Visual Tracking of Human Visitors under Variable-Lighting Conditions for a Responsive Audio Art Installation

Andrew B. Godbehere, Akihiro Matsukawa, Ken Goldberg

Abstract—**For a responsive audio art installation in a skylit atrium, we introduce a single-camera statistical segmentation and tracking algorithm. The algorithm combines statistical background image estimation, per-pixel Bayesian segmentation, and an approximate solution to the multi-target tracking problem using a bank of Kalman filters and Gale-Shapley matching. A heuristic confidence model enables selective filtering of tracks based on dynamic data. We demonstrate that our algorithm has improved recall and F_2-score over existing methods in OpenCV 2.1 in a variety of situations. We further demonstrate that feedback between the tracking and the segmentation systems improves recall and F_2-score. The system described operated effectively for 5-8 hours per day for 4 months; algorithms are evaluated on video from the camera installed in the atrium. Source code and sample data is open source and available in OpenCV.**

I. Introduction

We present the design of a computer vision system that separates video into "foreground" and "background", and then segments and tracks people in the foreground while being robust to variable lighting conditions. The system we present ran a successful interactive audio art installation called "Are We There Yet?" from March 31 - July 31 2011 at the Contemporary Jewish Museum in San Francisco, California. Using video collected during the operation of the installation, under variable illumination created by myriad skylights, we demonstrate a significant performance improvement over existing methods in OpenCV 2.1. The system runs in real-time (15 frames per second), requires no training datasets or calibration, and uses only a couple seconds of video to initialize.

Our system consists of two stages: first is a probabilistic foreground segmentation algorithm that identifies possible foreground objects using Bayesian inference with an estimated time-varying background model and an inferred foreground model, described in Section II. The background model consists of nonparametric distributions on RGB colorspace for every pixel in the image. The estimates are adaptive; newer observations are more heavily weighted than old observations to accommodate variable illumination. The second portion is a multi-visitor tracking system, described in Section III, which refines and selectively filters the proposed foreground objects. Selective filtering is achieved with a heuristic confidence model, which incorporates error covariances calculated by the multi-visitor tracking algorithm. For the tracking subsystem, in Section III, we apply a bank of Kalman filters [18] and match tracks and observations with the Gale-Shapley algorithm [13], with preferences related to the Mahalanobis distance weighted by the estimated error covariance. Finally, a feedback loop from the tracking subsystem to the segmentation subsystem is introduced: the results of the tracking system selectively update the background image model, avoiding regions identified as foreground. Figure 1 illustrates a system-level block diagram. Figure 2 offers an example view from our camera and some visual results of our algorithm.

The operating features of our system are derived from the unique requirements of an interactive audio installation. False negatives, i.e. people the system has not detected, are particularly problematic because they expect a response from the system and become frustrated or disillusioned when the response doesn't come. Some tolerance is allowed for false positives, which add audio tracks to the installation; a few add texture and atmosphere. However, too many false positives creates cacophony. Performance of vision segmentation algorithms is often presented in terms of precision and recall [30]; many false negatives corresponds to a system with low recall. Many false positives lowers precision. We discuss precision, recall, and the F_2-score in Section I-D.

Section IV contains an experimental evaluation of the algorithm on video collected during the 4 months the system operated in the gallery. We evaluate performance with recall and the F_2-score [16], [24]. Our results on three distinct tracking scenarios indicate a significant performance gain over the algorithms in OpenCV 2.1, when used with the recommended parameters. Further, we demonstrate that the feedback loop between the segmentation and tracking subsystems improves performance by further increasing recall and the F_2-score.

