

VIDEO ANALYTICS IN RETAIL ENVIRONMENT

Gregary Prince and Pankil Butala



Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

09 May, 2012

Technical Report No. ECE-2012-01

This Technical Report does not have any distribution restrictions.

Summary

This investigation for *video analytics in a retail environment* involved: Analyzing video feeds, provided by *PrismSkyLabs*, of a business day in a creamery. The analysis involved extraction of the foreground based on motion estimation/filtering and training of a person classifier based on images acquired from Google's search engine of people and common items found in a creamery. The motion based foreground estimator is based upon binary hypothesis testing with temporal filtering to discount transient motion as being mistaken for occupancy. Occupancy detection is accomplished by analyzing accumulated motion statistics. Upon occupancy declaration a k -NN classification algorithm is applied to classify potential occupancy by people or other items. The system model is based on a feature vector containing spatial coordinates, spatial change of illumination for each region of interest, as well as the Laplacian to detect edges. Rather than apply the conventional vector metric approach, our system focuses on region covariance matrices. A consequence of this fact, is that a proper metric is required to be able to compare distance between covariance matrices of regions. We employ a metric based on generalized eigenvalues of the covariance matrix. Two methods of validation were performed, (1) involved a leave part out cross validation approach in which the training dataset is randomly split into a new training set and a new query set. These two sets are then applied to the classification algorithm; and accuracy results are obtained. The same experiment was repeated ten times (10-fold cross-validation), the results of which were between 89% and 93% successful classification rate. (2) There is also an approach to validation based on the creamery videos, in which the entire training set is applied to the creamery video frames at the user determined regions of interest. The results of the real time validation ranged from 74.5% to 98.1% for motion analysis methods and 54.3% to 63.4% for region covariance methods. The analytics collected on the regions where people are identified are mainly the occupancy time. In conclusion, the proposed system performs fairly well but may be able to be improved by incorporating different features into the system model or applying a more mature classification system like a support vector machine or neural network.

Contents

1	Introduction	1
2	Related work	2
3	Proposed approach	4
3.1	Background Modeling	5
3.2	Motion Detection	7
3.3	Motion Filtering	8
3.4	Occupancy Detection	9
3.5	Feature extraction from the Foreground	11
3.6	Training	13
3.7	Classification	14
3.8	Validation	14
4	Results and conclusions	16
4.1	Occupancy Motion Analysis Results	16
4.2	Classification Results	19
4.3	Conclusion	20
5	Appendix: MATLAB Code	20

List of Figures

1	W4 System Architecture	2
2	System Block Diagram 1	4
3	Frame of Camera View 1	4
4	Frame of Camera View 2	5
5	Example Unreliable Background Map	6
6	System Block Diagram 2	7
7	Motion Detection and Filtering	9
8	Scenario with Correct Occupancy Detection	10
9	Scenario with Incorrect Occupancy Detection	11
10	Simple Illustration of the k -NN Classifier Concept	15
11	Cashier Analytics from Camera Perspective 1	17
12	Middle Table Analytics from Camera Perspective 1	17
13	Middle Table Analytics from Camera Perspective 2	18
14	Side Table Analytics from Camera Perspective 2	18

List of Tables

1	State of the Art Approaches to Detecting and Tracking People	3
2	Occupancy Motion Detection Results	19
3	Person Classification Results	19

1 Introduction

Surveillance cameras and networks have become ubiquitous. They are installed in grocery stores, restaurants, airports, public transit stops, casinos, etc. to name a few. Their primary (if not the only) functionality is to assist security personnel in providing monitoring and security services. Now imagine if an android (lets call it Joe) was assigned to monitor the field of view instead of the camera; due to Joe's interactive nature, he/she would be capable of providing so many more services. For example, at any given time, Joe could answer the following questions:-

1. How many people are present in the field of view ?
2. What are the activities being performed by the people ?
3. Does anything look 'suspicious' ?
4. etc...

Unfortunately, technology today is not advanced enough to give us our Joe. However, we can leverage the prevalent surveillance networks to provide answers to a subset of the relevant questions.

Researches have proposed different solutions to the problem by attempting to detect and track people within a scene. Many factors must be considered such as the type of background scene to be modeled, color or grayscale modeling, the number of camera perspectives, the type of detection approach, and the tracking metrics. All of these problems introduce various degrees of complexity depending on the need. Our project directs its attention at surveillance video from a restaurant/cafeteria setting provided by 'PrismSky Labs'. Using these videos, we propose to fulfill the following tasks:

Video Analytics aims to leverage existing security camera systems, apply novel video processing algorithms to the videos and come up with interesting analysis for businesses that would help them improve their sales, operations, customer service etc. and enhance the profits.

1. Primary Task (Goal)
 - (a) Initially detect customers in a region of interest (ROI).
 - (b) Confirm a customer is in a ROI by classification.
 - (c) Record the dwell time of customers.
2. Secondary Task (Stretch Goal, which we did not accomplish in the scope of this project)
 - (a) Estimate the length of customer queue.
 - (b) Combine different points of view to improve statistics.

2 Related work

In this section, we briefly describe some of the prior approaches to monitoring people in varying environments. W^4 is a system developed by Haritaoglu et al, which uses a surveillance camera network to detect, track and monitor activities of multiple people in an outdoor environment [HHD00]. They also implement primitive object tracking. Figure 1 illustrates the system architecture, which encapsulates a lot of complex functionality with the end objective to have the ability for the system to answer the following four questions when analyzing video frames: Who, What, Where, When, hence W^4

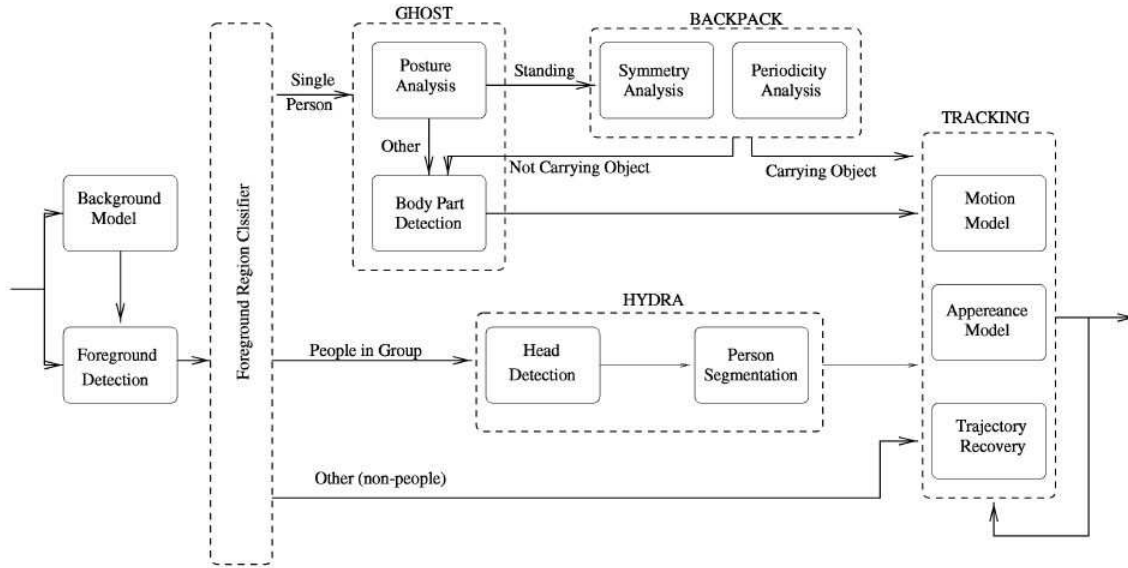


Figure 1: W4 System Architecture

This reference provides a very good survey of the approaches to detect and track people in different environments. Different works are summarized in Table 1, in which five principal qualities are discussed:

- The type of area (Indoor/Outdoor): There are implications to the video analytics approach based on whether the environment is indoor or outdoor. It is often more likely that the background model is stationary in an indoor environment as opposed to non-stationary features outdoors caused by wind and illumination changes.
- The type of sensor (Color/Grayscale): Colors have the ability to be used as discriminators in feature extraction. For example, people tend to wear clothes that are often colorful. These features can be useful in classification when illumination levels are sufficient enough to recognize them.

- The configuration of cameras within the area (Single/Stereo/Multiple): The use of the number of cameras is varied depending upon the desire to have views from multiple perspectives, using stereo to arrive at a 3D representation, or simply a static camera recording a scene from its stationary perspective.
- The detection approach (Gaussian/BiModal/Mixture) : The use of different types of kernels to detect objects in foreground is often determined by the type of setting the video is captured from. Often in an indoor environment Gaussian kernels are used.
- The tracking approach (Single/Multiple/Group): The determination of how people are to be tracked is important. Do people need to be individually tracked, or treat a group of people as an entity to be tracked is an important question to be answered.

System	Area	Sensor	Cameras	Detection	Tracking
-	Indoor(I) Outdoor(O)	Color(C) Grayscale(G)	Single(S) Stereo (O) Multiple (M)	Gaussian(S) BiModal (B) Mixture (M)	Single (S) Multiple (M) Group (G)
[WAD97]	I	C	S	S	S
[LFP98]	O	C	M	S	M
[Bou98]	O	G	S	S	M
[Grs99]	O	C	M	M	M
[OIB97]	I	G	M	S	S
[BeK99]	I	G	G	S	M
[HHD00]	O	G	S	B	M,G
[BDI96]	I	C	M	S	M
[RLW97]	I	C	S,O	S	S
[DGH98]	I	C	O	S	M
[SKB98]	I	C	M,O	S	M

Table 1: State of the Art Approaches to Detecting and Tracking People

3 Proposed approach

Our approach for a *video analytics* system is best illustrated by the system block diagram in Figure 2. The main idea being that any business can have a location engine which has a region of interest (ROI) specifier (e.g. near the cashier, seats near the windows, queue length etc.) and an analytic specifier (e.g. occupancy time, type of customer etc.) to gather important statistics that can be used in improving any aspect of their operations. The chief task is to determine if people are in fact occupying a certain ROI by analyzing available video. Two sample camera views of the videos provided are in Figures 3 and 4. Our proposed approach was to initially develop a background model that can extract the foreground.

