathrm{W} \cdot \mathbf{D} \cdot \mathbf{V}. \tag{7}$$

Once the transformation matrix $\mathbf{D}$ is found, the training is complete. The $\mathbf{D}$ matrix is then used to find the n-dimensional Fisher space vector $\vec{f}$ corresponding to some new feature vectors $\vec{v}$, as shown in (8). To compress the data into an n-dimensional space, the first n columns of $\mathbf{D}$ are used in place of $\mathbf{D}$. This will produce a vector $\vec{f}$ with n components.

$$\vec{f} = \vec{v}^\mathrm{T} \cdot \mathbf{D}. \tag{8}$$

To compare two Fisher space vectors $\vec{f_1}$ and $\vec{f_2}$, an essentially Euclidean distance is found between them, as shown in (9). This distance can then be used with a cutoff value, below which the points correspond to the same class and above which they correspond to different classes. It should be noted that since d is a distance in a warped version of the original space, units therefore cannot be accurately applied.

$$d(\vec{f_1},\vec{f_2}) = \sum_{i=1}^{n} (\vec{f}_{1,i} - \vec{f}_{2,i})^2. \tag{9}$$

To find the cutoff distance c_d that separates within-class distances from extra class distances, the distance between all Fisher space vectors and all other Fisher space vectors in the example set are found and stored in a vector $\vec{c}$. Two other vectors are created in this process as well, namely, $\vec{p}$ and $\vec{n}$. Vector $\vec{p}$ contains the within-class distances and vector $\vec{n}$ contains the extra class distances. Thus, the cutoff distance c_d is found by solving (10).

$$c_\mathrm{d} = \max_{\text{over } i} \left\{ \frac{1}{2} \left[\frac{\sum_j (\vec{p}_j < \vec{c}_i)}{\sum_j \vec{p}_j} + \frac{\sum_k (\vec{n}_k > \vec{c}_i)}{\sum_k \vec{n}_k} \right] \right\}. \tag{10}$$

C. Correlation

In the course of this research, no reliable way was discovered to compress all of the data from multiple views of a single pedestrian down to a single high-dimensional point. For that reason, a democratic matching process is used to correlate pedestrians.

The fact that pedestrian images get passed to the correlation module with their pedestrian tracking number is beneficial to the system. This allows the system to buffer images with the same tracking number before any classification is performed. Once the buffer size limit is reached (typically five), the distance between the FLD point for each image in the buffer and each image in the database is computed. For each image in the buffer, the distances between it and all of the images in the database are sorted. Assuming the scores are below the matching cutoff, the matches with the ten lowest scores are searched in order: the lowest score was first used to find the first match to another pedestrian tracking number that appears twice. Matches are found twice to ensure that new pedestrians are matching an entire class, as opposed to an outlier. When that match is found, it is declared to be the match for the buffered image. If all matches are above the cutoff threshold or a match is not found twice, the buffered image does not match anything and is declared to be a new pedestrian. This process is repeated for each image in the buffer.

Once each image in the buffer is matched to a specific pedestrian tracking number in the database, these matches are sorted by the second matching score found above. The first tracking number that two buffered images were matched to is then taken to be the match. If no match is repeated, the buffer is declared to be a new pedestrian, otherwise the entire buffer is then moved to whatever class in the database it was matched to. In the case that it did not match any class in the database twice, a new class is created for the buffer contents.

D. Stripe Shifting

A variation on the above method allows for comparing stripe patterns shifted up or down by one stripe across images. This was developed to account for some incorrectly segmented pictures of pedestrians that have either part of the head or the feet cut off. First, the FLD method is trained on the middle eight stripe segments of the pedestrians in the training data, instead of the regular ten stripe segments. This creates the feature vector $\vec{v}_{\mathrm{S}_2}$ shown in (11). Using this feature vector, the transformation matrix $\mathbf{D}_S$ becomes a 24 $\times$ 24 matrix when calculated.

$$\begin{aligned} \vec{v}_{\mathrm{S}_1} &= [H_1, S_1, 0.45L_1, \ldots, H_8, S_8, 0.45L_8]^\mathrm{T} \\ \vec{v}_{\mathrm{S}_2} &= [H_2, S_2, 0.45L_2, \ldots, H_9, S_9, 0.45L_9]^\mathrm{T} \\ \vec{v}_{\mathrm{S}_3} &= [H_3, S_3, 0.45L_3, \ldots, H_{10}, S_{10}, 0.45L_{10}]^\mathrm{T}. \end{aligned} \tag{11}$$

When comparing actual data, three Fisher space vectors are created for each pedestrian using (8): $\vec{f_1}$ corresponding to feature vector $\vec{v}_{\mathrm{S}_1}$, $\vec{f_2}$ corresponding to feature vector $\vec{v}_{\mathrm{S}_2}$, and $\vec{f_3}$ corresponding to feature vector $\vec{v}_{\mathrm{S}_3}$. When comparing two pedestrians, Euclidean distances between the middle FLD point of the first pedestrian and the three FLD points of the second pedestrian are found sequentially. To keep the distance measure symmetric, the distances between the middle FLD point of the second pedestrian and the three FLD points of the first pedestrian are also then found. The minimum of these six distances is then taken to be the matching distance, d_S, between the two points. This can be seen in (12).