A. Related Work

The structure of the computer vision system we propose is inspired by algorithms in OpenCV 2.1 [5], [8], [17], [22], which offers a variety of probabilistic foreground detectors, including both parametric and nonparametric approaches, along with several multi-target tracking algorithms, utilizing, for example, the mean-shift algorithm [10] and particle filters [28]. Another approach applies the Kalman Filter on any detected connected component, and doesn't attempt collision

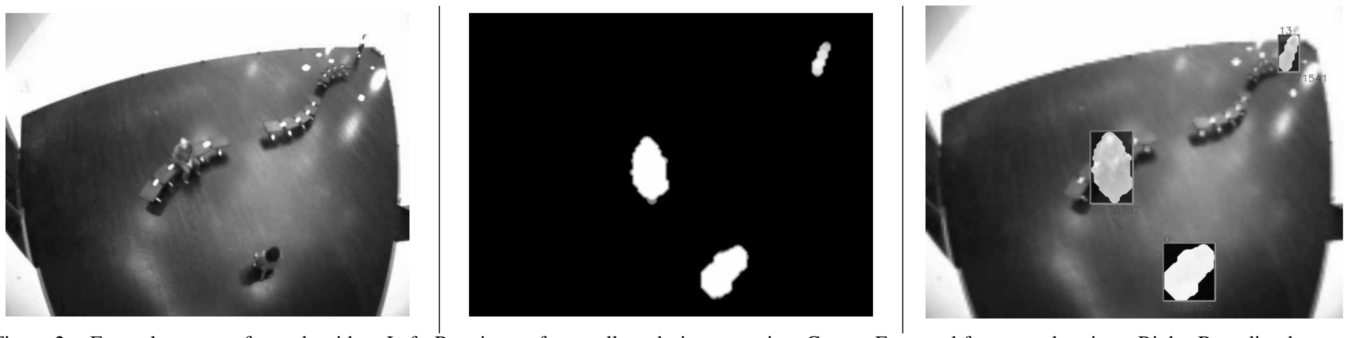

Figure 2. Example output of our algorithm. Left: Raw image from gallery during operation. Center: Extracted foreground regions. Right: Bounding boxes of tracked foreground objects and annotated confidence levels.

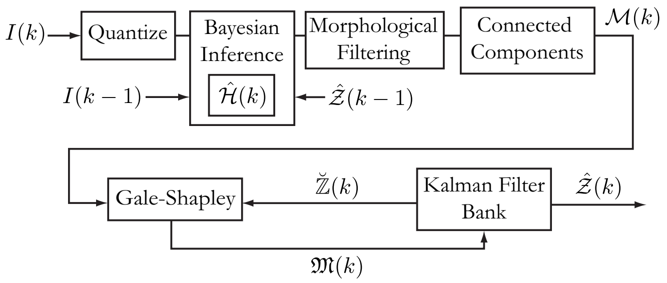

Figure 1. Algorithm Block Diagram. An image $I(k)$ is quantized in color-space, and compared against the statistical background image model, $\hat{\mathcal{H}}(k)$, to generate a posterior probability image. This image is filtered with morphological operations and then segmented into a set of bounding boxes, $\mathcal{M}(k)$, by the connected components algorithm. The Kalman filter bank maintains a set of tracked visitors $\hat{\mathbb{Z}}(k)$, and has predicted bounding boxes for time k, $\breve{\mathbb{Z}}(k)$. The Gale-Shapley matching algorithm pairs elements of $\mathcal{M}(k)$ with $\breve{\mathbb{Z}}(k)$; these pairs are then used to update the Kalman Filter bank. The result is $\hat{\mathcal{Z}}(k)$, the collection of pixels identified as foreground. This, along with image $I(k)$, is used to update the background image model to $\hat{\mathcal{H}}(k+1)$. This step selectively updates only the pixels identified as background.

resolution. We evaluated these algorithms for possible use in the installation, although they exhibited low recall, i.e. people in the field of view of the camera were too easily lost, even while moving. This problem arises from the method by which the background model is updated: every pixel of every image is used to update the histogram, so pixels identified as foreground pixels are used to update the background model. The benefit is that a sudden change in the appearance of the background in a region is correctly identified as background; the cost is the frequent misidentification of pedestrians as background. To mitigate this problem, our approach uses dynamic information from the tracking subsystem to filter results of the segmentation algorithm, so only the probability distributions associated with background pixels are updated.