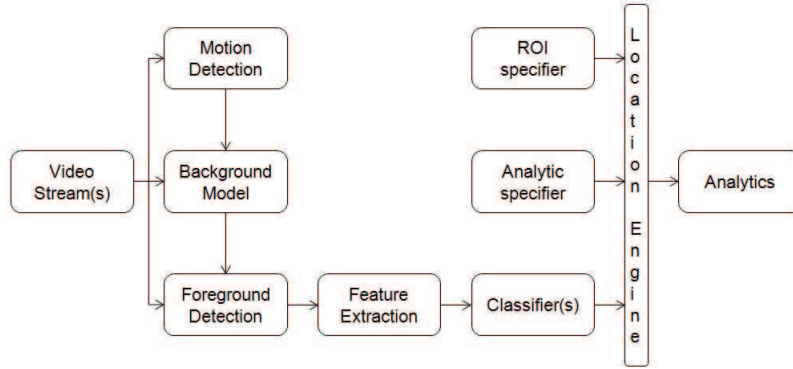


Figure 2: System Block Diagram 1



Figure 3: Frame of Camera View 1

Based on the detected foreground, we extract certain features of interest over the ROI and detect a person in the ROI. Once a person is classified in a ROI, analytics

can be gathered.



Figure 4: Frame of Camera View 2

The project sponsor, *Prism Skylabs*, provided two videos, obtained by a closed circuit television (CCTV) security camera system within a California creamery, to attempt to gather analytics on their occupants. The perspectives of each video can be observed in Figures 3 and 4. Each frame of these videos is 1280x720 pixels with three (3) channels (RGB) in a Matroska Multimedia Container (.mkv) format. The videos were originally taken at twenty five (25) frames per second over a span of one (1) business day and were then subsampled to fit the running time to approximately thirty five (35) minutes. However subsampling ratios of the videos differ, thus providing two videos that are not frame by frame synchronized. This ultimately causes frame divergence as time progresses.

The provided videos cannot be directly processed in MATLAB, as a decoder for .mkv formats is not natively supported. The VLC player was used to convert .mkv format to .mp4 (MPEG-4) which is natively supported by the *VideoReader* class in MATLAB. Both videos are independently and jointly analyzed to produce semantically meaningful occupancy data/analytics.

3.1 Background Modeling

A challenge in modeling the background lies in generating a background model despite varied amounts of foreground motion. "Seated" and "people in queue" move very differently and thus background model will take a long time to initialize. Obtaining a good initial background model would be possible from the video of the coffee house immediately upon its opening since customer foot traffic shall be at a minimum; however, we proceeded to model the background with the motion filled video we were given.



Figure 5: Example Unreliable Background Map

For the background model, the initial goal was to accomplish one of the following two tasks:

- Initialize the background model by averaging a one (1) minute run of the creamery at the start of the day when none or minimal customers would be present in the creamery thus giving a good background. Using an adaptive slow periodic update of the background can compensate for illumination changes, chair/table shifts, and even object left behind on the table (ex: flower vase) etc. Motion detection can be used to separate static from moving pixels in each frame and selectively update the background. We were unable to use this method because we could not acquire a video with the aforementioned criterion.
- In the absence of a clean one (1) minute run at the beginning of the day, the approach was to declare an undecided background frame and use motion detection to selectively update the background model with static pixels. However, since there was significant motion within the regions of interest in the creamery throughout the length of the video, this method proved futile as well.

The attempt at modeling the background is illustrated in Figure 5. There is a large amount of background ambiguity in the regions where the motion is most common. In the absence of a reliable background model, we introduced a modification into the system model. This assumption states that anything that is static is the background and anything that is moving is the foreground. The output of the motion detection block is now treated as the foreground that serves as the input to capturing analytics. The modified system block diagram is shown in Figure 6.

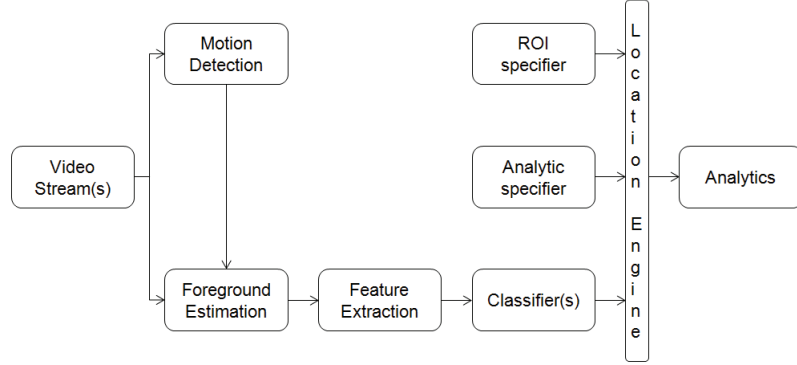


Figure 6: System Block Diagram 2

Once the foreground is detected, this video is fed to the feature extraction block. The first feature is aggregated filtered motion within the specified ROI. The other feature used is a covariance matrix. The analytic specifier could be occupancy duration, person count, queue length, etc. The location engine is a business-specific block, containing information about what the business is, where the ROIs are located, what cameras have a view of the ROI, and other details specific about the business. The system can be developed independent of any information about the business as it isolates business specific information from fundamental video processing and feature extraction.

Initially videos were converted to grayscale to run the video processing algorithms since color information is not used. The algorithm consists of the following modules :

- Motion Detection
- Motion Filtering
- Occupancy Detection
 1. Motion-based approach
 2. Covariance Matrix approach

3.2 Motion Detection

Binary hypothesis testing was used as the method for motion detection.

$$\text{Decision} = \begin{cases} M, & \text{if } P(Y(x, y)|M) > \eta \\ S, & \text{if } P(Y(x, y)|S) < \eta \end{cases} \quad (1)$$

$Y(x, y)$ is the intensity of a pixel at location (x, y) . The pixel can either be determined moving or stationary. Under hypothesis M the pixel is assumed to be moving, while hypothesis S implies the pixel is stationary. The hypothesis threshold is $\eta = \theta \frac{\sigma_M^2}{\sigma_S^2}$. The

stationary variance is $\sigma_S^2 = 10$, while the variance ratio $\frac{\sigma_M^2}{\sigma_S^2} = 5$, and $\theta = 1$. Motion detection could be improved using Markov Random Field (MRF) analysis with *Ising* potential and second order neighborhoods. However the execution time over all frames for the entire video proved exceptionally long and therefore not incorporated into the motion detection algorithm.

3.3 Motion Filtering

Motion detection provides the hypotheses of stationary or moving pixels on a frame by frame scale; however, not all motion is viewed as motion of interest. Therefore, filtering of transient motion and retaining of persistent motion is applied to capture the motion of interest (e.g. people as opposed to light reflecting into the creamery from passing cars, or screen changes on the screen of patron’s laptops, etc.). Imagine a waiter walking about a creamery. For occupancy sensing, the goal is to suppress the waiter’s motion when he/she is walking around in background. Alternately, visualize a door behind a table. People are constantly entering and leaving the creamery. For good occupancy analytics, we need to filter out their motion as well. Consequently a two step transient motion filtering is applied to aid in reducing false detections of occupancy. To visualize motion detection and temporal filtering, refer to Figure 7.

3.3.1 Motion Aggregation

The output of motion detection gives a *flag frame* where each pixel is assigned a flag indicating whether motion is detected or not. This frame contains flags corresponding to both, transient and persistent motion. In this motion aggregation step, we pixel-wise sum all flags within a temporal window of length T_{ma} . $T_{ma} = 8s$ was used for this project.

3.3.2 Motion Thresholding

This aggregated motion frame is then subject to thresholding based on how much transient motion is desired to be suppressed; a factor of 0.5 was used. This implies that a pixel will be marked with a *persistent movement map* if more than 50% of the buffered pixels in the past T_{ma} seconds indicate motion and it provides good results.