$$\begin{aligned} d_S(\vec{f}_{11},\vec{f}_{12},\vec{f}_{13},\vec{f}_{21},\vec{f}_{22},\vec{f}_{23}) = \min\Big[& d(\vec{f}_{12},\vec{f}_{21}), d(\vec{f}_{12},\vec{f}_{22}), \\ & d(\vec{f}_{12},\vec{f}_{23}), d(\vec{f}_{22},\vec{f}_{11}), d(\vec{f}_{22},\vec{f}_{12}), d(\vec{f}_{22},\vec{f}_{13}) \Big]. \end{aligned} \tag{12}$$

E. Loitering Detection

To detect loitering from the classes within the database, the time stamps passed to the correlation module with each pedestrian instance are used. The time difference between the first recorded frame the pedestrian was detected and the last recorded frame the pedestrian was detected is checked for each class in the database. If this time difference is greater than 20 min, the class is then checked for gaps in detection greater than 10 min. If no gaps are found, the pedestrian is declared to be loitering. Thus, if the time frames a pedestrian has been seen are represented as

$$t = \{t_0, t_1, \ldots, t_n\}$$

then the pedestrian is classified as loitering if, for all t_i, $t_n - t_0 > 20$ min and $t_i - t_{i-1} < 10$ min.

VI. Results

To test the system, a 30-min video was captured of a local bus stop with six "drug dealers" loitering about, while the general public went about using the bus stop as normal. The video was captured using a digital camcorder from an elevated position in a nearby building. These individuals wandered about the bus stop randomly, even leaving the scene briefly and returning. The video was then saved at 720×480 resolution for processing. Because of the processing resources required by the background segmentation portion of the segmentation module for a video of this resolution, processing cannot be performed in real time with current hardware. The picture shown in Fig. 1 was taken from this video.

For the tests discussed here, the FLD portion of the correlation module was trained using a set of 205 images of pedestrians from a different video corresponding to 41 different pedestrians.

A. Pedestrian Segmentation

The segmentation module is reliable in detecting pedestrians within the scene. From the 30-min video, 65,490 images of pedestrians have been segmented, accounting for 2769 different tracking instances. Fig. 4 shows a screenshot of the segmentation module tracking several pedestrians. Since ground truth is difficult to find for such a large number of images, ground truth was found for a smaller set of 8644 images comprising 729 tracking instances. This set was then used to test the accuracy of the method.

A presence history diagram shows that pedestrians are being tracked at any instance of time during the life of a scene. An example 35-s presence history diagram is shown in Fig. 5. Because a bus stop is not very large and tracking only occurs when a pedestrian can be well segmented, pedestrians will not usually be tracked for more than a few seconds at a time. It is important to note that when going from a 35-s scale to a scale of 30 min, the normal short tracking instances start to look like dots although they clearly are not when viewed up close. The ground-truth presence history diagram for the

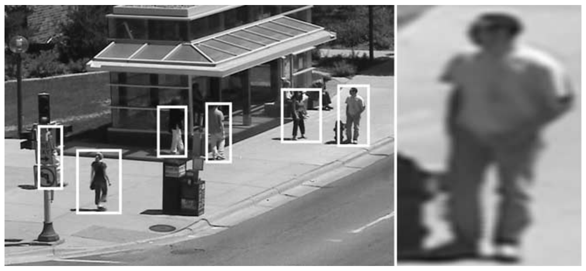
Fig. 4. Example frame of the segmentation module tracking multiple pedestrians.

tracked pedestrians in our test video can be seen in Fig. 6. Note the long lines of tracking instances corresponding to the six drug dealers that loitered about the scene.

In general, the segmentation module is very successful in capturing full-body snapshots of individuals by themselves. Fig. 7 shows some examples of well-segmented pedestrians. When problems are encountered in segmentation, the most common problem is obtaining a snapshot of a pedestrian with a portion of their head or legs cut off as shown in Fig. 8(a). Another problem that occurs rarely is obtaining a snapshot of multiple pedestrians as shown in Fig. 8(b).

B. Discussion of Matching Metric

The pedestrian classification algorithm must match a new pedestrian with the correct pedestrian stored in its database if instances of that pedestrian already exist or in a new class entirely if the pedestrian has not been seen before. The number of pedestrians stored within the database varies over time as the matching procedure is an on-line method. Accuracy can be measured by finding whether each new pedestrian is assigned to the correct pedestrian class in the database or not. But this does not mean that a 50% accuracy using this measure can be simulated with a coin flip; a 50% accuracy means that there is a 50% chance that the algorithm will match a new pedestrian to the correct class in the database when a variable number of classes are in the database. The baseline could be thought of as assigning a pedestrian randomly to a class within the database. This results in an accuracy inversely proportional to the number of pedestrians within the database and is very low once a lot of pedestrians are added. For the test discussed here, the baseline would be less than 1% at the end of the run.