The class of algorithm we employ is not the only class available for the problem of detecting and tracking pedestrians in video. A good overview of the various approaches is provided by Yilmaz et al. [40]. Our foreground segmentation algorithm is derived from a family of algorithms which model every pixel of the background image with probability distributions, and use these models to classify pixels as foreground or background. Many of these algorithms are parametric [9], [14], leading to efficient storage and computation. In outdoor scenes, mixture-of-gaussian models capture complexity in the underlying distribution that single gaussian distribution models miss [17], [31], [34], [41]. Ours is nonparametric: it estimates the distribution itself rather than its parameters. For nonparametric approaches, kernel density estimators are typically used, as distributions on color-space are very high-dimensional constructs [11]. To efficiently store distributions for every pixel, we make a sparsity assumption on the distribution similar to [23], i.e. the random variables are assumed to be restricted to a small subset of the sample space.

Other algorithms use foreground object appearance models, leaving the background unmodeled. These approaches use support-vector-machines, AdaBoost [12], or other machine learning approaches in conjunction with a training dataset to develop classifiers that are used to detect objects of interest in images or videos. For tracking problems, pedestrian detection may take place in each frame independently [1], [37]. In [29], these detections are fed into a particle-filter multi-target tracking algorithm. These single-frame detection approaches have been extended to detecting patterns of motion, and Viola et al. [38] show that incorporation of dynamical information into the segmentation algorithm improves performance. Our algorithm is based on different operating assumptions, notably requiring very little training data; initialization uses only a couple seconds of video.

A third, relatively new approach, is Robust-PCA [7], which neither models the foreground nor the background, but assumes that the video sequence may be decomposed as $I = L + S$, where L is low-rank and S is sparse. The relatively constant background image generates a "low-rank" video sequence, and foreground objects passing through the image plane introduce sparse errors into the low-rank video sequence. Candes et al. [7] demonstrate the efficacy of this approach for pedestrian segmentation, although the algorithm requires the entire video sequence to generate the segmentation, so it is not suitable for our real-time application.

Generally, multi-target tracking approaches attempt to find the precise tracks that each object follows, to maintain identification of each object [4]. For our purposes, this is unnecessary, and we avoid computationally intensive approaches like particle-filters [28], [29], [39]. Our sub-optimal approximation of the true maximum likelihood multi-target tracking algorithm allows our system to avoid exponential complexity [4] and to run in real-time. Similar object-to-track matching utilizing the Gale-Shapley matching algorithm is explored in [2].

Other authors have pursued applications of control algorithms to art [3], [15], [19], [20], [21], [32], and the emerging applications signal a growing maturity of control technology in its ability to coexist with people.

B. Notation

We consider a length N image sequence, denoted $\{I\}_{k=0}^{N-1}$. The k^{th} image in the sequence is denoted $I(k) \in \mathcal{C}^{w \times h}$, where w and h are the image width and height in pixels, respectively, and $\mathcal{C} = \{(c_1, c_2, c_3) : 0 \leq c_i \leq q-1\}$ is the color-space for a 3-channel video. For our 8-bit video, $q = 256$, but quantization described in Section II-A will alter q. We downsample the image by a factor of 4 and use linear interpolation before processing, so w and h are assumed to refer to the size of the downsampled image. Denote the pixel in column j and row i of the k^{th} image of the sequence as $I_{ij}(k) \in \mathcal{C}$. Denote the set of possible subscripts as $\mathfrak{I} \equiv \{(i,j) : 0 \leq i < h, 0 \leq j < w\}$, referred to as the "index set", and $(0,0)$ is the upper-left corner of the image plane. For this paper, if $A \subset \mathfrak{I}$, let $A^c \subset \mathfrak{I}$ and $A \bigcup A^c = \mathfrak{I}$. Define an inequality relationship for tuples (x,y) as $(x,y) \leq (u,v)$ if and only if $x \leq u$ and $y \leq v$.