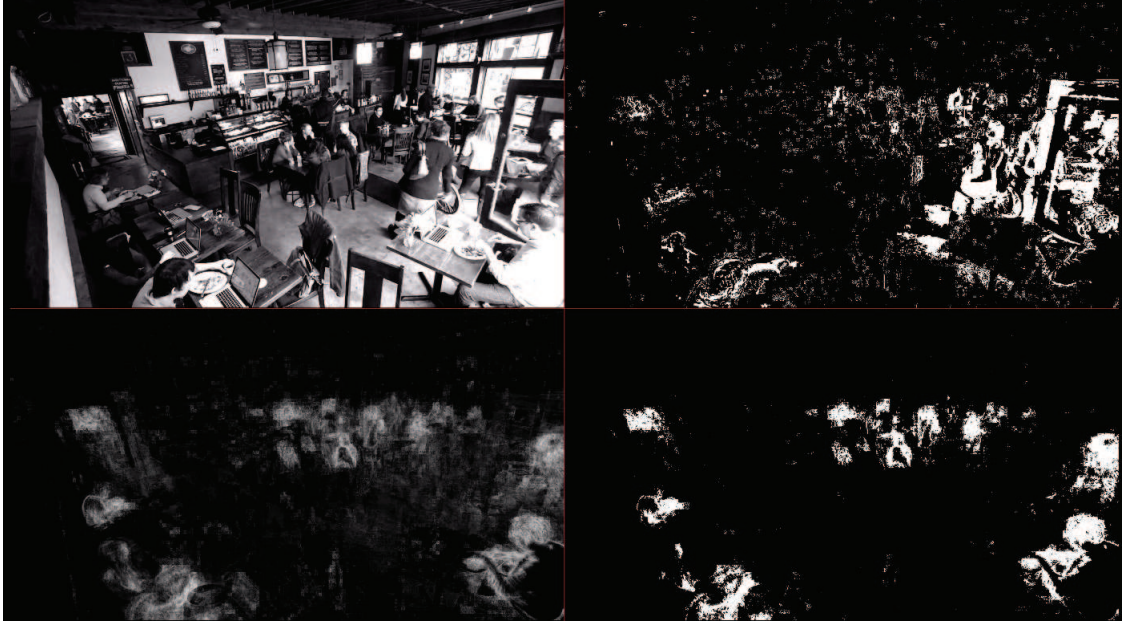


Figure 7: Motion Detection and Filtering

The image in top left quadrant shows a frame at a time instant within the creamery. The image in top right quadrant shows the binary hypothesis (equation 1) tested persistent movement map. One can see all motion being accounted for in this approach. Transient motion, for example, is due to the door closing and the two people walking close to the door. Persistent motion, for example, is due to the person at the table in the bottom of the image close to the door. The image in the bottom left quadrant shows the motion aggregated frame over the past interval T_{ma} . One can note the effect is two-fold: a lot of *ghost* images are observed due to past motion being carried over and different intensities of motion based on how transient (black) / persistent (white) they are. The image in bottom right quadrant is obtained from the aggregated motion frame by thresholding. You can clearly see that the transient motions are filtered out while persistent motions are clearly captured.

3.4 Occupancy Detection

After motion aggregation and thresholding, occupancy analytics algorithms are applied to extract information in a four-step process.

- Select an ROI: We selected two rectangular bounding boxes in each video stream. In camera view 1 we chose the middle table and the cashier. In camera view 2 we chose the side table at bottom right in field of view and again middle table.
- For each thresholded motion aggregated frame, the motion flags within the selected region of interest are summed.

- If this aggregated motion is above a certain threshold (noise level), declare the ROI to be occupied. This decision is called instantaneous frame-wise occupancy.
- The instantaneous frame-wise occupancy works well for most situations because people do produce a lot of observable motion even when they are just sitting at a table. To account for the other very short interval of times when the person moved infrequently, a temporal vote based scheme was introduced to finalize the occupancy decision. Over a preset time window, instantaneous frame-wise occupancy votes are aggregated and declaration of occupation occurs if the count is above a preset minimum value. This approach yields a smooth and more accurate occupancy decision metric.

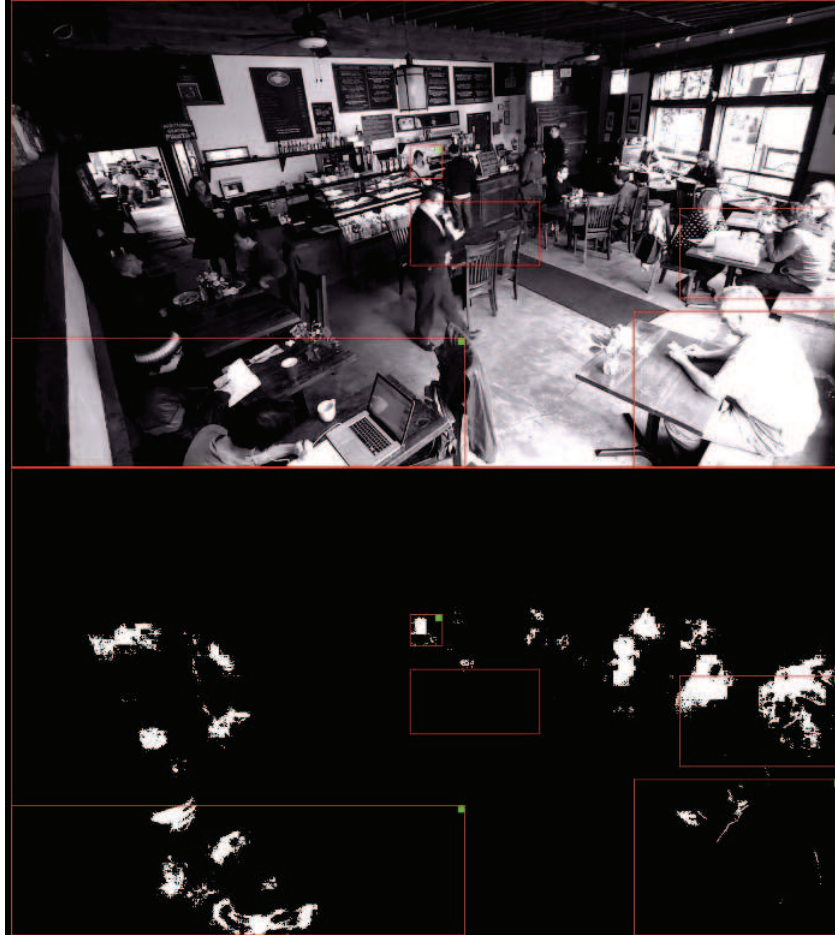


Figure 8: Scenario with Correct Occupancy Detection

Figure 8 provides an example of a correct occupancy detection in the ROI's. The ROI's are indicated by red rectangles. The little green squares in the top right corner of the ROI illustrates occupancy, whereas the absence of one illustrates the ROI being unoccupied. For the middle table, the method works quite well in classifying

the person walking in the ROI as transient motion. The transient motion is filtered out from the aggregated frame-wise occupancy decision and the system declares the middle table unoccupied.

Figure 9 shows an example of incorrect occupancy detection for the middle table. As the queue forms behind the table, the slow persistent motion is captured within the ROI and a declaration of being occupied occurs (denoted by the small green square in the upper righthand corner of the ROI). There is an opportunity to improve this behavior by combining the results of both cameras if the cameras are in fact synchronized.



Figure 9: Scenario with Incorrect Occupancy Detection

3.5 Feature extraction from the Foreground

In [TPM06], the use of covariance matrices as region descriptors is proposed. The covariance matrix provides a way of fusing multiple features that might be correlated. For each pixel inside a rectangular image ROI R , the k^{th} sample feature of the j^{th} training set is given by the length M vector $\xi_{j,k}$. We can represent the region R with

the $M \times M$ covariance matrix of feature points for the j^{th} training set as \mathbf{C}_j , with its associated mean vector μ_{ξ_j} :

$$\mathbf{C}_j = \frac{1}{n-1} \sum_{k=1}^n (\xi_{j,k} - \mu_{\xi_j})(\xi_{j,k} - \mu_{\xi_j})^T \quad (2)$$

$$\mu_j = \frac{1}{n} \sum_{k=1}^n \xi_{j,k} \quad (3)$$

The covariance matrix of feature points can incorporate many features of images, such as various color representations, position, luminance, etc. The chief idea is that given the features of interest over a certain region of the image can be compactly represented and the covariance of the region of interest can be compared to another region's covariance. The similarity of the two, in loose terms, can be heuristically thought of as a *match* between the two sub-images. Therefore, if each sub-image has similar covariance matrices, one may conclude that a person is present in the ROI or not. One of the main issues with this approach, however, is the fact that covariance matrices are, by definition, positive semi-definite quantities; this property hinders the ability to define a *metric* between any two covariance matrices. This metric is required if we hope to determine the *similarity* between two region covariance matrices.

A metric d is valid if and only if it meets the 3 (three) properties of all distance metrics:

- $d(\mathbf{C}_i, \mathbf{C}_j) \geq 0 \ \forall i, j \in R$, and $d(\mathbf{C}_i, \mathbf{C}_j) = 0$ only if $(\mathbf{C}_i = \mathbf{C}_j)$
- $d(\mathbf{C}_i, \mathbf{C}_j) = d(\mathbf{C}_j, \mathbf{C}_i) \ \forall i, j \in R$
- $d(\mathbf{C}_i, \mathbf{C}_j) + d(\mathbf{C}_i, \mathbf{C}_k) \geq d(\mathbf{C}_j, \mathbf{C}_k) \ \forall i, j, k \in R$

In [FoM99], the following metric, which exploits the concept of generalized eigenvalues and eigenvectors, for covariance matrices is proposed:

$$d(\mathbf{C}_i, \mathbf{C}_j) = \sqrt{\sum_{m=1}^n \ln^2 \lambda_m(\mathbf{C}_i, \mathbf{C}_j)} \quad (4)$$

where \mathbf{C}_i and \mathbf{C}_j are two covariance matrices to be compared, in our context from the i^{th} acquired image and j^{th} trained image. From an information theoretic point of view, the information of a Gaussian random variable scales with the natural logarithm of the variance $\ln \sigma^2$. In [FoM99], their assumption is that for non-Gaussian sources, $d^2 = \sum_m \ln^2 \lambda_m$. The $\lambda_m(\mathbf{C}_i, \mathbf{C}_j)$ are the m^{th} sets' generalized eigenvalues computed between the covariance matrices, which are found by solving the following eigenvalue problem, with eigenvectors \mathbf{x}_i :

$$\lambda_m \mathbf{C}_i \mathbf{x}_m = \mathbf{C}_j \mathbf{x}_m \quad (5)$$

Lastly, to obtain d , the $\sqrt{(\cdot)}$ operator is applied to the squared metric. One idea is to define the feature vector as follows:

$$\xi = \left(x, y, \frac{\partial Y(x, y)}{\partial x}, \frac{\partial Y(x, y)}{\partial y}, \frac{\partial^2 Y(x, y)}{\partial x^2} + \frac{\partial^2 Y(x, y)}{\partial y^2} \right)^T \quad (6)$$

The feature variables x, y are the coordinates of the pixel within the region, R ; The video is transformed into the YC_bC_r color space, and the luminance Y is retained while the color chromas are discarded. The first order derivatives of the image luminance with respect to x and y , $\frac{\partial Y}{\partial x}$ and $\frac{\partial Y}{\partial y}$ are computed. as well as the Laplacian of the luminance $\frac{\partial^2 Y(x, y)}{\partial x^2} + \frac{\partial^2 Y(x, y)}{\partial y^2}$.