Accuracy in this sense can be further analyzed; instead of just keeping track of whether a classification was correct or incorrect, how the classification was correct or incorrect can also be examined. A true positive is when the new pedestrian instance is correctly matched to a pedestrian class within the database that contains past instances of the same pedestrian. A false positive is when the new pedestrian instance is matched to a pedestrian class within the database that does not contain past instances of the same pedestrian; the pedestrian instance can represent a completely new pedestrian or already have instances of itself elsewhere in the database. A true negative occurs when the new pedestrian instance is correctly assigned to a new class in the database because there are no previous instances of the

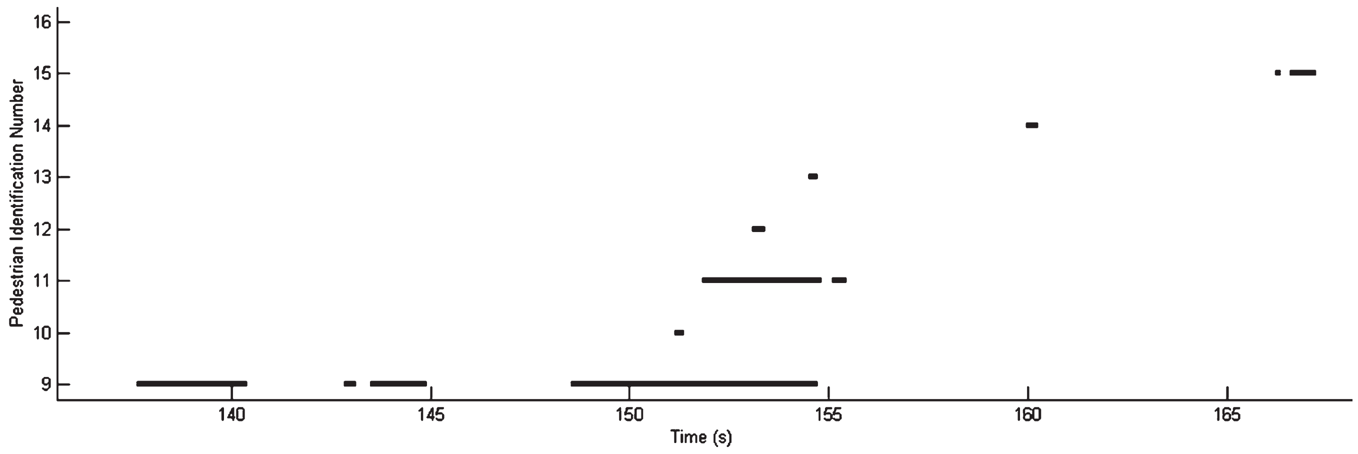

Fig. 5. Example pedestrian history diagram, demonstrating pedestrians 9 through 15 being tracked over a 35-s interval.

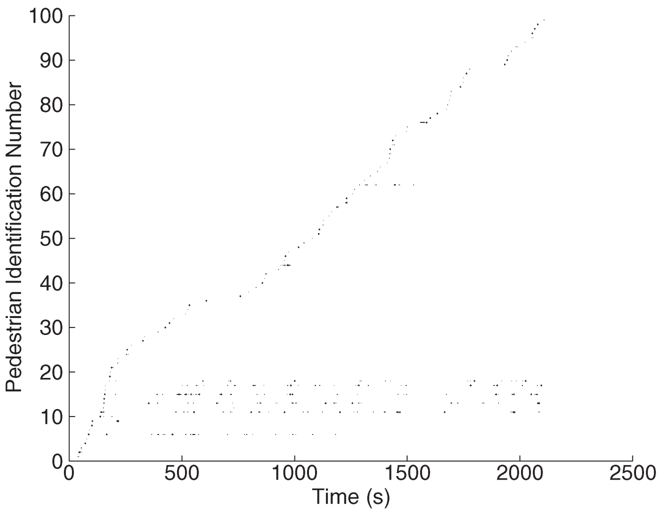

Fig. 6. Ideal case presence history diagram for the bus stop video.

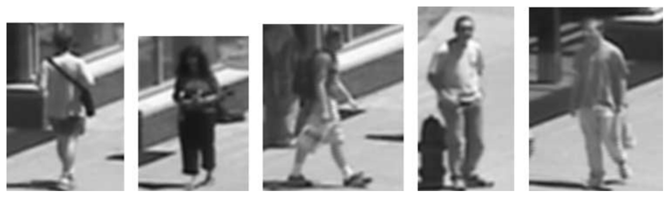

Fig. 7. Examples of well-segmented pedestrians.

