EC720 PROJECT, FINAL REPORT: VIDEO ANALYTICS IN A RETAIL ENVIRONMENT

*Molly Crane and Laura Ross**

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

May 8, 2012

Technical Report No. ECE-2012-02

*Author is employed at MIT Lincoln Laboratory.

The Lincoln Laboratory portion of this work was sponsored by the Department of the Air Force under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, recommendations, and conclusions are those of the author and not necessarily endorsed by the United States Government.

Contents

List of Figures

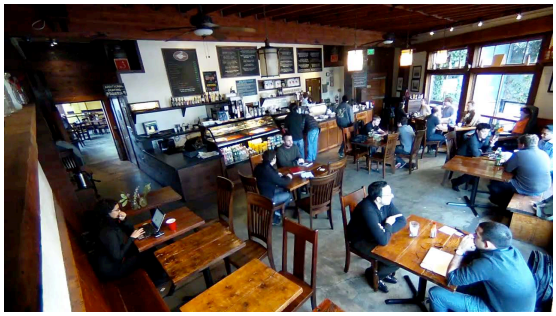
1 Introduction

Video surveillance systems are now commonplace installations in retail environments that range in scale from small, independently owned locations to massive, nationwide chain stores. While the cameras serve as a line of defense against theft and the like, there are myriad other uses for the volumes of data these cameras produce daily. Of particular interest to store owners are methods to translate video footage into meaningful statistics that will enable better tailoring of business operations to meet existing (and predict future) consumer demands. However, implementation of this type of video analysis is still an area of ongoing research, for many challenges are implicit in designing algorithms that can accomplish this translation. They must be able to detect non-rigid humans, then handle tracking them through occlusions, through varying degrees of crowd density, across different angles of view for multiple cameras focused on the same scene, and across times during which detected humans move entirely out of cameras' ranges of view. Among the institutions now endeavoring to find solutions to these complex video analytics problems is PrismSkyLabs, a small San Francisco-based company working towards offering retail clients access to consumer statistics produced using clients' standard surveillance video footage. One such potential client is The Creamery, a busy food and coffee shop situated near a Bay Area Regional Transportation (BART) stop in San Francisco, whose video footage will be utilized for the purposes of this project.

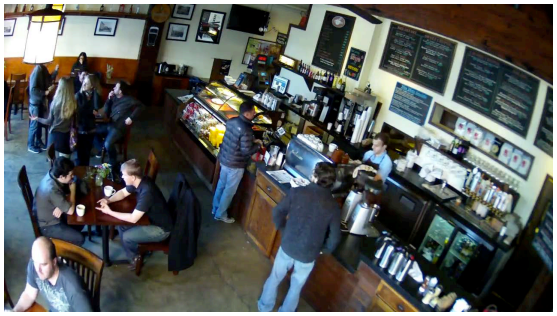
2 Problem Statement

Several statistics are of interest to prospective video analytics retail clients. Chief among these statistics is the number of customers entering the store, coupled with their times of entrance. "Dwell time," or time customers spend in the shop, is also a valuable statistic, particularly when paired with determinations of whether dwelling customers made purchases and whether dwelling customers were alone or in groups. Further, the average wait time for food delivery during particular times of the day is also informative, as are statistics regarding the length of the line of customers waiting for counter service. Of course, the holy grail of retail video analytics involves fusing the above statistics for individual customers, allowing businesses to understand the habits of repeat consumers over time. For the purposes of this project, the primary deliverable is an algorithm that can detect new consumers as they enter and translate these detections into a reasonably accurate customer count over time. Extension deliverables include computing each customer's dwell time and any other meaningful statistics.

While producing an algorithm that can accomplish the aforementioned tasks is difficult in an ideal setting, designing one for The Creamery poses particular challenges given the layout of the shop and the dearth of available data. The stationary cameras installed in The Creamery provide two distinct views of the shop's main area (Fig. 1), which has two separate entrances: the main entrance, which is not directly visible from either camera viewing angle, and the back entrance, a doorway (visible at far left in Fig. 1a) that leads to an adjacent business called The Taqueria; the same individual owns both establishments, and the owner leaves passage between the businesses through this doorway unimpeded at all times. Hence, an accurate customer count must detect consumers entering the shop from both locations. Additionally, the available footage (which is H.264-encoded at a resolution of 720p and a frame rate of approximately 3.25 frames/second) comprises only approximately one full day of business, with extremely limited additional footage available for the purposes of training algorithms or building necessary dictionaries. Because of the limited range of visibility with respect to the entrances of the cameras and because of the limited volume of training footage, developing an algorithm capable of detecting and tracking consumers has been fraught with greater challenges than those anticipated, though many of these have been successfully dispensed with, yielding a reasonably effective algorithm that will serve as an excellent base to be further developed going forward.