The motivation behind this attribute selection is to obtain a covariance matrix that can efficiently discriminate between different persons in the field of view. We assume that provided the specified ROI and the level of motion estimated within it that the location x, y is a logical choice to have in the feature vector. Additionally, the first order derivatives provide information about the edges in the image. When a person is seated, we expect to be able to detect edges to be present, as opposed to smoother, or even the absence of, edges when the space is vacant. Lastly, the Laplacian is used to provide edge detection.

3.6 Training

Having implemented the defined feature vector ξ , distance metric based on generalized eigenvalues, $d(C_i, C_j)$ and a k -nearest neighbor k -NN classifier, the next task is to efficiently gather training and query data to provide the classifier with enough general knowledge to make it effective.

Our training approach involved acquiring images of commonly found items in a creamery, such as people (men, women, boys, and girls), laptops, cups, mugs, and even newspapers. This approach provides the classifier with not only a diverse perspective of table top items, but also a wide variety of people of varying sizes and features. Furthermore, it should be noted that many images were acquired from different angles to provide the classifier with many perspectives of what a *person* looks like in its feature space.

In total, $m = 368$ images of common creamery items were retrieved from Google's search engine, with the total number of people being 246 (99 men, 64 women, 37 boys, 46 girls), 38 cups, 17 mugs, 41 laptops, and 26 newspapers. In this phase all m images are identified and loaded into the training script and the user applies a binary label to each ROI as person or non-person. For each of these m images the training process was performed manually; each image has its covariance matrix evaluated and its binary label.

Herein some of the concerns regarding the information used to train the classifier are documented, along with the rationale to proceed with including them.

- Firstly, the aim of the training set attempts to have silhouettes of people in the sitting and standing position. It was difficult to find many silhouette images with sufficient variability for the dataset. The consequence being that we may be adding the slightly different silhouettes to the dataset and accumulating some sort of bias in the classifier. It was chosen to include all of these image acquisitions in the dataset to account for potential slight posture variations or so we hypothesized.
- Secondly, approximately a third of all acquired images are, in fact, not of people, but could have been more rich in terms of the variations. One thought was to find images of people with newspapers or sitting in front of laptops, but the option was not investigated, and we went with individual non-person objects such as mugs, laptops, newspapers etc. to train the classifier.

3.7 Classification

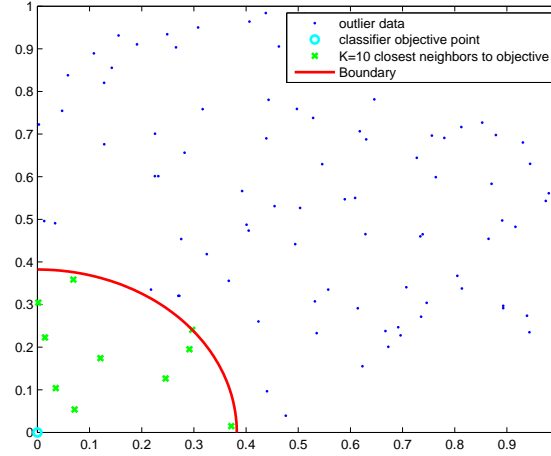
Once the training phase is complete, the algorithm proceeds to attempt to classify new features against the training set. The task of classification involves comparing a collection of training datasets (with stored *features* and *labels*) to a newly encountered dataset (with its own *features*) and assign the new dataset a classification label. In this context, as mentioned previously, the labels are binary (e.g. person or non-person). The algorithm chosen to perform the classification in this investigation is the k Nearest Neighbors (k -NN) algorithm. k is a user specified constant which determines how many training samples are required to be closest, in terms of a defined metric, to the new feature for querying. Often times metrics such as *Hamming* and *Euclidean* are applied to feature vectors, but for this investigation the aforementioned covariance metric is employed to determine the k nearest neighbors. One problem that often occurs is training phase biased towards one decision; a simple fix to this issue is to apply weights to the classifier. Furthermore it is a common practice to have k be an odd integer, as this avoids ties, simplifying the problem to a simple majority vote among the k closest samples. In our experiments, we set $k = 3$.

A simple example of a k -NN classifier output is depicted in Figure 10, where $k = 10$. This example is trying to minimize the metric (Euclidean in this case) between the point of interest (the origin) with the observations within the space. As one may see, the 10 closest points to the origin are returned (interior of red curve) and the remaining data points are discarded (exterior of red curve).

This classification task is performed on each new dataset in the covariance matrix feature space. Nonetheless, the simple idea of selecting the k closest neighbors, per the covariance metric, to the dataset of interest still holds.

3.8 Validation

There are two (2) validation approaches performed in this investigation, one in which cross validation is performed using leave-part-out-cross-validation (LPOCV) and one

Figure 10: Simple Illustration of the k -NN Classifier Concept

involving real-time querying with entirely new (unseen) frames of video of the provided creamery videos. The performance of the classifier will be assessed in terms of predictive accuracy (percentage of correctly classified samples). It should be noted that with respect to generating ground truth for the creamery video frames, it's left to human inspection of the frame sequence (as we do not know precisely how many people, cups, etc. are there or not).

For cross-validation, LPOCV is repeated exactly 10 times as recommended in [RTL09] using the entire training dataset since it is sufficiently large. Each time, the dataset is broken into two groups: 90% for training, and 10% for querying. This is done randomly (native MATLAB function *crossvalind* provides the random partitions), thereby preserving the proportions of occupied/vacant spaces in both of the new datasets. This provides mean measure of the accuracy. Note that each sample from the original dataset is used both for training and querying, but never at the same time. Most of these measures fell between 89 and 93% classification success, with a mean accuracy of 90.7%. A set of Matlab commands is provided to illustrate this LPOCV operation.

```
[trainset, queryset] = crossvalind('Resubstitution', N, [P,Q])
% split dataset into two groups, Train and Query
% can also use, depending on the type of validation,
% 'HoldOut', 'LeaveMOut', 'Kfold', etc. Method types
[outputclass] = knnclassify2(trainset, queryset, k)
%provides the output classification between the two datasets
groundtruth = cell2mat(queryset(:,2));
% create ground truth matrix from the queryset
cm = confusionmat(obtainedclasses, groundtruth)
% inspect the results compute the confusion matrix
```

```
acc = sum(diag(cm)) / sum(cm(:));
%accuracy is the sum of the diagonal against the total
%number of query elements.
```

The second approach to validation is to have the now trained classifier operate on ROI's that the motion analysis routine has identified to confirm that a person is in that ROI. We take each frame of the video over that ROI and classify the image against the dictionary to determine person classification.

4 Results and conclusions

4.1 Occupancy Motion Analysis Results

The four figures below illustrate plots from the occupancy detection algorithm for the following :

1. Cumulative motion flags within region of interest vs time
2. Instantaneous/Filtered occupancy decision vs time
3. Occupancy detection count vs time (y-axis label on the plots is a misnomer)
4. Total time for which region of interest has been occupied vs time
5. Normalized occupancy vs time. This gives the fraction of time over total time for which the given region of interest has been occupied.