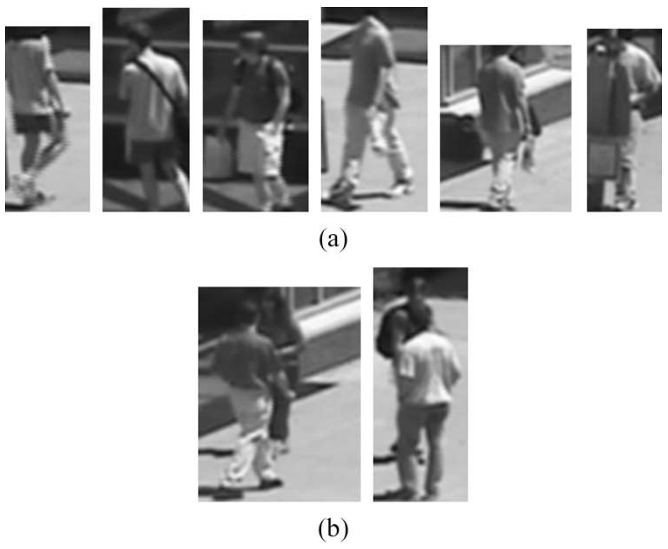

Fig. 8. Examples of poorly segmented pedestrians. (a) Snapshots of pedestrians with either their head or legs cut out of the frame. (b) Snapshots of multiple pedestrians mistakenly thought to only be a single pedestrian.

class in the database. Finally, a false negative occurs when a new pedestrian instance is assigned to a new database class while past instances of the same pedestrian exist within the database.

C. Ideal Case Results

In the ideal case, the database is maintained in a perfect state by correcting the classification algorithm when it makes an incorrect match. The reason for this is to keep the error from compounding as much as possible. Fig. 9 shows the overall accuracy of the method as the FLD space distance cutoff is varied. The reason why the cutoff is varied is because the training method tends to choose a cutoff that is far too large to use with so many pedestrians using a training set of only 41 pedestrians. The best cutoff turned out to be a distance of 500 when using a seven-dimensional Fisher space.

The time-varying accuracy of the algorithm with the best cutoff is shown in Fig. 10. There are two curves plotted here. The curve labeled continuous represents the mean accuracy of the system for the entire run, while the curve labeled windowed represents the mean accuracy of the system over the past 30 comparisons. The windowed curve shows system performance over a short period of time prior, while the continuous curve shows the overall performance.

D. Compounded Case Results

In the compounded case, the database is not corrected when the classification algorithm makes incorrect matches. Since the instances of various pedestrians can get intermixed with

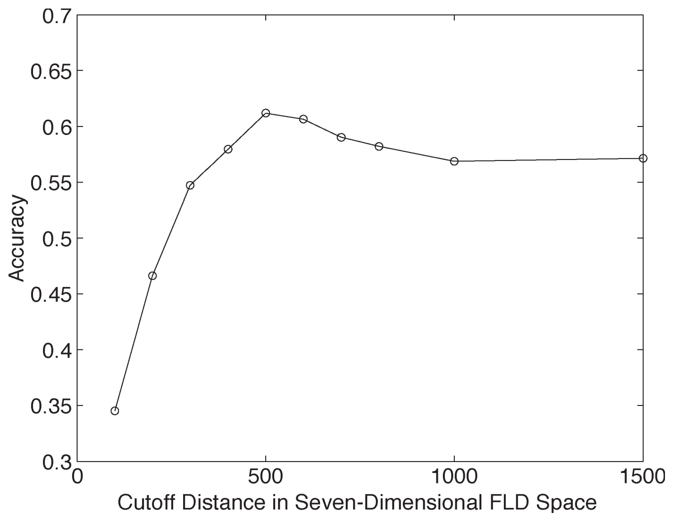

Fig. 9. Accuracy variation for the ideal error case when the regular nonstripe-shifting method is used.

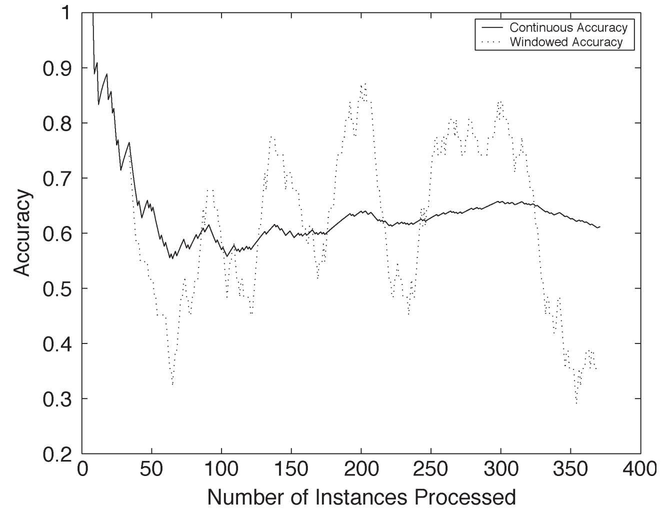

Fig. 10. Time-varying algorithm accuracy for the ideal error case when the regular nonstripe-shifting method is used and the cutoff is $c_d = 500$.