(a) Back view



(b) Side view

Figure 1: Camera views of The Creamery. (a) View from far left corner, entryway at right, facing the service counter. (b) View from the side, looking down the length of the sitting area with the service counter at right.

3 Literature Review

The area of video analytics in a retail environment is relatively new, and hence related literature is virtually non-existent. Consequently, the problem itself is not well-defined, and the range of potential solutions is unlimited. In the process of developing our own solution to the specific problem of video analytics for The Creamery, we drew primarily and heavily upon the covariance-based work of Fatih Porikli, Oncel Tuzel, and Peter Meer, whose trio of papers on region covariance as a descriptor for detection and classification [5], human detection using covariance-based training and classification [6], and covariance-based tracking [3] provided an excellent and applicable theoretical framework. We also explored an alternative tracking approach using eigentracking and particle filters based on the work of David Ross, et. al. [4].

4 Implementation

4.1 Overview

Because the original image size for each frame leads to cumbersome computation, we first prefilter and downsample by 2 each frame as it is processed, leading to a 75% reduction in complexity. To facilitate customer entrance detection given that neither camera provides a direct view of both entrances, we utilize only the sequence captured by the camera whose view is shown in Fig. 1a, for it is only this viewpoint that allows detection for both entrances. We then predefine regions, dubbed the "main" and "Taqueria" windows, through which entering/exiting customers must pass (see Fig. 2); our detection algorithm is performed over only these windows in each frame. It is assumed that customers will be upright upon entrance, that there are no other entrances, that there is no need to discriminate between customers and employees as they pass through these windows, and that – because the video sequence does not begin with an empty Creamery – customers present in the establishment at the time the algorithm begins execution will not be included in the customer count.

Then, in each frame, covariance matrices are computed over 6 features in each of 12 sub-regions in both the main and Taqueria windows utilizing the approach described in [6]. Based upon a linear combination of the "votes" from each of these sub-regions, a detection for each window either is or is not declared. If a detection is declared, we apply motion detection utilizing a first-order Markov random field model to produce a binary difference image in order to specifically locate the entering customer within the detection window. To this motion-based difference, a morphological closing operation is applied to fill in any holes, and the centroids of the (white) moving regions are calculated. We assume the centroid located in the white region with the largest area corresponds to the detected human, and center a fixed-size tracking box at this centroid location. At this point, a target is set for

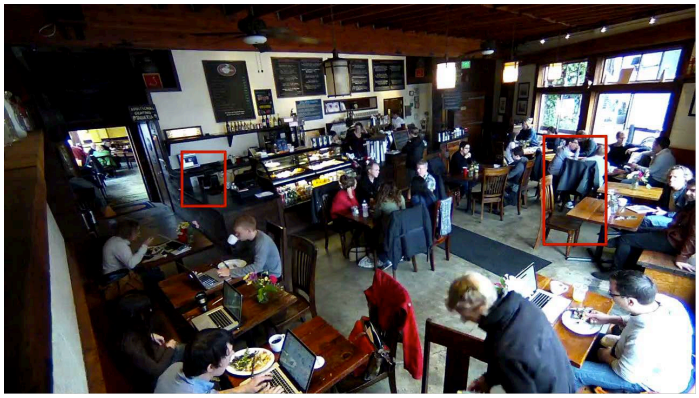


Figure 2: The Taqueria (left) and main (right) entrance windows. The windows are the fixed regions over which covariance matrices are computed in each frame to facilitate customer detection.

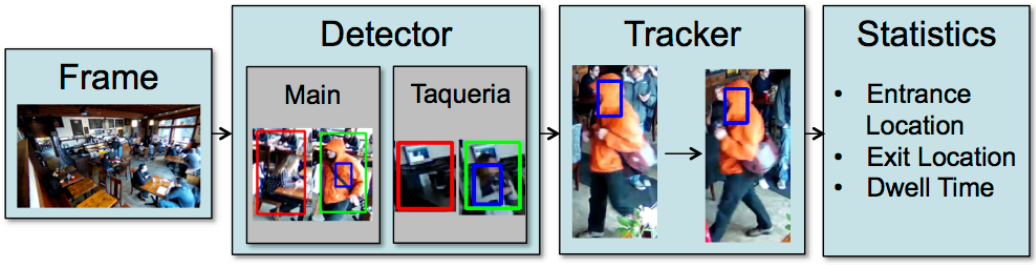


Figure 3: Algorithm overview. The sequence of processing steps begins with a classification-based detection algorithm. Upon detection, tracking boxes are automatically affixed to targets, which are then tracked using a covariance-based tracker. Upon exit, statistics including means of entrance/exit and dwell time are computed.