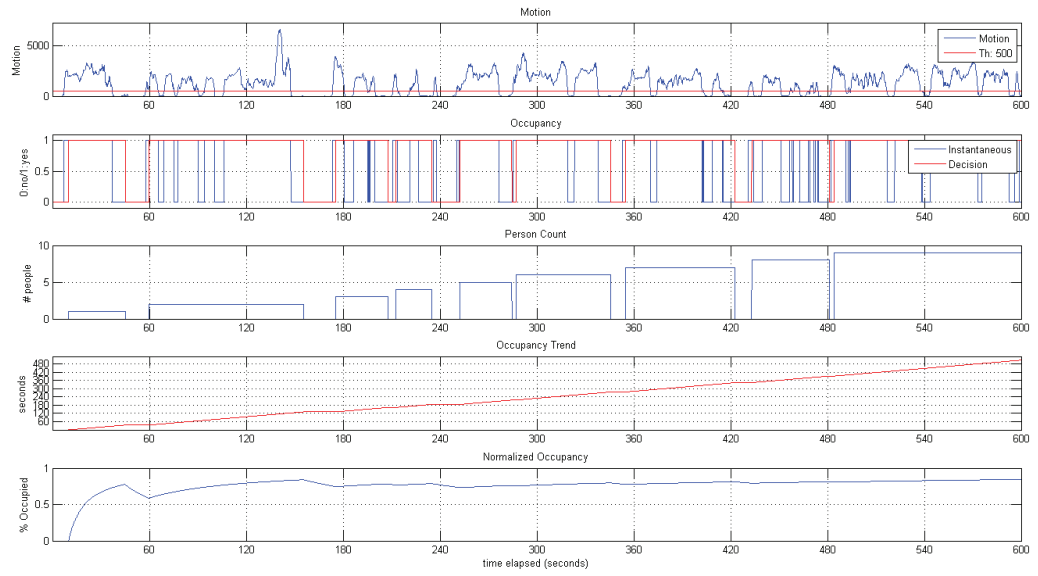


Figure 11: Cashier Analytics from Camera Perspective 1

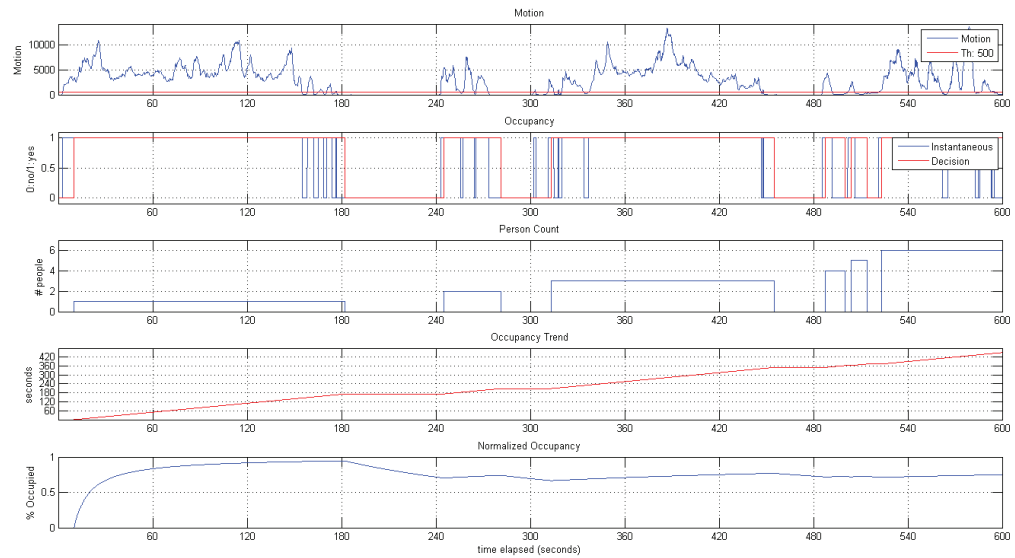


Figure 12: Middle Table Analytics from Camera Perspective 1

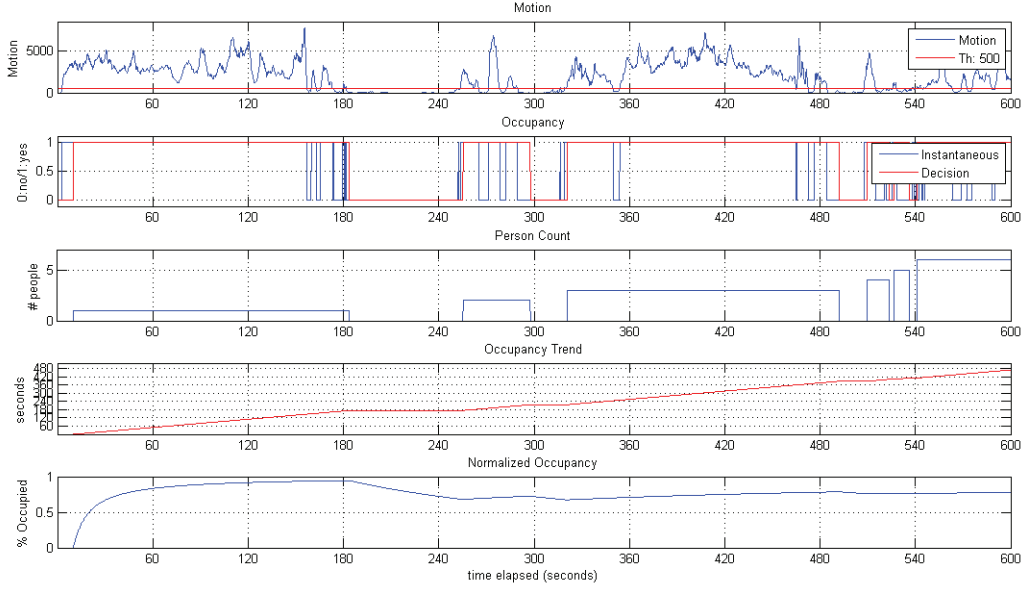


Figure 13: Middle Table Analytics from Camera Perspective 2

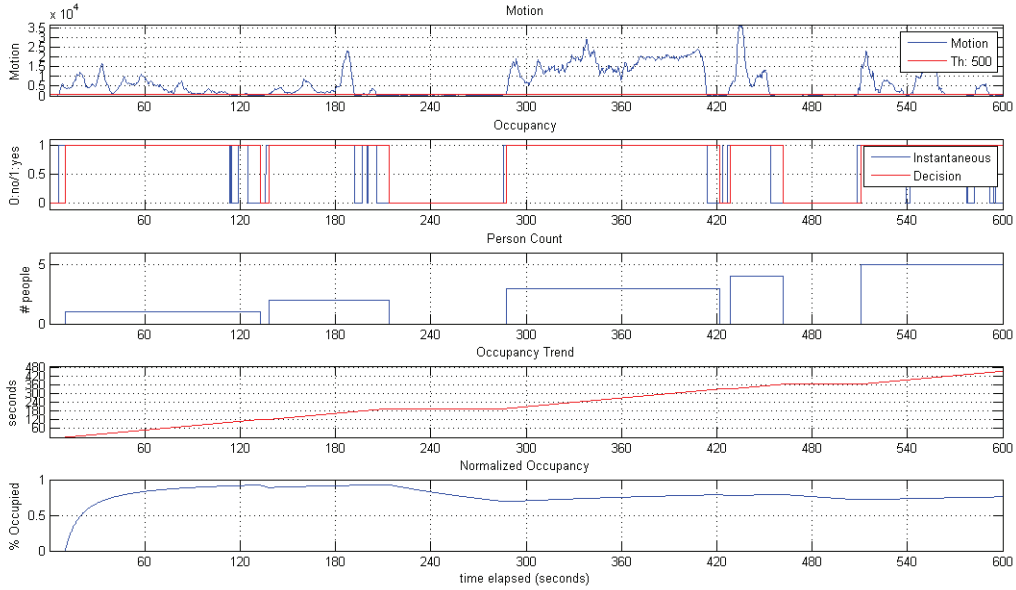


Figure 14: Side Table Analytics from Camera Perspective 2

Table 2 illustrates the performance of the motion based occupancy detection using the four ROI inspected in Figures 11 through 14. The *ground truth* is compiled by human inspection and is subject to some error. The video is inspected second by

second over a ten minute interval and we visually determine how much time the ROI is occupied by people. The detection row illustrates the time the algorithm predicts there is a person in that ROI based on the level of aggregated motion that is observed. The middle table from camera perspective 1 and side table from camera perspective 2 provide the closest detection match to ground truth, while the cashier from camera perspective 1 and the middle table from camera perspective 2 do not yield as desirable performance. The decrease in performance can be attributed to large concentration of occlusions in these areas, which lead to false detections.

Table 2: Occupancy Motion Detection Results

Occupancy	Cashier(1)	Middle Table(1)	Side Table(2)	Middle Table(2)
Ground Truth	377 s	478 s	460 s	402 s
Detection	513 s	469 s	470 s	485 s

4.2 Classification Results

The k -NN classification algorithm operating on region covariance matrices yields decent performance under the environments tested. As discussed in the validation section, the performance was gauged using both a LPOCV method using acquired images and provided results in the range of 89% to 93% for person detection. After training the classifier with the acquired images, the classifier operated on various ROI in the video streams from the camera perspectives as illustrated in Figures 2 and 3. Table 3 illustrates the average success percentage and the standard deviation of proper classification of people in the ROI.

Table 3: Person Classification Results

Occupancy	Cashier(1)	Middle Table(1)	Side Table(2)	Middle Table(2)
% Success Avg.	58.2%	63.4%	54.3%	62.1%
% Std Dev	14.8%	12.6%	15.6%	11.9%

There are many avenues one may explore in an attempt to improve its performance such as enriching the dictionary used to train the classifier with images not only of isolated objects and isolated people, but people accompanied with common creamery items. Furthermore, the application of a more robust classification algorithm e.g. Support Vector Machine (SVM) or Neural Network (NN), could potentially improve the success rate of accurately classifying people in the ROI. Moreover, modifying the feature vector to include more descriptive features relevant to smooth objects or even implementing a smooth corner detection algorithm could improve upon the results obtained.

4.3 Conclusion

We proposed a system that does not use a background model, and therefore the foreground is never truly extracted. The system is heavily dependent on motion and interprets filtered motion in an ROI as a strong probability for the presence of people. Results have illustrated that this approach performs well and in the environments tested the most false detections occur in regions where occupations are uncommon. Furthermore, the motion occupancy detector provides a good scope for improvement by person detection classification. Based on the video perspectives available, classification of actual people was not very predictable beyond 50/50 odds. Given our visual inspection, the hypothesis that aggregated motion is strongly correlated to the presence of a ROI proved more reliable than the classification of people in the video based upon off-line training of the classifier using Google acquired images of common creamery items and people. Improvements to the system may be able to occur if a background model could be obtained. Obtaining a video of an empty creamery would enable a more accurate background model and then the foreground could potentially be extracted. Improvement in the classification could be achieved by modifying the training approach and the feature vectors used.