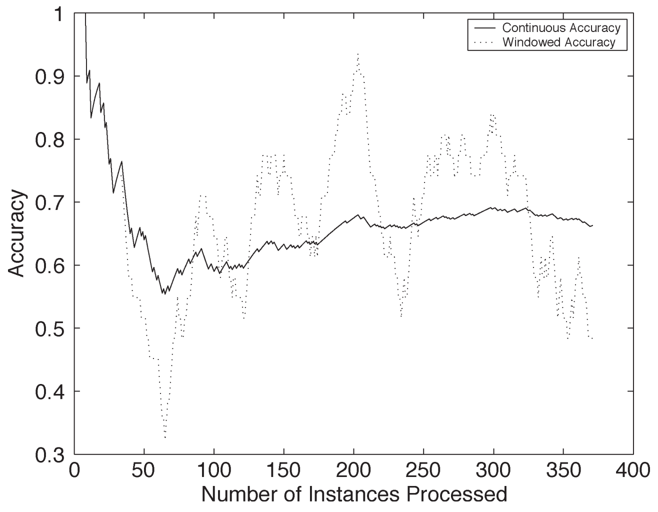

Fig. 11. Time-varying algorithm accuracy for the compounded error case when the regular nonstripe-shifting method is used and the cutoff is $c_d = 500$.

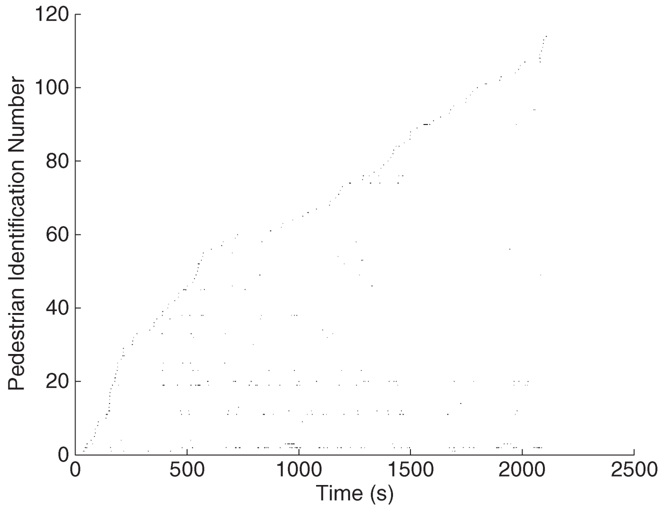

Fig. 12. Presence history diagram computed for the compounded error case when the regular, nonstripe-shifting method is used and the cutoff is $c_d = 500$.

each other, the definition of accuracy must be more closely examined. For the results presented here, a classification is considered to be a true positive if a pedestrian instance is matched to a database class that has at least one other instance of the same pedestrian, if instances of that pedestrian exist in the database already. A false positive is declared if the pedestrian instance is matched to a class that does not contain any other instances of the same pedestrian. A true negative is declared if the pedestrian instance is correctly identified as a new pedestrian. Finally, a false negative is declared when a pedestrian instance is identified as a new pedestrian while there are other instances of the same pedestrian in the database. The cutoff distance found for the ideal case is used here because otherwise a very large cutoff would be chosen that would place all pedestrians in the same class. This is because the definition of correctness in the compounded case would value placing all pedestrians together into one class.

The time-varying accuracy of the algorithm with the best cutoff is shown in Fig. 11. The curve labeled continuous represents the mean accuracy of the system for the entire run, while the curve labeled windowed represents the mean accuracy of the system over the past 30 comparisons.

Finally, Fig. 12 shows the presence history diagram for the pedestrians as they were classified by the nonstripe-shifting algorithm in the compounded error case. This can be compared with the ground-truth presence history diagram shown in Fig. 6.

E. Stripe-Shifting Results

The results when the stripe-shifting algorithm is implemented are shown for both the compounded and the ideal cases. The definitions of correctness are the same as those used previously. Fig. 13 shows the overall accuracy of the

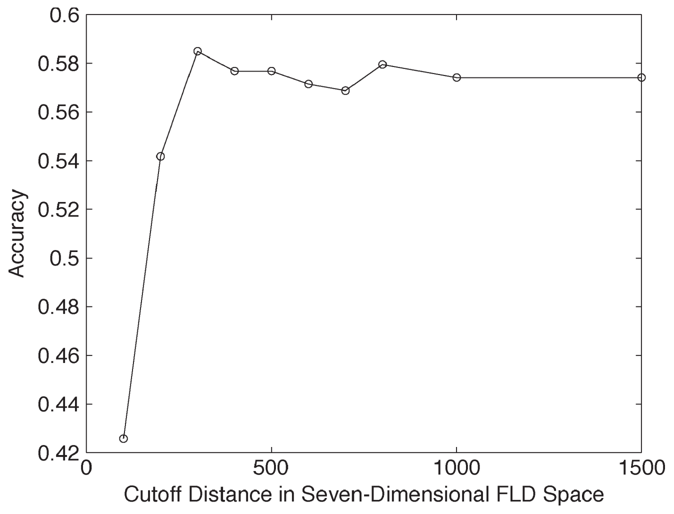

Fig. 13. Accuracy variation for the ideal error case when the stripe-shifting method is used.