this customer, and we recompute the covariance matrix over the tracking window using a 9-feature vector that leverages RGB information; tracking then proceeds by using this matrix as the beginning of a history of covariance matrices and as the current mean covariance matrix for this target, again utilizing an approach based on the work of Tuzel, Porikli, and Meer [6],[3]. In subsequent frames, all of the above steps execute as previously detailed, and for the targets already being tracked, gated searches are performed in order to find the new box locations whose covariance matrices are least dissimilar from the stored mean covariance matrices of the corresponding targets. Once found, target locations are moved to these positions, the latest target covariance matrices are recalculated, and the covariance matrix histories and means are updated accordingly. Once a tracking box's position moves beyond the boundaries of one of the entrance windows, the corresponding customer is assumed to have left The Creamery. Upon exit, the algorithm records the doorway(s) through which the customer came in and left, as well as the customer's dwell time.

4.2 Details

4.2.1 Feature Matrix, Integral Image, and Covariance Computation

As in [5], we first establish a feature vector that shall be defined at every pixel in a given image and over which the covariance of features for a given region will be computed. In the course of implementation, several features including x and y locations, image intensities, gradients, second-order derivatives, magnitude of gradients, and gradient orientations were evaluated in different combinations to determine the choice of features that produced the best results for a given application.

For the purposes of classification-based detection, we build in a modicum of invariance to ranges of clothing color and patterns by eliminating the use of color or intensity as features, yielding the following 6-dimensional feature vector:

$$F_D(x,y,:) = \begin{bmatrix} x & y & \left|\frac{\partial I(x,y)}{\partial x}\right| & \left|\frac{\partial I(x,y)}{\partial y}\right| & \left|\frac{\partial^2 I(x,y)}{\partial x^2}\right| & \left|\frac{\partial^2 I(x,y)}{\partial y^2}\right| \end{bmatrix} \tag{1}$$

To form the feature matrix, we first compute each of these features over the image, then concatenate them together in the third dimension to form a $360 \times 640 \times 6$ feature matrix, which can then be straightforwardly utilized to compute the covariance matrices over the sub-regions of interest as follows. If the scalar mean of each of the 2-D features is calculated, then the covariance for any given $M \times N$ sub-region is

$$C_R = \frac{1}{MN} \sum_{k=1}^{MN} (\mathbf{F}_D - \boldsymbol{\mu}_R)(\mathbf{F}_D - \boldsymbol{\mu}_R)^T \tag{2}$$

where $\boldsymbol{\mu}_R$ is the vector of the scalar means for the pixels contained within the region of computation. Performing the above computation is burdensome, however, when it must be computed thousands of times for a given frame. Hence, to expedite the computation, our algorithm makes use of the integral image approach devised by Porikli, Tuzel and Meer and enumerated in [5]. Simply put, the integral image for a 2-D matrix is the running sum of all values above and to the left of a given matrix element. By computing these integral images for each 360×640 feature running down the depth of the 3-D feature matrix, the self-terms in the covariance matrix calculation are easily computed, and by computing these integral images across 6×6 matrices of cross-features running the width of the 3-D matrix, the cross-terms are easily computed as well. Thus, as in [5], the integral images are computed as:

$$P(x',y',i) = \sum_{x<x',y<y'} F(x,y,i) \qquad Q(x',y',i,j) = \sum_{x<x',y<y'} F(x,y,i)F(x,y,j) \qquad i,j = 1,\ldots,6$$

where P is the $360\times640\times6$ integral image matrix for self-features and Q is the $360\times640\times6\times6$ integral image for cross-features. While, for the sake of space, the full derivation will not be repeated here, it is fairly straightforward to show that, once integral images are computed, covariance matrices over a rectangular region containing n pixels and bounded at the top left by (x',y') and at the bottom right by (x'',y'') can be computed as:

$$\begin{aligned} \mathbf{C}_{R(x',y';x'',y'')} = \frac{1}{n-1}\Big[& \mathbf{Q}_{x'',y''} + \mathbf{Q}_{x',y'} - \mathbf{Q}_{x'',y'} - \mathbf{Q}_{x',y''} \\ & - \frac{1}{n}\left(\mathbf{p}_{x'',y''} + \mathbf{p}_{x',y'} - \mathbf{p}_{x',y''} - \mathbf{p}_{x'',y'}\right)\left(\mathbf{p}_{x'',y''} + \mathbf{p}_{x',y'} - \mathbf{p}_{x',y''} - \mathbf{p}_{x'',y'}\right)^T \Big] \end{aligned} \tag{3}$$

where

$$\mathbf{p}_{x,y} = \begin{bmatrix} P(x,y,1) & P(x,y,2) & \ldots & P(x,y,d) \end{bmatrix}^T, \qquad \mathbf{Q}_{x,y} = \begin{bmatrix} Q(x,y,1,1) & \ldots & Q(x,y,1,d) \\ \vdots & \ddots & \vdots \\ Q(x,y,d,1) & \ldots & Q(x,y,d,d) \end{bmatrix}$$

The points used for the covariance calculation for region R are illustrated in Figure 4.