5 Appendix: MATLAB Code

```
%*****
% EC720 PROJECT: VIDEO ANALYTICS IN RETAIL ENVIRONMENT
% Spring 2012
% Pankil Butala
% Gregary Prince
%*****

close all;
clear all;
clc;

% Variables
INISEC = 5;
bN = 1024;
VAR = 10;
bFALSE = 0;
bTRANS = 0;
bTRUE = 1;
bVALID = 1;
SIGMA = 10;
SIGMA2 = 2*SIGMA;
fTAP = 5;
```



```

bNUMCHS = 1;
bCHNUM = 1;
vNUMCHS = 3;
vDIMTM = 4;

vFile = 'vid1_1gray.avi';
jvFile = 'jVid.avi';
vRdr = VideoReader(vFile);
vFrmRt = vRdr.FrameRate;
vWid = vRdr.Width;
vHgt = vRdr.Height;
vNFrm = vRdr.NumberOfFrames;
vNFrm = min([vNFrm;vFrmRt*INISEC]);

Bkgd = struct('avg',zeros(vHgt,vWid,bNUMCHS,'uint8'),...
    'min',intmax('uint8')*ones(vHgt,vWid,bNUMCHS,'uint8'),...
    'max',intmin('uint8')*ones(vHgt,vWid,bNUMCHS,'uint8'),...
    'dMax',intmin('uint8')*ones(vHgt,vWid,bNUMCHS,'uint8'),...
    'valid',bTRANS*ones(vHgt,vWid,bNUMCHS,'uint8'));

frms = zeros(vHgt,vWid,vNUMCHS,2,'uint8');
frms(:,:,1) = read(vRdr,1);
Bkgd.min = min(frms(:,:,bCHNUM,1),Bkgd.min);
Bkgd.max = max(frms(:,:,bCHNUM,1),Bkgd.max);
Bkgd.avg = frms(:,:,bCHNUM,1);
tBuf = double(Bkgd.avg);

fIdx = 2;
for t=2:1:bN
    frms(:,:,fIdx) = read(vRdr,t);
    Bkgd.min = min(frms(:,:,bCHNUM,fIdx),Bkgd.min);
    Bkgd.max = max(frms(:,:,bCHNUM,fIdx),Bkgd.max);
    tBuf = (tBuf*(t-1) + double(frms(:,:,bCHNUM,fIdx)))/t;
    Bkgd.avg = uint8(tBuf);

    dFr = abs(frms(:,:,bCHNUM,2)-frms(:,:,bCHNUM,1));
    dFr = min(dFr,SIGMA2);
    dFr(ind2sub(size(dFr),find(dFr == SIGMA2))) = intmin('uint8');
    Bkgd.dMax = max(dFr,Bkgd.dMax);

    fIdx = rem(fIdx,2) + 1;
end

```

```

figure;
subplot(2,2,1);
imshow(Bkgd.min,[]);
title(['Pixelwise minimum over ' num2str(bN) ' frames']);
subplot(2,2,2);
imshow(Bkgd.max,[]);
title(['Pixelwise maximum over ' num2str(bN) ' frames']);
subplot(2,2,3);
imshow(Bkgd.avg,[]);
title(['Pixelwise average over ' num2str(bN) ' frames']);
subplot(2,2,4);
imshow(Bkgd.dMax,[]);
title(['Pixelwise absolute maximum error for consecutive...
frames over ' num2str(bN) ' frames']);

%*****
% EC720 PROJECT: VIDEO ANALYTICS IN RETAIL ENVIRONMENT
% Spring 2012
% Pankil Butala
% Gregary Prince
%*****
close all;
clear all;
clc;

vFileBase = 'vid1_1gray';
vFile = [vFileBase '.avi'];
vRdr = VideoReader(vFile);
vFrmRt = vRdr.FrameRate;
vWid = vRdr.Width;
vHgt = vRdr.Height;
vNFrm = vRdr.NumberOfFrames;

mFileBase = ['m_' vFileBase];

% % Variables
% INISEC = 600;
% MOTSEC = 8;
% bNUMCHS = 1;
% bCHNUM = 1;
% vNUMCHS = 3;
% vDIMTM = 4;
% bN = INISEC*vFrmRt;

```

```

% Variables
INISEC = 600;
MOTSEC = 4;
bNUMCHS = 1;
bCHNUM = 1;
vNUMCHS = 3;
vDIMTM = 4;
bN = INISEC*vFrmRt;

mVARS = 10;
mVARM = 5*mVARS;
mTHETA = 1;
mDETCNTTH = 0.5;
mTHSCL = 4.0;
mREPS = 5;
mNORD = 2;

% win = struct('R',481,'C',961,'hgt',240,'wid',320);
win = struct('R',1,'C',1,'hgt',720,'wid',1280);

vNFrm = min([vNFrm;vFrmRt*INISEC]);

mFrame = zeros(2*vHgt+1,vWid,vNUMCHS,'uint8');
maFrame = struct('motBuf',zeros(win.hgt,win.wid,...
    bNUMCHS,MOTSEC*vFrmRt,'uint8'),...
    'motAcc',zeros(win.hgt,win.wid,bNUMCHS,'uint8'),...
    'motDet',zeros(win.hgt,win.wid,bNUMCHS,'uint8'),...
    'motAccTh',mDETCNTTH*MOTSEC*vFrmRt);
%
    'motAccTh',0,...
frms = zeros(win.hgt,win.wid,vNUMCHS,2);
tBuf = read(vRdr,1);
frms(:,:,,1) = double(tBuf(win.R:win.R+win.hgt-1,win.C:win.C+win.wid-1,:));

fIdx = 2;
bIdx = 1;

% frm1 = read(vRdr,1);
% img = drawRect(frm1, 481,961,240,320,[255,0,0]);
% figure;
% imshow(img,[]);

% ** FIXED THRESHOLD FILE NAME **

```

```

mFile = [mFileBase, '_FT', ...
        '_win_R', num2str(win.R), '_C', num2str(win.C), ...
        '_H', num2str(win.hgt), '_W', num2str(win.wid), ...
        '_r', num2str(INISEC), 's', ...
        '_b', num2str(MOTSEC), 's', ...
        '_s', num2str(mVARS), ...
        '_m', num2str(mVARM), ...
        '_T', num2str(mTHETA), ...
        '_det', num2str(mDETCNTTH, 2)];

% % ** MARKOV RANDOM FIELD FILE NAME **
% mFile = [mFileBase, '_mdMRF', num2str(mNORD), ...
%         '_win_R', num2str(win.R), '_C', num2str(win.C), ...
%         '_hgt', num2str(win.hgt), '_wid', num2str(win.wid), ...
%         '_runSec', num2str(INISEC), ...
%         '_bufSec', num2str(MOTSEC), ...
%         '_Vs', num2str(mVARS), ...
%         '_Vm', num2str(mVARM), ...
%         '_T', num2str(mTHETA), ...
%         '_TSC', num2str(mTHSCL), ...
%         '_reps', num2str(mREPS), ...
%         '_detPerc', num2str(mDETCNTTH, 2)];

mWrtr = VideoWriter(mFile);
mWrtr.FrameRate = vFrmRt;
open(mWrtr);
for t=2:1:vNFrm
    mFrame(1:vHgt, :, :) = read(vRdr, t);
    frms(:, :, :, fIdx) = double(mFrame(win.R:win.R+win.hgt-1, ...
    win.C:win.C+win.wid-1, :));
%     frms(:, :, :, fIdx) = double(mFrame(1:vHgt, :, :));

maFrame.motAcc = maFrame.motAcc - maFrame.motBuf(:, :, bCHNUM, bIdx);

[E Th Diff2] = mdFixTh(frms(:, :, bCHNUM, 1), frms(:, :, bCHNUM, 2), ...
    mVARS, mVARM, mTHETA);
maFrame.motBuf(:, :, bCHNUM, bIdx) = E;
%     maFrame.motBuf(:, :, bCHNUM, bIdx) = mdMrf(Diff2, E, ...
%     mVARS, mVARM, mTHETA, mTHSCL, mREPS, mNORD);
maFrame.motAcc = maFrame.motAcc + maFrame.motBuf(:, :, bCHNUM, bIdx);
%     maFrame.motAcc = sum(maFrame.motBuf(:, :, bCHNUM, :), vDIMTM);
[rs cs] = find(maFrame.motAcc > maFrame.motAccTh);
maFrame.motDet(:) = 0;

```

```

maFrame.motDet(sub2ind(size(maFrame.motDet),rs,cs)) = 1;

mFrame(vHgt+1,:,1) = 255;
mFrame(vHgt+1+win.R:vHgt+win.R+win.hgt,...
        win.C:win.C+win.wid-1,1) = 255*maFrame.motDet;
mFrame(vHgt+1+win.R:vHgt+win.R+win.hgt,...
        win.C:win.C+win.wid-1,2) = 255*maFrame.motDet;
mFrame(vHgt+1+win.R:vHgt+win.R+win.hgt,...
        win.C:win.C+win.wid-1,3) = 255*maFrame.motDet;
mFrame(1:vHgt,,:,:) = drawRect(mFrame(1:vHgt,,:,:),win.R,win.C,...
        win.hgt,win.wid,[255 0 0]);
mFrame(vHgt+2:2*vHgt+1,,:,:) = drawRect(mFrame(vHgt+2:2*vHgt+1,,:,:),...
        win.R,win.C,win.hgt,win.wid,[255 0 0]);
writeVideo(mWrtr, mFrame);

fIdx = rem(fIdx,2) + 1;
bIdx = rem(bIdx,MOTSEC*vFrmRt) + 1;
end
close(mWrtr);
% figure;
% subplot(2,2,1);
% imshow(maFrame.min,[]);
% title(['Pixelwise minimum over ' num2str(bN) ' frames']);
% subplot(2,2,2);
% imshow(maFrame.max,[]);
% title(['Pixelwise maximum over ' num2str(bN) ' frames']);
% subplot(2,2,3);
% imshow(maFrame.avg,[]);
% title(['Pixelwise average over ' num2str(bN) ' frames']);
% subplot(2,2,4);
% imshow(maFrame.dMax,[]);

%*****
% EC720 PROJECT: VIDEO ANALYTICS IN RETAIL ENVIRONMENT
% Spring 2012
% Pankil Butala
% Gregory Prince
%*****
close all;
clear all;
clc;