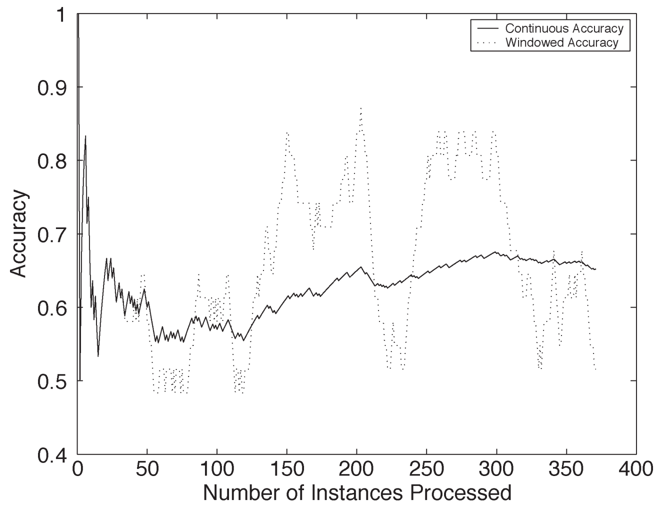

Fig. 15. Time-varying algorithm accuracy for the compounded error case when the stripe-shifting method is use and the cutoff is $c_d = 300$.

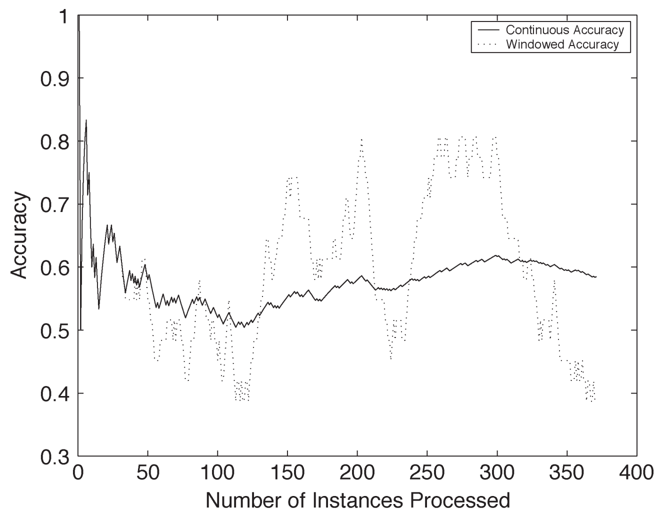

Fig. 14. Time-varying algorithm accuracy for the ideal error case when the stripe-shifting method is use and the cutoff is $c_d = 300$.

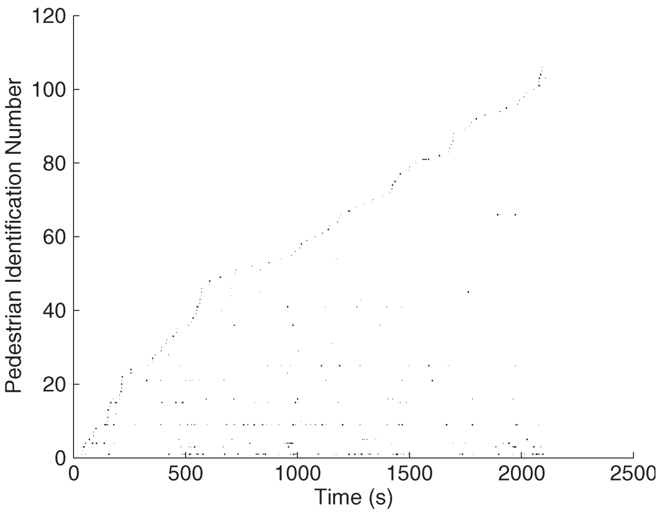

Fig. 16. Presence history diagram computed for the compounded error case when the stripe-shifting method is used and the cutoff is $c_d = 300$.

method as the FLD space distance cutoff is varied. Again, the cutoff is varied because the training method tends to choose a cutoff that is far too large. The best cutoff distance is 300 in the seven-dimensional Fisher space for the stripe-shifting method.

The time-varying accuracy of the ideal case of the stripe-shifting algorithm utilizing the best cutoff is shown in Fig. 14. The compounded case is shown in Fig. 15. In both of these plots, the curve labeled continuous represents the mean accuracy of the system for the entire run, while the curve labeled windowed represents the mean accuracy of the system over the past 30 comparisons.

Finally, Fig. 16 shows the presence history diagram for the scene computed by the stripe-shifting method. This can be compared with the ground-truth presence history diagram shown in Fig. 6.

F. Discussion of Results

The side-by-side comparison of the results from the different variations of the method can be seen in Table I. As expected, the accuracy for the compounded cases is higher than the accuracy in the corresponding ideal cases. This is because of the somewhat looser definition of correctness that must be applied to make sense of the compounded case. The compounded case is the most important to look at because it gives a good indication of how the method will perform in a real-life situation. As can be seen in Table I, the most accurate of the methods overall is the nonstripe-shifting version of the algorithm.

Using the compounded error case results, the nonstripe-shifting method to identify loitering pedestrians yields mixed results. For example, instances of three of the drug dealers were placed together as a single class in the database. That class was identified as loitering, and visual inspection of the

TABLE I
COMPARISON OF METHOD VARIATIONS

	Ideal Case		Compounded Case	
	Not Shifted	Shifted	Not Shifted	Shifted
True positives	171	166	190	191
True negatives	56	51	56	51
False positives	86	99	67	74
False Negatives	58	55	58	55
Accuracy	61.2%	58.5%	66.3%	65.2%

results make it clear that the three individuals involved were at the bus stop for a long time. One of the other drug dealers was separated into two separate classes in the database, both of which were recorded as loitering. Finally, the last two drug dealers were properly identified alone and placed in classes separately, and both were recorded as loitering. No other classes in the database, representing other passersby, were recorded as loitering.