The color of each pixel is represented by a random variable, $\mathbf{I}_{ij}(k) \sim H_{ij}(k)$, where $H_{ij}(k) : \mathcal{C} \to [0,1]$ is a probability mass function. Using a "lifting" operation L, map each element $c \in \mathcal{C}$ to unique axes of $\mathbb{R}^{q^3}$ with value $[H_{ij}(k)](c)$ to represent probability mass functions as vectors (or normalized histograms), a convenient representation for the rest of the paper. Note that $\vec{1}^T H_{ij}(k) = 1$, when conceived of as a vector; $\vec{1} \in \mathbb{R}^{q^3}$. Denote an estimated distribution as $\hat{H}_{ij}(k)$. Let $\hat{\mathcal{H}}(k) = \{\hat{H}_{ij}(k) : (i,j) \in \mathfrak{I}\}$ represent the background image model, as in Figure 1.

A foreground object is defined as an 8-connected collection of pixels in the image plane corresponding to a visitor. Define the set of foreground objects at time k as $\mathbb{X}(k) = \{\chi_n \subset \mathfrak{I} : n < R(k)\}$, where χ_n represents an 8-connected collection of pixels in the image plane, and $R(k)$ represents the number of foreground objects at time k. Let $F(k) = \bigcup_{\chi \in \mathbb{X}(k)} \chi$ be the set of all pixels in the image associated with the foreground. We define the minimum bounding box around each contiguous region of pixels with the upper left and lower right corners: let $x_n^+ = \arg\min_{(i,j)\in\mathfrak{I}}(i,j)$ s.t. $(i,j) \geq (u,v)\ \forall (u,v) \in \chi_n$, and $x_n^- = \arg\max_{(i,j)\in\mathfrak{I}}(i,j)$ s.t. $(i,j) \leq (u,v)\ \forall (u,v) \in \chi_n$. The set of pixels within the minimum bounding box of χ_n is $\bar{\chi}_n = \{(i,j) : x_n^- \leq (i,j) \leq x_n^+\}$. Then, let $\overline{F}(k) = \bigcup_{n<R(k)} \bar{\chi}_n$, the set of all pixels within the minimum bounding boxes around each foreground object. $\overline{F}(k) \subset \mathfrak{I}$ is referred to as the foreground bounding box support of the image $I(k)$.

The tracking algorithm returns a set $\hat{\mathcal{Z}}(k) \subset \mathfrak{I}$, indicating the pixels identified as foreground, described in more detail in Section III. Throughout, variants of the symbol $\mathbb{Z}$ will refer to collections of tracks, not to the set of integers.

C. Assumptions

With this notation, we make the following assumptions:

1) Foreground regions of images are small: let $B(k) \equiv F(k)^c$ represent the set of pixels associated with the background. Assume that $|B(k)| \gg |F(k)|$.

2) The color distribution of a given pixel changes slowly relative to the frame rate. The appearance is allowed to change rapidly, as with a flickering light, but the distribution of colors at a given pixel must remain essentially constant between frames. In practice, this condition is only violated in extreme situations, as when lights are turned on or off. High-level logic helps the algorithm recover from a violation of this assumption : Interpreting $H_{ij}(k)$ as a vector, $\exists \epsilon > 0$ such that for all i,j,k, $||H_{ij}(k) - H_{ij}(k+1)|| < \epsilon$, where ϵ is small.

3) To limit memory requirements, we store only a small number of the total possible histogram bins. To avoid a loss of accuracy, we make an assumption that most elements of $H_{ij}(k)$ are 0 : the support of the probability mass function $H_{ij}(k)$ is sparse over $\mathcal{C}$.

4) By starting the algorithm before visitors enter the gallery, we assume that the image sequence contains no foreground objects for the first few seconds : $\exists K > 0$ such that $R(k) = 0\ \forall k < K$.

5) Pixels corresponding to visitors have a color distribution distinct from the background distribution: consider a foreground pixel $I_{ij}(k)$ such that $(i,j) \in F(k)$, has probability mass function $\mathcal{F}_{ij}(k)$. The background distribution at the same pixel is $H_{ij}(k)$. Interpreting distributions as vectors, $||\mathcal{F}_{ij}(k) - H_{ij}(k)|| > \delta$ for some $\delta > 0$. While this property is necessary in order to detect a visitor, it is not sufficient, and we use additional information for classification.