```

```

vFileBase = 'vid1_1gray';
vFile = [vFileBase '.avi'];
vRdr = VideoReader(vFile);
vFrmRt = vRdr.FrameRate;
vWid = vRdr.Width;
vHgt = vRdr.Height;
vNFrm = vRdr.NumberOfFrames;

mFileBase = ['m_' vFileBase];

% Variables
INISEC = 20;
MOTSEC = 2;
bNUMCHS = 1;
bCHNUM = 1;
vNUMCHS = 3;
vDIMTM = 4;
bN = INISEC*vFrmRt;

mVARS = 10;
mVARM = 5*mVARS;
mTHETA = 1;
mDETCNTTH = 0.5;
mTHSCL = 4.0;
mREPS = 5;
mNORD = 2;

vNFrm = min([vNFrm;vFrmRt*INISEC]);
mFrame = zeros(2*vHgt+1,vWid,vNUMCHS,'uint8');
maFrame = struct('motBuf',zeros(vHgt,vWid,bNUMCHS,MOTSEC*vFrmRt,'uint8'),...
                'motAcc',zeros(vHgt,vWid,bNUMCHS,'uint8'),...
                'motDet',zeros(vHgt,vWid,bNUMCHS,'uint8'),...
                'motAccTh',mDETCNTTH*MOTSEC*vFrmRt);
%
                'motAccTh',0,...
frms = zeros(vHgt,vWid,vNUMCHS,2);
frms(:,:,,1) = double(read(vRdr,1));
fIdx = 2;
bIdx = 1;

% ** FIXED THRESHOLD FILE NAME **
mFile = [mFileBase,'_mdFT','_bufSec',num2str(MOTSEC),...
        '_Vs',num2str(mVARS),...
        '_Vm',num2str(mVARM),...

```

```

        '_T', num2str(mTHETA), ...
        '_detPerc', num2str(mDETCNTTH, 2)];

% % ** MARKOV RANDOM FIELD FILE NAME **
% mFile = [mFileBase, '_mdMRF', num2str(mNORD), ...
%         '_bufSec', num2str(MOTSEC), ...
%         '_Vs', num2str(mVARS), ...
%         '_Vm', num2str(mVARM), ...
%         '_T', num2str(mTHETA), ...
%         '_TSC', num2str(mTHSCL), ...
%         '_reps', num2str(mREPS), ...
%         '_detPerc', num2str(mDETCNTTH, 2)];

mWrtr = VideoWriter(mFile);
mWrtr.FrameRate = vFrmRt;
open(mWrtr);
for t=2:1:bN
    mFrame(1:vHgt, :, :) = read(vRdr, t);
    frms(:, :, :, fIdx) = double(mFrame(1:vHgt, :, :));

    maFrame.motAcc = maFrame.motAcc - maFrame.motBuf(:, :, bCHNUM, bIdx);

    [E Th Diff2] = mdFixTh(frms(:, :, bCHNUM, 1), frms(:, :, bCHNUM, 2), ...
        mVARS, mVARM, mTHETA);

    %     maFrame.motBuf(:, :, bCHNUM, bIdx) = mdMrf(Diff2, E, ...
    %         mVARS, mVARM, mTHETA, mTHSCL, mREPS, mNORD);
    maFrame.motBuf(:, :, bCHNUM, bIdx) = E;
    maFrame.motAcc = maFrame.motAcc + maFrame.motBuf(:, :, bCHNUM, bIdx);
    %     maFrame.motAcc = sum(maFrame.motBuf(:, :, bCHNUM, :), vDIMTM);
    [rs cs] = find(maFrame.motAcc > maFrame.motAccTh);
    maFrame.motDet(:) = 0;
    maFrame.motDet(sub2ind(size(maFrame.motDet), rs, cs)) = 1;

    mFrame(vHgt+1, :, 1) = 255;
    mFrame(vHgt+2:2*vHgt+1, :, 1) = 255*maFrame.motDet;
    mFrame(vHgt+2:2*vHgt+1, :, 2) = 255*maFrame.motDet;
    mFrame(vHgt+2:2*vHgt+1, :, 3) = 255*maFrame.motDet;
    writeVideo(mWrtr, mFrame);

    fIdx = rem(fIdx, 2) + 1;
    bIdx = rem(bIdx, MOTSEC*vFrmRt) + 1;
end

```

```

close(mWrtr);
% figure;
% subplot(2,2,1);
% imshow(maFrame.min, []);
% title(['Pixelwise minimum over ' num2str(bN) ' frames']);
% subplot(2,2,2);
% imshow(maFrame.max, []);
% title(['Pixelwise maximum over ' num2str(bN) ' frames']);
% subplot(2,2,3);
% imshow(maFrame.avg, []);
% title(['Pixelwise average over ' num2str(bN) ' frames']);
% subplot(2,2,4);
% imshow(maFrame.dMax, []);

clear all;
clc;

vFile = 'm_vid3_1gray_FT_win_R1_C1_H720_W1280_r600s_b4s_s10_m50_T1_...
det0.5_5WIN_mACnt500_mW10_mFr0.2_tMot.avi';
bImg = getBkgnd(vFile, [480 510], struct('R', 521, 'C', 1, 'hgt', 200, 'wid', 700));
figure;
% imshow(bImg, []);
save('Win5Bkgd.mat');

clear all;
clc;

% vFileBase = 'vid3_1grayTest';
vFileBase = 'vid3_1gray';
vFile = [vFileBase '.avi'];
% win = struct('R', 481, 'C', 961, 'hgt', 240, 'wid', 320);
win = struct('R', 521, 'C', 1, 'hgt', 200, 'wid', 700);
% bImg = getBkgnd(vFile, [215 270], win);
% load('winBkgndWrksp.mat', 'bImg');
load('Win5Bkgd.mat', 'bImg');
bImg = uint8(bImg);
SD = 5;

vRdr = VideoReader(vFile);
vFrmRt = vRdr.FrameRate;
vWid = vRdr.Width;
vHgt = vRdr.Height;
vNFrm = vRdr.NumberOfFrames;

```



```

% Variables
INISEC = 600;
MOTSEC = 8;
bNUMCHS = 1;
bCHNUM = 1;
vNUMCHS = 3;
vDIMTM = 4;
bN = INISEC*vFrmRt;

sFileBase = ['s_' vFileBase];
sFile = [sFileBase,'_SIL',...
        '_win_R',num2str(win.R),'_C',num2str(win.C),...
        '_H',num2str(win.hgt),'_W',num2str(win.wid)];

sFrame = zeros(2*vHgt+1,vWid,vNUMCHS,'uint8');

sWrtr = VideoWriter(sFile);
sWrtr.FrameRate = vFrmRt;
open(sWrtr);

% for t=1:bN
for t=1:500
    sFrame(1:vHgt,,:) = read(vRdr,t);
    [fg flag] = getFgnd(sFrame(win.R:win.R+win.hgt-1,...
                               win.C:win.C+win.wid-1,1),bImg,SD);
    sFrame(vHgt+1,:,1) = 255;
    sFrame(vHgt+1+win.R:vHgt+win.R+win.hgt,...
           win.C:win.C+win.wid-1,1) = 255*flag;
    sFrame(vHgt+1+win.R:vHgt+win.R+win.hgt,...
           win.C:win.C+win.wid-1,2) = 255*flag;
    sFrame(vHgt+1+win.R:vHgt+win.R+win.hgt,...
           win.C:win.C+win.wid-1,3) = 255*flag;
    sFrame(1:vHgt,,:) = drawRect(sFrame(1:vHgt,,:),win.R,win.C,...
                               win.hgt,win.wid,[255 0 0]);
    sFrame(vHgt+2:2*vHgt+1,,:) = drawRect(sFrame(vHgt+2:2*vHgt+1,,:),...
                                           win.R,win.C,win.hgt,win.wid,[255 0 0]);
    writeVideo(sWrtr, sFrame);
end
close(sWrtr);

function img = drawRect(img,R,C,Hgt,Wdt,clr)

```

```

% [iHgt iWid iChnl] = size(img);
% rs = [R*ones(1,Wdt) (R+Hgt-1)*ones(1,Wdt) R:R+Hgt-1 R:R+Hgt-1];
% cs = [C:C+Wdt-1 C:C+Wdt-1 C*ones(1,Hgt) (C+Wdt-1)*ones(1,Hgt)];
%
% for ch = 1:iChnl
%     chs = ch*ones(1,2*Wdt+2*Hgt);
%     img(sub2ind([iHgt iWid iChnl],rs,cs,chs)) = clr(ch);
% % end
% img(R,C:C+Wdt-1,:) = clr;
% img(R+Hgt-1,C:C+Wdt-1,:) = clr;
% img(R:R+Hgt-1,C,:) = clr;
% img(R:R+Hgt-1,C+Wdt-1,:) = clr;

for r=R:R+Hgt-1
    img(r,C,:) = clr;
    img(r,C+Wdt-1,:) = clr;
end
for c=C:C+Wdt-1
    img(R,c,:) = clr;
    img(R+Hgt-1,c,:) = clr;
end

function bImg = getBkgnd(vFile,tRange,win)

bCHNUM = 1;

vRdr = VideoReader(vFile);
vFrmRt = vRdr.FrameRate;

fStart = tRange(1)*vFrmRt + 1;
fStop = tRange(2)*vFrmRt;
fNUM = fStop-fStart+1;

bImg = zeros(win.hgt,win.wid);

for f=fStart:fStop
    frm = double(read(vRdr,f));
    bImg = bImg + squeeze(frm(win.R:win.R+win.hgt-1,...
        win.C:win.C+win.wid-1,bCHNUM))/fNUM;
    clear frm;
end

function [iFg iFlag] = getFgnd(img,bImg,SD)