All of the loitering classes were somewhat noisy, with 11% of the instances corresponding to nonloitering pedestrians. Despite this misclassification error percentage, every loitering pedestrian was featured prominently in the loitering classes identified by the system. The method was correctly found and was classified as loitering all of the loitering individuals. It can be claimed that all of the loitering individuals in the bus stop were found by the system, since the three that were placed together as one were all loiterers. Thus, the nonstripe-shifting method works very well at identifying loitering individuals despite some misclassifications.

The stripe-shifting method has some interesting properties worth looking at. By comparing the time-varying algorithm's correctness plots shown in Figs. 11 and 15, one can see that the stripe-shifting method limits the short-term windowed incorrectness of the algorithm. This can be seen by comparing the minima of the "windowed" curves in these two figures. While this is an interesting result, it turns out that reigning in the outliers like this does not actually help in the identification of loitering individuals, even if it does help correlate instances better.

When the results of the compounded stripe-shifting method are used to identify loitering, the results are much worse than with the nonstripe-shifting method. Four of the drug dealers and many passersby were placed together as a single class in the database and were identified as loitering. Another three classes identified as loitering were made up of an agglomeration of many pedestrians. Only one of the drug dealers was successfully classified with himself/herself using this method. Overall, this method made the classifications far noisier than they were when the nonstripe-shifting method was used. It is not nearly as accurate in the long run. The stripe-shifting method was created to allow more pedestrians to be matched to each other. Unfortunately, these results indicate that this extra matching capability serves only to increase the percentage of incorrect matches.

VII. Conclusion

The system presented in this paper discusses a single camera system that detects loitering individuals at an inner-city bus stop, but it can be used to detect loitering people in any outdoor public place. The system is divided into two main segments. The segmentation module processes the video and extracts images of individual pedestrians from it using background segmentation and blob tracking. These images, along with their most probable backgrounds, time stamp, and tracking number are then passed to the correlation module. The correlation module implements an appearance-based on-line classification technique that uses the short-term biometric of clothing color to cluster incoming pedestrian images into classes comprised of images of a single pedestrian. To determine if a given class is loitering, the time stamps associated with it are analyzed.

The results presented show that short-term biometrics, such as clothing color, can be used to match previously seen people when the use of long-term biometrics is not an option, with an approximately 66% accuracy. Despite this rather low accuracy, this matching can be successfully used to identify individuals loitering about an outdoor scene. The system presented demonstrates this ability by successfully identifying all six loitering individuals in a 30-min example video of a bus stop. Because prolonged loitering behavior in public transportation areas such as bus stops is indicative of drug dealing, the use of this system can be used to make such areas safer in the future.

Acknowledgment

We are thankful for the thoughtful comments of the reviewers. We would also like to thank Stefan Atev for the use of his background segmentation and blob tracking code. Finally, we would like to thank the drug dealers who helped by loitering about a campus bus stop for our testing video, namely, Ben, Eric, Harini, Nicola, and Stefan.

References

[1] G. Gasser, N. Bird, O. Masoud and N. Papanikolopoulos, "Human activities monitoring at bus stops," in *Proc. IEEE Int. Conf. Robotics and Automation*, New Orleans, LA, Apr. 2004, vol. 1, pp. 90–95.

[2] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Comput. Vision Pattern Recognit.*, Fort Collins, CO, Jun. 1999, vol. 2, pp. 246–252.

[3] S. Atev, O. Masoud and N. Papanikolopoulos, "Practical mixtures of Gaussians with brightness monitoring," in *Proc. IEEE Intelligent Transportation Systems Conf.*, Washington, DC, 2004, pp. 423–428.

[4] P. W. Power and J. A. Schoonees, "Understanding background mixture models for foreground segmentation," in *Proc. Image Vision Comput.*, Auckland, New Zealand, Nov. 2002, pp. 267–271.

[5] M. Kass, A. Witkin and D. Terzopoulos, "Snakes; active contour models," *Int. J. Comput. Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[6] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, pp. 564–577, May 2003.

[7] P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 711–720, Jul. 1997.

[8] S. K. Nayar, S. A. Nene and H. Murase, "Real-time 100 object recognition system," in *Proc. IEEE Int. Conf. Robotics and Automation*, Minneapolis, MN, Apr. 1996, vol. 3, pp. 2321–2325.

[9] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1972.

[10] K. R. Muller, S. Mika, G. Ratsh, K. Tsuda and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, pp. 181–201, Mar. 2001.

[11] T. Zhao and R. Nevatia, "Bayesian human segmentation in crowded situations," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, Madison, WI, Jun. 2003, vol. 2, pp. 459–466.

[12] A. Elgammal, R. Duraiswami, D. Harwood and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proc. IEEE*, vol. 90, no. 7, pp. 1151–1163, Jul. 2002.