6) Visitors move slowly in the image plane relative to the camera's frame-rate : Formally, assuming $\chi_i(k)$ and $\chi_i(k+1)$ refer to the same foreground object at different times, there is a significant overlap between $\chi_i(k)$ and $\chi_i(k+1)$: $\frac{|\chi_i(k) \cap \chi_i(k+1)|}{|\chi_i(k) \cup \chi_i(k+1)|} > O$, $O \in (0,1)$, where O is close to 1.

7) Visitors move according to a straight-line motion model with Gaussian process noise in the image plane : such a model is used in pedestrian tracking [25] and is used in tracking the location of mobile wireless devices [27]. Further, the model can be interpreted as a rigid body traveling according to Newton's laws of motion. We also assume that the time between each frame is approximately constant, so the Kalman filter system matrices of Section III are constant.

D. Problem Statement

Performance of each algorithm is measured as a function of the number of pixels correctly or incorrectly identified as belonging to the foreground bounding box support, $\overline{F}(k)$. First, tp refers to the number of pixels the algorithm correctly identifies as foreground pixels: $tp(k) = |\overline{F}(k) \bigcap \hat{\mathcal{Z}}(k)|$. fp is the number of pixels incorrectly identified as foreground pixels: $fp(k) = |\overline{F}(k)^c \bigcap \hat{\mathcal{Z}}(k)|$. Finally, fn is the number of pixels identified as background that are actually foreground pixels: $fn(k) = |\overline{F}(k) \bigcap \hat{\mathcal{Z}}(k)^c|$. As in [30], define "precision" as $\mathfrak{p} = \frac{tp}{tp+fp}$ and "recall" as $\mathfrak{r} = \frac{tp}{tp+fn}$. For our interactive installation, recall is more important than precision, so we use the F_2-score [16], [24], a weighted harmonic mean that puts more emphasis on recall than precision:

$$\mathfrak{F}_2 = \frac{5\mathfrak{p}\mathfrak{r}}{4\mathfrak{p} + \mathfrak{r}} \tag{1}$$

The problem is then: for each image $I(k)$ in sequence $\{I\}_{k=0}^{N-1}$, find a collection of foreground pixels $\hat{\mathcal{Z}}(k)$ such that $\mathfrak{F}_2(k)$ is maximized. The optimal value at each time is 1, which corresponds to an algorithm returning precisely the bounding boxes of the true foreground objects: $\hat{\mathcal{Z}}(k) = \overline{F}(k)$. We use Equation 1 to evaluate our algorithm in Section IV.

II. Probabilistic Foreground Segmentation

In this section, we focus on the top row of Figure 1, which takes an image $I(k)$ and generates a set of bounding boxes of possible foreground objects, denoted $\mathcal{M}(k)$. $\hat{\mathcal{Z}}(k)$, the final estimated collection of foreground pixels, is used with $I(k)$ to update the probabilistic background model for time $k+1$.

A. Quantization

We store a histogram $\hat{H}_{ij}(k)$ on RGB color-space for every pixel. $\hat{H}_{ij}(k)$ must be sparse by Assumption I-C3, so the number of exhibited colors is limited to F_{max}, a system parameter. Noise in the camera's electronics, however, spreads the support of the underlying distribution, threatening the sparsity assumption. To mitigate this effect, we quantize the color-space. We perform a linear quantization, given parameter $q < 256$, and interpreting $I_{ij}(k) \in \mathcal{C}$ as a vector, $\hat{I}_{ij}(k) = \lfloor \frac{q}{256} I_{ij}(k) \rfloor$. The floor operation reflects the typecast to integer in software in each color channel. Note that this changes the color-space $\mathcal{C}$ by altering q as indicated in Section I-B.