4.2.2 Dissimilarity

Once covariance matrices are computed, we need some means to compare their similarity for the purposes of covariance matrix use in detection and tracking; however, covariance matrices do not exist in Euclidean space, and hence we cannot simply compute their "distance." There are a few existing methods for computing the likeness of two covariance matrices, but the one utilized by

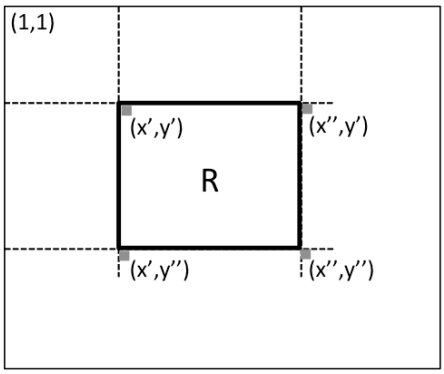


Figure 4: Illustration of the region points used for the covariance calculation.

Porikli, et. al. and the one implemented in our algorithm is the "dissimilarity metric" described in [1]. For two covariance matrices, $\mathbf{C}_i$ and $\mathbf{C}_j$, the metric is:

$$\rho(\mathbf{C}_i, \mathbf{C}_j) = \sqrt{\sum_{k=1}^{d} \log^2 \lambda_k(\mathbf{C}_i, \mathbf{C}_j)} \tag{4}$$

where λ_k are the generalized eigenvalues of $\mathbf{C}_i$ and $\mathbf{C}_j$. Hence, the more similar the two matrices are, the closer the generalized eigenvectors are to forming the identity matrix, thus the closer the generalized eigenvalues are to 1, and thus the closer the sum of the squared natural logarithms of the generalized eigenvalues are to 0. Hence, the closer $\rho(\mathbf{C}_i, \mathbf{C}_j)$ is to 0, the more alike the two matrices; conversely, the larger $\rho(\mathbf{C}_i, \mathbf{C}_j)$, the more dissimilar the two matrices.

4.2.3 Mean Covariance (Model) Computation and Update

For the purposes of detection, comparing to a dictionary of particular covariance matrices produced from a training set will inevitably fail, for no dictionary can capture all possible viewpoints, pose changes, and shape deformations of entering humans; it is instead desirable to have some aggregate notion of covariance that maximally captures these possibilities within a single covariance matrix. Likewise, for the purposes of tracking, modeling a target's appearance by the covariance of a region around the target will certainly fail over time as the non-rigid target moves, undergoing pose variations and shape deformations, unless the model is updated to include the covariances of regions around the target in later frames. To apply this notion of aggregated covariance (aggregate either in the sense of a single matrix that represents several different humans' covariances or in the sense of a single matrix that represents a single human's covariance over time), both our detection and tracking algorithms follow the lead of Porikli et. al. and make use of the notion of the mean covariance matrix described in [6] and [3].

As with the dissimilarity metric, finding the mean of covariance matrices is not mathematically straightforward, for the matrices do not exist in Euclidean space. Because covariance matrices are positive semi-definite, they can be viewed as existing on a Riemannian manifold, which is a space that is locally Euclidean and on which distances are given by geodesics rather than straight line lengths. Without delving deeply into the heavy underlying mathematics, it suffices to say that the locally Euclidean tangent plane for a positive semi-definite matrix can be reached from the manifold space by a logarithmic mapping, and the manifold space can be reached from the tangent plane by an inverse exponential mapping. Given two positive semi-definite matrices $\mathbf{C}_1$ and $\mathbf{C}_2$, these mappings are given by:

$$\log_{\mathbf{C}_1}(\mathbf{C}_2) = \mathbf{C}_1^{\frac{1}{2}} \log\left(\mathbf{C}_1^{-\frac{1}{2}} \mathbf{C}_2 \mathbf{C}_1^{-\frac{1}{2}}\right) \mathbf{C}_1^{\frac{1}{2}}, \qquad \exp_{\mathbf{C}_1}(\mathbf{C}_2) = \mathbf{C}_1^{\frac{1}{2}} \exp\left(\mathbf{C}_1^{-\frac{1}{2}} \mathbf{C}_2 \mathbf{C}_1^{-\frac{1}{2}}\right) \mathbf{C}_1^{\frac{1}{2}} \tag{5}$$

where logarithmic, exponential, and square root functions are applied in the matrix sense. According to the mathematics in [6], on the locally Euclidean tangent plane the distance between two matrices is then given by:

$$d^2(\mathbf{C}_1, \mathbf{C}_2) = tr\left(\log^2\left(\mathbf{C}_1^{-\frac{1}{2}}\mathbf{C}_2\mathbf{C}_1^{-\frac{1}{2}}\right)\right)$$