[13] J. Ben-Arie, Z. Wang, P. Pandit and S. Rajaram, "Human activity recognition using multidimensional indexing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 1092–1104, Aug. 2002.

[14] O. Masoud and N. Papanikolopoulos, "Recognizing human activities," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, Miami, FL, Jul. 2003, pp. 157–162.

[15] I. Haritaoglu, D. Harwood and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 809–830, Aug. 2000.

[16] A. Bevilacqua, "Effective object segmentation in a traffic monitoring application," in *Proc. Indian Conf. Computer Vision, Graphics, and Image Processing*, Ahmedabad, India, Dec. 2002, pp. 125–130.

[17] O. Masoud and N. Papanikolopoulos, "Using geometric primitives to calibrate traffic scenes," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Sendai, Japan, 2004, vol. 2, pp. 1878–1883.

[18] T. Zhao, R. Nevatia and F. Lv, "Segmentation and tracking of multiple humans in complex situations," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, Kauai, HI, Dec. 2001, vol. 2, pp. 194–201.

[19] National Center for Health Statistics. Anthropometric reference data, United States, 1988–1994. [Online]. Available: http://www.cdc.gov/nchs/about/major/nhanes/Anthropometric%20Measures.htm.

[20] R. A. Fisher, "The statistical utilization of multiple measurements," *Ann. Eugen.*, vol. 8, pp. 376–386, 1938.

[21] S. Mika, G. Ratsch, J. Weston, B. Scholkopf and K.-R. Muller, "Fisher discriminant analysis with kernels," *Neural Netw. Signal Process.*, vol. IX, pp. 41–48, 1999.

Nathaniel D. Bird received his B.S. degree in computer engineering from Ohio Northern University, Ada, OH, in 2003. He is currently pursuing the Ph.D. degree in computer science at the University of Minnesota, Minneapolis, MN.

He currently works as a Graduate Research Assistant at the University of Minnesota, Minneapolis, MN. His research interests include computer vision, artificial intelligence, transportation, and homeland security applications. He has taken part in two NSF-sponsored research experiences for undergraduates programs during the summers of 2001 and 2002.

Mr. Bird is a member of the Tau Beta Pi and Phi Kappa Phi honor societies. He is a recipient of the ITS Institute Student of the Year Award, and the Matthew J. Huber Award for Excellence in Transportation Research and Education. Finally, one of his papers was a finalist for Best Vision Paper at ICRA 2004.

Osama Masoud received the B.S. and M.Sc. degrees in computer science from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, in 1992 and 1994, respectively, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, in 2000.

He is currently a Research Associate at the Department of Computer Science and Engineering at the University of Minnesota. In the past, he was a Postdoctoral Associate at the same department and served as the Director of Research and Development at Point Cloud Inc., Plymouth, MN. His research interests include computer vision, robotics, transportation applications, and computer graphics.

Mr. Masoud is the recipient of a Research Contribution Award from the University of Minnesota, the Rosemount Instrumentation Award from Rosemount Inc., and the Matt Huber Award for Excellence in Transportation Research. One of his papers (coauthored by N. P. Papanikolopoulos) was awarded the IEEE VTS 2001 Best Land Transportation Paper Award.

Nikolaos P. Papanikolopoulos (S'88–M'93–SM'01) was born in Piraeus, Greece, in 1964. He received the Diploma in engineering in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1987, the M.S.E.E. degree in electrical engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, in 1988, and the Ph.D. in electrical and computer engineering from CMU in 1992.

Currently, he is a Professor in the Department of Computer Science at the University of Minnesota, Minneapolis, MN, and the Director of the Center for Distributed Robotics. He was a McKnight Land-Grant Professor at the University of Minnesota for the period 1995–1997. He was the recipient of the Kritski Fellowship in 1986 and 1987. His research interests include robotics, sensors for transportation applications, control, and computer vision. He has authored or coauthored more than 190 journal and conference papers in the abovementioned areas (43 refereed journal papers).

Dr. Papanikolopoulos was a finalist for the Anton Philips Award for Best Student Paper in the 1991 IEEE Int. Conf. on Robotics and Automation, the recipient of the NSF Research Initiation and Early Career Development Awards and the Faculty Creativity Award during his stint at the University of Minnesota in 1995–1997, and the recipient of the Best Video Award in the 2000 IEEE Int. Conf. on Robotics and Automation. One of his papers (coauthored by O. Masoud) was awarded the IEEE VTS 2001 Best Land Transportation Paper Award. Finally, he received grants from DARPA, DHS, Sandia National Laboratories, NSF, Microsoft, INEEL, U.S. Army, U.S. Air Force, USDOT, MN/DOT, Honeywell, and 3M.

Aaron Isaacs is the Manager of Facilities Planning for Metro Transit, Minneapolis, MN. He has been employed 32 years at Metro Transit in various planning and management jobs in the areas of route and schedule planning, operations productivity analysis, facilities planning, and creation of transit advantages. The latter work resulted in the metrowide network of shoulder bus lanes and led to an exploration of ways to apply ITS to transit.