B. Histogram Initialization

We use the first T frames of video as training data to initialize each pixel's estimated probability mass function, or background model. Interpret the probability mass function $\hat{H}_{ij}(k)$ as a vector in $\mathbb{R}^{q^3}$, where each axis represents a unique color. We define a lifting operation $L : \mathcal{C} \to \mathcal{F} \subset \mathbb{R}^{q^3}$ by generating a unit vector on the axis corresponding to the input color. The set $\mathcal{F}$ is the "feature set," representing all unit vectors in $\mathbb{R}^{q^3}$. Let $f_{ij}(k) = L(\hat{I}_{ij}(k)) \in \mathcal{F}$ be a feature (pixel color) observed at time k. Of the T observed features, select the $F_{tot} \leq F_{max}$ most recently observed unique features; let $\mathbb{I} \subset \{1, 2, \ldots T\}$, where $|\mathbb{I}| = F_{tot}$, be the corresponding time index set. (If $T > F_{max}$, it is possible that F_{tot}, the number of distinct features observed, exceeds the limit F_{max}. In that case, we throw away the oldest observations so $F_{tot} \leq F_{max}$.) Then, we calculate an average to generate the initial histogram: $\hat{H}_{ij}(T) = \frac{1}{F_{tot}} \sum_{r \in \mathbb{I}} f_{ij}(r)$. This puts equal weight, $1/F_{tot}$, in F_{tot} unique bins of the histogram.

C. Bayesian Inference

We use Bayes' Rule to calculate the likelihood of a pixel being classified as foreground (F) or background (B) given the observed feature, $f_{ij}(k)$. To simplify notation, let $p(F|f)$ represent the probability that pixel (i, j) is classified as foreground at time k given feature $f_{ij}(k)$. Using Bayes' rule and the law of total probability,

$$p(B|f) = \frac{p(f|B)p(B)}{p(f|B)p(B) + p(f|F)p(F)}$$

We calculate $p(f|B) = f_{ij}(k)^T \hat{H}_{ij}(k)$, as $\hat{H}_{ij}(k)$ represents the background model. The prior probability that a pixel is foreground is a constant parameter, $p(F)$, a design parameter that affects the sensitivity of the segmentation algorithm. As there are only two labels, $p(B) = 1 - p(F)$. Without a statistical model for the foreground, however, we cannot calculate Bayes' rule explicitly. Making use of Assumption I-C5, we let $p(f|F) = 1 - p(f|B)$, which has the nice property that if $p(f|B) = 1$, then the pixel is certainly identified as background, and if $p(f|B) = 0$, the pixel is certainly identified as foreground. After calculating posterior probabilities for every pixel, the posterior image is $P(k) \in [0, 1]^{w \times h}$ where $P_{ij}(k) = p(F|f_{ij}(k)) = 1 - p(B|f_{ij}(k))$.

D. Filtering and Connected Components

Given the posterior image, $P(k)$, we perform several filtering operations to prepare a binary image for input to the connected components algorithm. We perform a morphological open followed by a morphological close on the posterior image with a circular kernel of radius r, a design parameter, using the notion of morphological operations on greyscale images discussed in [36], [35]. Such morphological operations have been used previously in segmentation tasks [26]. Intuitively, the morphological open operation will reduce the estimated probability of pixels that aren't surrounded by a region of high-probability pixels, smoothing out anomalies. The close operation increases the probability of pixels that are close to regions of high-probability pixels. The two filters together form a sort of smoothing operation, yielding a modified probability image $\breve{P}(k)$.

We apply a threshold with level $\gamma \in (0, 1)$ to $\breve{P}(k)$ to generate a binary image $\mathfrak{P}(k)$. This threshold acts as a decision rule: if $\breve{P}_{ij}(k) \geq \gamma$, $\mathfrak{P}_{ij}(k) = 1$, and otherwise, $\mathfrak{P}_{ij}(k) = 0$, where 1 corresponds to "foreground" and 0 to "background". Then, we perform morphological open and close operations on $\mathfrak{P}_{ij}(k)$; operating on a binary image, these morphological operations have their standard definition. The morphological open operation will remove any foreground region smaller than the circular kernel of radius r', a design parameter. The morphological close operation fills in any region too small for the kernel to fit without overlapping an existing foreground region, connecting adjacent regions.

On the resulting image, the connected components algorithm detects 8-connected regions of pixels labeled as foreground. For this calculation, we make use of OpenCV's `findContours()` function [6] which returns both contours of connected components, used in Section III-B, and the set of bounding boxes around the connected components, denoted $\mathcal{M}(k)$. These bounding boxes are used by the tracking system in Section III, so we represent them as vectors: for $m \in \mathcal{M}(k)$, $m \in \mathbb{R}^4$ with axes representing the x, y coordinates of the center, along with the width and height of the box.