So, for a group of covariance matrices, we seek the mean as the matrix that minimizes the sum of squared distances, i.e., $\bar{\mathbf{C}} = \text{argmin}_{\mathbf{C}} \sum_{t=1}^{T} d^2(\mathbf{C}, \mathbf{C}_t)$. Clearly, this matrix cannot be computed directly, but can be found instead by gradient descent, one means for which is given in [2] as:

$$\bar{\mathbf{C}}^{i+1} = \exp_{\bar{\mathbf{C}}^i}\left(\frac{1}{T}\sum_{t=1}^{T}\log_{\bar{\mathbf{C}}^i}(\mathbf{C}_t)\right) \tag{6}$$

The number of iterations of the gradient descent algorithm that are performed for each mean covariance update is up to the engineer; Porikli et. al. imply that 10 iterations are sufficient for convergence to the minimum, and hence we also utilize that number of iterations. In addition, it is worth noting that the above gradient descent algorithm weights all covariance matrices during the past T observations equally, regardless of their distance in time or their dissimilarity from the mean. Hence, this mean computation can be weighted in any number of ways so as to better bias the mean matrix toward its most significant contributors; to this end, we also incorporated a parameter that allows adjustment of the gradient descent algorithm to either weight the distances by inverse dissimilarity (placing more emphasis on more similar matrices) as:

$$\bar{\mathbf{C}}^{i+1} = \exp_{\bar{\mathbf{C}}^i}\left(\frac{1}{\sum_{t=1}^{T}\rho^{-1}(\mathbf{C}_t, \bar{\mathbf{C}}^*)}\frac{1}{T}\sum_{t=1}^{T}\rho^{-1}(\mathbf{C}_t, \bar{\mathbf{C}}^*)\log_{\bar{\mathbf{C}}^i}(\mathbf{C}_t)\right) \tag{7}$$

or exponentially by distance in time (placing more emphasis on more recent matrices) as:

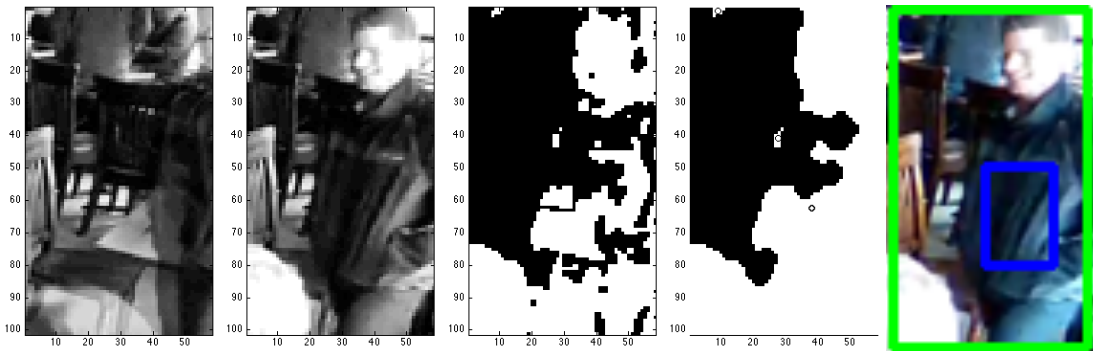
$$\bar{\mathbf{C}}^{i+1} = \exp_{\bar{\mathbf{C}}^i}\left(\frac{1}{\sum_{t=1}^{T}\alpha^t}\frac{1}{T}\sum_{t=1}^{T}\alpha^t\log_{\bar{\mathbf{C}}^i}(\mathbf{C}_t)\right) \tag{8}$$

4.3 The Algorithms

4.3.1 Detection

The detection approach we ultimately utilized was derived from [6], and is based upon a machine learning algorithm that trains a strong classifier using LogitBoost. Because there was not enough available training data, it was necessary to utilize a sequence of 5,000 frames from the end of the long test sequence as a training set, though we of course did not later utilize any of this section of footage for testing. Using these 5,000 frames, we established ground truth for human entrances in both the main and taqueria windows, labeling images that contained human entrances as '1', and images that did not as '0.' Of these 5,000 images, approximately 50 contained entering customers, and these 50 became the human training set; from the remaining 4,950, a set of 200 images was randomly chosen to represent the background, and became the background training set. These 250 images were fed into the LogitBoost algorithm along with a set of several hundred possible sub-regions for each window, which was manually defined by sliding windows of various sizes over each detection window.