```

```

[hgt wid] = size(img);

iFlag = zeros(hgt,wid,'uint8');
iFg = img - bImg;

for r=1:1:hgt
    for c=1:1:wid
        if(iFg(r,c) > SD)
            iFlag(r,c) = 1;
        end
    end
end

end

function [fMIN fMAX dfMAX] = initBkgnd(frames,SIGMA)
% INITBKGND initialize background.
% *INPUT:*
% [m n d fframes] = initBkgnd(iVR,fStart,fStop,sigma,fTAP)
% iVR:      input VideoReader
% fStart:    starting frame idx
% fStop:     last frame idx
% SIGMA:     variance
% fTAP:      # taps for median filter
% *OUTPUT:*
% min:       pixel minimum over fStart:fStop
% max:       pixel maximum over fStart:fStop
% dmax:      pixel maximum difference below 2*SIGMA over fStart:fStop
% oVR:       output VideoReader

% iVR = VideoReader(iVidFile);
% oVidFile = 'fVid.avi';
% % STAGE 1: run a median filter.
% % DIMY = 1;
% % DIMX = 2;
% % DIMCH = 3;
NUMCH = 3;
DIMTM = 4;
fT = size(frames,DIMTM);
% vWid = iVR.Width;
% vHgt = iVR.Height;
% % vNFrm = fStop-fStart+1;
%
% oVW = VideoWriter(oVidFile);
% oVW.FrameRate = iVR.FrameRate;

```

```

%
% frames = zeros(vHgt,vWid,NUMCH,fTAP,'uint8');
% open(oVW);
% for t=fStart:fStart+fTAP-2
%     frames(:,:,:,t-fStart+1) = read(iVR,t);
%     writeVideo(oVW,frames(:,:,:,t-fStart+1));
% end
% rdFrmIdx = fTAP;
% for t=fStart+fTAP-1:1:fStop
%     frames(:,:,:,rdFrmIdx) = read(iVR,t);
%     data = squeeze(median(frames,DIMTM));
%     writeVideo(oVW,data);
%     rdFrmIdx = rdFrmIdx + 1;
%     if(rdFrmIdx > fTAP)
%         rdFrmIdx = 1;
%     end
% end
% close(oVW);
% clear frames;
% oVR = videoReader(oVidFile);

```

```

% oVidFile = VidFile;
% % STAGE 2: get model
fMIN = min(frames,[],DIMTM);
fMAX = max(frames,[],DIMTM);
frZ0 = frames(:,:,:,1:fT-1);
frZ1 = frames(:,:,:,2:fT);
dFr = abs(frZ1-frZ0);
clear frZ0;
clear frZ1;
[I J V] = find(dFr < 2*SIGMA);
D = sparse(I,J,V);
dfMAX = max(D,[],DIMTM);
clear I;
clear J;
clear V;

```

```

%acquisition of datasets
clc
clear all
close all

```

```
dataset = {}; %initialization of the training dataset

N = 500; %how many images are we going to train on?

for jj = 1:N,
cd s_vid3_1grayTest_SIL_win_R481_C961_H240_W320
fnameLoad = sprintf('%d.mat',jj);
load(fnameLoad); %returns frm uint8 variable into workspace
cd ..
im = double(frm);
X = getCovarianceMatrix(im); %this is the covariance matrix
%imshow(im,[]);
y = input('Assign a Label to this Image to Train the Classifier: ', 's');
dataset = [dataset; {X y}];
{X y}
end

save dataset.mat

function [dist] = covMatDistance(A,B)
dist = sqrt(sum(log(eig(A,B)).^2));

function [outputclass] = knnclassify2(trainset, queryset, k)

neighborIds = kNN(trainset, queryset, k);

n = size(neighborIds,1);
outputclass = zeros(n, 1);

for i=1:n
    classes = zeros(1,k);
    for j=1:k
        classes(j) = trainset{neighborIds(i,j),2};
    end

    outputclass(i) = mode(classes);

end
```

```

function [neighborIds] = kNN(trainset, queryset, k)

%for each element in the query set,
%k columns with the k nearest neighbor IDs from the training set
neighborIds = zeros(size(queryset,1),k);
neighborDistances = neighborIds;

%trainset is a cell, remember that.
numDataVectors = size(trainset,1);
numQueryVectors = size(queryset,1);

for i=1:numQueryVectors,

    vec = queryset{i};
    %dist = distances from the current vector to every vector in the
    %training set

    dist = zeros(numDataVectors,1);
    for j=1:numDataVectors
        dist(j) = covMatDistance(vec, trainset{j,1});
    end

    %dist = sum((repmat(queryMatrix(i,:),numDataVectors,1)-dataMatrix).^2,2);

    [sortval sortpos] = sort(dist,'ascend');
    neighborIds(i,:) = sortpos(1:k);
    neighborDistances(i,:) = sortval(1:k);
end

function [covMat] = getCovarianceMatrix(img)

w=size(img,2);
h=size(img,1);
n=w*h;

%YCbCr space
Y=img(:,:,1);
% Cb=img(:,:,2);
% Cr=img(:,:,3);

```

```

%first order derivatives
[FX, FY] = gradient(double(Y));

%Laplacian Operator
[L] = del2(double(Y));

%x, y coordinates variables
[x,y] = meshgrid(1:size(img,2), 1:size(img,1));

F=zeros(5,n);
k=1;
for i=1:h
    for j=1:w
        F(:,k)=[x(i,j) y(i,j) FX(i,j) FY(i,j) L(i,j)]';
        k=k+1;
    end
end

%mean vector mu
mu = sum(F,2)/n;

%compute the covariance matrix
p = zeros(5,5);
for i=1:n
    x=(F(:,i)-mu)*(F(:,i)-mu)';
    p=p+x;
end
C=p.*(1/(n-1)); %the covariance matrix

covMat = C;

```

References

- [1] [FoM99] Foerstner, W., Moonen, B., *A Metric for Covariance Matrices*, 1999.
- [2] [HHD00] Haritaoglu, I., Harwood, D., Davis, L., *W4: Real Time Surveillance of People and Their Activities*, IEEE Transactions on Pattern Analysis and Machine

Intelligence. Vol 22, No.8 August 2000

- [3] [RTL09] Refaeilzadeh, P. Tang, L., and Liu, L. *Cross Validation*. In Encyclopedia of Database Systems, Editors: M. Tamer and Ling Liu. Springer, 2009.
- [4] [TPM06] Tuzel, O., Porikli, F., Meer, P., *Region Covariance: A Fast Descriptor for Detection and Classification*, ECCV, 2006.
- [5] [WAD97] Wren C., Azarbayejani A., Darrell, T., and Pentland A. "Pffinder: Real Time Tracking of the Human Body," *IEEE Trans. Pattern Analysis and Machine Intelligence* vol 19, no 7, July 1997.
- [6] [LFP98] Lipton, A., Fujiyoshi, H., and Patil, R. "Moving Target Detection and Classification from Real Time Video," *Proc IEEE Workshop Application of Computer Vision*, 1998.
- [7] [Bou98] Boult, T. "Frame-Rate Multibody Tracking for Surveillance," *Proc. DARPA Image Understanding Workshop*, 1998.
- [8] [GrS99] Grimson, E. and Stauffer, C. "Adaptive Background Mixture Models for Real Time Tracking," *Proc. Computer Vision and Pattern Recognition Conf.*, 1999.
- [9] [OlB97] Olson, T. and Brill, F. "Moving Object Detection and Event Recognition Algorithms for Smart Cameras," *Proc. DARPA Image Understanding Workshop*, pp.159-175, 1997.
- [10] [BeK99] Beymer, D. and Konolige, K. "Real Time Tracking of Multiple People Using Stereo," *Proc. IEEE Frame Rate Workshop*, 1999.
- [11] [BDI96] Bobick, A., Davis, J., Intille, S., Baird, F., Campbell, L., Irinov, Y., Pinhanez, C., and Wilson, A., "Kidsroom: Action Recognition in an Interactive Story Environment," *Technical Report 398, MIT Perceptual Computing*, 1996.
- [12] [RLW97] Rehg, J., Loughlin, M., and Waters, K., "Vision for a Smart Kiosk," *Computer Vision and Pattern Recognition*, 1997.
- [13] [RTL09] Refaeilzadeh, P. Tang, L., and Liu, L. *Cross Validation*. In Encyclopedia of Database Systems, Editors: M. Tamer Nzs and Ling Liu. Springer, 2009.
- [14] [DGH98] Darell, T., Gordon, G., Harville, M., and Woodfill, J., "Integrated Person Tracking Using Stereo, Color, and Pattern Detection," *Computer Vision and Pattern Recognition*, 1998.
- [15] [SKB98] Shafer, A., Krumm, J., Brumitt, B., Meyers, B., Czerwinski, M., and Robbins, D. "The New EasyLiving Project at Microsoft," *Proc. DARPA/NIST Smart Spaces Workshop*, 1998.