With these inputs, LogitBoost works by combining several weak classifier regions into a single strong classifier as follows. Consider for the moment only a single sub-region from the set of possible sub-regions. The covariance matrix over this sub-region is calculated for all 50 human images using the feature vector in (1) and the integral image covariance calculation in (3). Then, using these 50 covariance matrices, the mean covariance matrix for the sub-region is computed with equal weighting using (6). Next, the covariance matrices for all of the 200 background images are computed, and all 250 covariance matrices – both human and background – are projected into the tangent space



(a) Background Model (b) Original Image (c) Motion via MRF (d) Closed, Centroid (e) Final Detection

Figure 5: Automatically setting a tracking window. Once a customer's entrance is detected, first-order Markov random field motion detection is applied, yielding a binary image in which white areas correspond to (moving) foreground objects and black areas correspond to background. Holes in this image are closed using a morphological closing operator, then centroids of closed white regions are calculated. The white region with the largest area is assumed to be the entering human, and a fixed-size tracking box centered on the corresponding centroid location is placed on the original image.

of the previously computed mean covariance utilizing the mappings in (5). In this tangent space, a linear regression is performed with the goal of finding the line that maximally separates the background covariance matrices from the human covariance matrices; it is highly unlikely that such a linear discriminator will perfectly separate the two, and hence there is a misclassification error that is associated with the chosen line. This process is then repeated during the same iteration for every one of the possible sub-regions fed into LogitBoost; the sub-region whose linear discriminator leads to the smallest misclassification error is then chosen as the first weak classifier region. Before proceeding to the next iteration, LogitBoost determines which images were incorrectly classified by the linear discriminator in the chosen region, and increases the weighting of these images for the next iteration while decreasing the weighting of the correctly classified images.

In the next iteration, the process is repeated again, this time utilizing the same mean covariance calculation in (5), but with the adjusted weights produced by the previous iteration. Again, at the conclusion of the iteration, another sub-region is chosen as the best weak classifier, this weak classifier is added in a linear combination to the previously chosen weak classifier, and weights are again adjusted based on which images were incorrectly classified using the chosen combination of weak classifiers. Theoretically, this process can continue until saturation; however, due to time constraints, we chose to iterate until 12 weak classifiers were produced for each detection window, yielding final strong classifiers for each window that are linear combinations of their respective 12 underlings. Once the training is completed off-line, the strong classifier can simply be applied to images in the video sequence to determine detections. For each new frame, the covariance matrix over each of the 12 chosen sub-regions in each window is calculated; the resulting covariance matrices are then fed into the strong classifier for each window, each of which makes a binary decision regarding whether a customer is entering The Creamery.

4.3.2 Tracking

Once a detection has been declared by the classification algorithm, a tracking window is set by the first-order Markov random field motion detection/morphological closing/centroid-finding sequence of operations, as discussed in the implementation overview section above (see Fig. 5).

Upon setting the tracking window, the target's feature matrix, integral image, and covariance

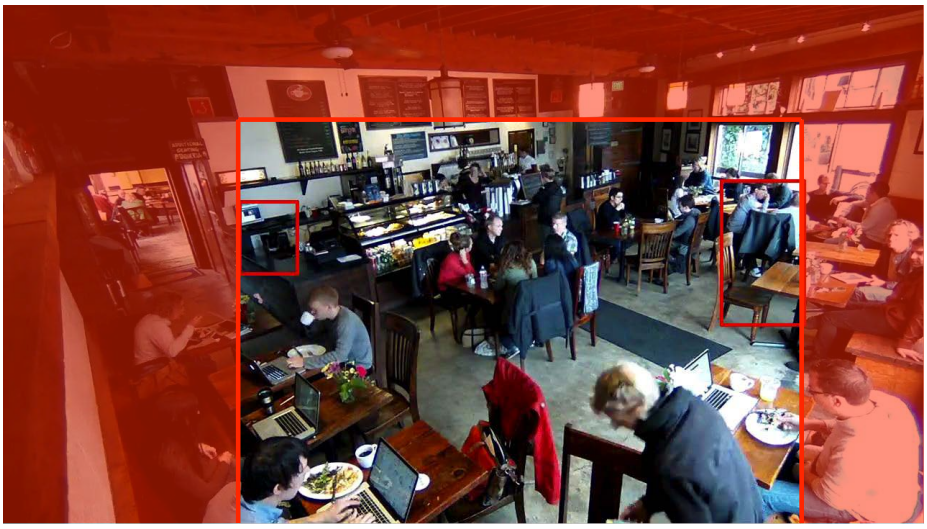


Figure 6: Exit boundaries. Once a customer's tracker location crosses into the red area, the customer is assumed to have exited The Creamery, the track is removed from the tracking data structure, and pertinent statistics regarding the tracked customer are output to the console.

are immediately recalculated using a feature vector that enables the algorithm to now capitalize upon variation in target appearance in the RGB space:

$$F_T(x, y, :) = \left[x \quad y \quad R(x,y) \quad G(x,y) \quad B(x,y) \quad \left|\frac{\partial I(x,y)}{\partial x}\right| \quad \left|\frac{\partial I(x,y)}{\partial y}\right| \quad \left|\frac{\partial^2 I(x,y)}{\partial x^2}\right| \quad \left|\frac{\partial^2 I(x,y)}{\partial y^2}\right|\right] \tag{9}$$

This new target is then added to a structure that maintains pertinent information on all targets being simultaneously tracked. For each target, the frame corresponding to his/her initial detection is stored, as are the target's last location, mean covariance matrix, and history of covariance matrices dating back 10 frames. In each subsequent frame and for each target, we perform a gated search over the surrounding region of each target's last location, seeking the location whose covariance matrix is least dissimilar from the target's stored mean covariance. This becomes the target's newest location, and the corresponding matrix is added to the target's covariance matrix history. If the track has been maintained longer than 10 frames, the oldest covariance matrix is removed from the history to make room for the new addition. Then, the mean covariance is updated by recomputing the mean over the new matrix history using the time-weighted gradient descent approach in (8). This process continues for every target until a target's tracking box location passes outside of the boundary determined by the outermost edges of the detection windows. That is, a target continues to be tracked until it is determined (based on the tracker's location) that the target has exited through either the main or the Taqueria window (see Fig. 6). At this point, pertinent statistics including the customer's entrance window, exit window, and dwell time are output to the console.

4.4 Challenges and Other Approaches Tried

In addition to the challenging nature of the video analytics problem posed even against the backdrop of an ideal setting with unlimited data and optimal camera viewpoints, the problem of video analytics for The Creamery posed significant specific challenges. As has already been alluded to, the available data was very limited, comprising less than a single full day of business at The Creamery and neither camera offered a view that included the main entrance directly. Because our algorithm hinged upon being able to monitor both of the entrances simultaneously, it was necessary to further constrain the scope of available data by limiting the sequences used for training and testing only to one very short and one long clip recorded by the camera whose viewpoint is shown in Fig. 1a. Using only

the short sequence for detection training yielded unsatisfactory results regardless of the detection approach used. Further complicating matters is the fact that, for the purpose of tracking, a 15 frames/second frame rate – nearly five times the rate of The Creamery footage – is considered quite challenging. Finally, it bears mentioning that the above algorithm was implemented in MATLAB, and as such, there was limited processing capability that prevented fully loading the video sequence into memory before processing. Consequently, it was necessary to devise an approach that allowed frame-by-frame processing through the MATLAB Central function `mmread` using a nested function call. Because of this, it was also necessary to devise a method for storing key variables in the base workspace following each frame's processing, then reloading them for each subsequent frame.

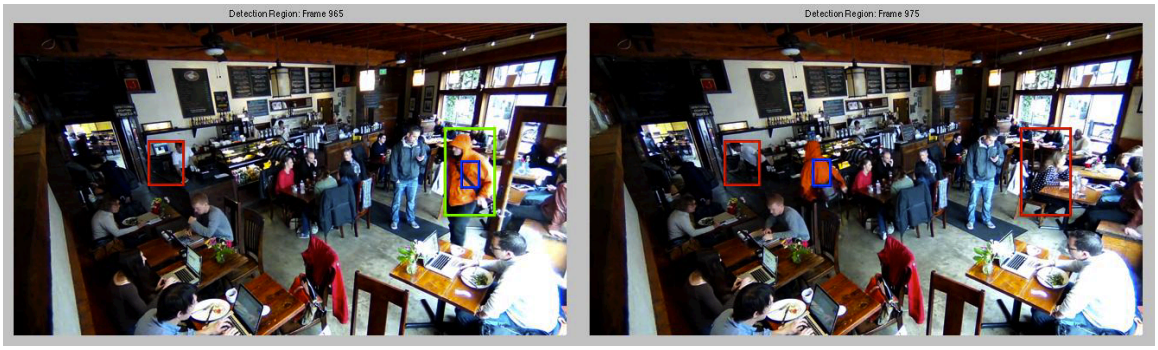
Originally, our detection approach utilized a Bayesian decision model. Under this model, we computed background and human dictionaries for each window using the short training clip, and compared the covariance of the entire detection window to each of the entries in the corresponding human and background dictionaries, declaring a detection whenever the minimum dissimilarity over all comparisons was produced by a comparison to an entry in a human dictionary. The detection rate for this approach (approximately 60%) was unsatisfactory, and we explored options for improvement including window repositioning, window resizing, variance normalization, and employing alternate features in the feature matrix. Ultimately, none of these attempts yielded substantive improvement, but opting for the classification detection algorithm produced a successful detection rate of approximately 80%. It should be noted that the detection rate was 100% for the first 2,000 frames, but worsened over time. At present, we posit that this is due to the use of a fixed threshold in the strong classifier decision-making process; however, it may be necessary to employ a variable threshold to account for such things as global changes in luminance over the course of a business day.

We also tested various alterations to the tracking algorithm, including exhaustive search, varying the sizes of the automatically-set tracking windows, and employing alternate means of mean covariance computation weighting, as enumerated in (7) and (8). Further, we tested a completely different tracking algorithm on the video, but found that the algorithm – despite the fact that it is an excellent tracker on higher frame rate videos – produced far less impressive results than were produced with the covariance tracker. The algorithm tested, based on the incremental learning tracker in [4], utilizes eigenbases as target appearance models and updates the eigenbases on-line to account for changes in appearance. Also, rather than using the gated exhaustive search approach employed by the covariance tracker, it uses a particle filter approach. However, because the incremental learning tracker assumes smooth motion and because it is designed for grayscale images and thus cannot capitalize upon the information inherent in a three-element color space, this alternative tracker could not maintain any track beyond a span of two successive frames.

5 Experimental Results

For the purposes of visualizing the results of the algorithm, the detection windows are shown by red boxes. If a detection occurs, the box surrounding the corresponding detection window turns green. Two things may happen visually at this point: if the motion detection algorithm does not yield a foreground region whose area is above a fixed threshold, the automatic tracking-box setter cancels the detection, and the detection window turns yellow. (Note: This was a failsafe initially built in as part of an attempt to improve our first approach to detection. However, it worked so well that we chose to leave it in despite the success of the classification detection approach.) Otherwise, a blue box will pop up on the entering customer; this box delineates that customer's initial tracking window. In each subsequent frame (ideally) the blue target boxes track their owners, and the detection window visual notifications proceed as above.

Based on this, our experimental results are fairly straightforward. The detection algorithm works very well. There are occasional instances where target boxes are too poorly initially located to permit successful tracking, though these instances are infrequent. Considering the frame rate issues, our tracking algorithm also works remarkably well; however, it is here that the algorithm (or, perhaps, the dataset) demands substantial improvement. Tracks are often maintained for two



(a) Frame 1 (b) Frame 11

Figure 7: Example of successful tracking. Here we were able to maintain track until the person exited the Creamery via the Taqueria.

(a) Frame 1 (b) Frame 8

Figure 8: Example of unsuccessful tracking. Here the person initializes two trackers (in subsequent frames) and then "loses" them when he passes through regions with similar color/texture as his jacket (the tracked object).

dozen consecutive frames, but are easily drawn off their targets if the targets reposition so as to incorporate background pixels into the mean covariance calculation. Additionally, targets can pass tracking boxes back and forth, and because target motion due to the low frame rate is so choppy, it is no small feat that the algorithm works at all, given that the covariance matrix of a target in one frame may be substantially different from the covariance matrices of the target in previous frames. It is possible that setting tighter boxes or allowing variable tracker box sizing may help address some of these issues, though the best first remedy would be smooth motion (i.e., higher frame rate data).

6 Conclusions and Future Work

The algorithm we have designed holds tremendous promise and should continue to be tweaked for further improvement. It appears that incorporating a variable threshold in the detection portion of the algorithm, as well as training on a larger set of data, over a larger set of window sub-regions, and for a larger set of weak classifiers, may in fact lead to nearly a 100% detection rate. Additionally, refining the means of automatic track setting should be explored with the goal of ensuring trackers are centered on targets without enclosing any background pixels. It is also likely that the tracking algorithm would benefit from including scalability with respect to target tracking boxes. However,

regardless of the initial windows set or how they are scaled as tracking proceeds, it is unlikely that any sustained tracking will be nearly always successful on a sequence that has undergone severe temporal decimation, as has The Creamery footage. With this in mind, it would be worthwhile to apply the algorithm in its present form to a Creamery sequence that has not been temporally downsampled, which would allow a fair assessment of the track-setting and tracking components of the algorithm as they now stand. Clearly, strong video analytics results cannot possibly be produced without successful implementation of tracking. However, if it were found that our tracking algorithm were accurate on a higher frame rate sequence, it would be easily adaptable to translation of surveillance video into meaningful consumer statistics.

References

[1] W. Forstner and B. Moonen, "A metric for covariance matrices," tech. rep., Dept. of Geodesy and Geoinformatics, Stuttgart University, 1999.

[2] X. Pennec, P. Fillard, and N. Ayache, "A riemannian framework for tensor computing," *International Journal of Computer Vision*, vol. 66, no. 1, pp. 41–66, 2006.

[3] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on means of riemannian manifolds," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[4] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, pp. 125–141, 2008.

[5] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3952 of *Lecture Notes in Computer Science*, pp. 589–600, Springer, 2006.

[6] O. Tuzel, F. Porikli, and P. Meer, "Human detection va classification on riemannian manifolds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007.