E. Updating the Histogram

The tracking algorithm takes $\mathcal{M}(k)$, the list of detected foreground objects, as input and returns $\hat{\mathcal{Z}}(k)$, the set of pixels identified as foreground. To update the histogram, we make use of feature $f_{ij}(k)$, defined in Section II-B.

First, the histogram $H_{ij}(k)$ is not updated if it corresponds to a foreground pixel: if $(i,j) \in \hat{\mathcal{Z}}(k)$, then $H_{ij}(k+1) = H_{ij}(k)$.

Otherwise, let $\mathcal{S}$ represent the support of the histogram $H_{ij}(k)$, or the set of non-zero bins: $\mathcal{S} = \{x \in \mathcal{F} : x^T H_{ij}(k) \neq 0\} \subset \mathcal{F}$. By the sparsity constraint, $|\mathcal{S}| \leq F_{max}$. If feature $f_{ij}(k)$ has no weight in the histogram ($f_{ij}(k)^T H_{ij}(k) = 0$) and there are too many features in the histogram ($|\mathcal{S}| = F_{max}$), a feature must be removed from the histogram before updating to maintain the sparsity constraint. The feature with minimum weight (one arbitrarily selected in event of a tie) is removed and the histogram is re-normalized. Selecting the minimum: $\mathfrak{f} \in \arg\min_{x \in \mathcal{S}} x^T H_{ij}(k)$. Removing $\mathfrak{f}$ and renormalizing: $H_{ij}(k) = (H_{ij}(k) - (\mathfrak{f}^T H_{ij}(k))\mathfrak{f})/(1 - \mathfrak{f}^T H_{ij}(k))$.

Finally, we update the histogram with the new feature: $H_{ij}(k+1) = (1-\alpha)H_{ij}(k) + \alpha f_{ij}(k)$. The parameter α affects the adaptation rate of the histogram. Given that a particular feature $f \in \mathcal{F}$ was last observed τ frames in the past and had weight ω, the feature will have weight $\omega(1-\alpha)^\tau$. As α gets larger, the past observations are "forgotten" more quickly. This is useful for scenes in which the background may change slowly, as with natural lighting through the course of a day.

III. Multiple Visitor Tracking

Lacking camera calibration, we track foreground visitors in the image plane rather than the ground plane. Once the foreground/background segmentation algorithm returns a set of detected visitors, the challenge is to track the visitors to gather useful state information: their position, velocity, and size in the image plane.

Using Assumption I-C7, we approximate the stochastic dynamical model of a visitor as follows: $z_i(k+1) = Az_i(k) + q_i(k)$, $m_i(k) = Cz_i(k) + r_i(k)$, $q_i(k) \sim \mathcal{N}(0, Q)$, $r_i(k) \sim \mathcal{N}(0, R)$, $R = \sigma I$,

$$A = \begin{bmatrix} A' & 0 & 0 \\ 0 & A' & 0 \\ 0 & 0 & I_2 \end{bmatrix}, \; A' = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \; Q = \begin{bmatrix} Q_x & 0 & 0 \\ 0 & Q_y & 0 \\ 0 & 0 & Q_s \end{bmatrix}$$

where I_2 is a 2-dimensional identity matrix. State vector $z_i(k) \in \mathbb{R}^6$ encodes the x-position, x-velocity, y-position, y-velocity, width, and height of the bounding box respectively, relative to the center of the box. $m_i(k) \in \mathbb{R}^4$ represents the observed bounding box of the object. $Q, R \succ 0$ are the covariances, parameters for the algorithm. Let $\mathbb{Z}(k) = \{z_i(k) : i < Z(k)\}$ be the true states of the $Z(k)$ visitors. Let $\hat{\mathbb{Z}}(k) = \{\hat{z}_i(k) : i < \hat{Z}(k)\}$ be the set of $\hat{Z}(k)$ estimated states. Let $\breve{\mathbb{Z}}(k) = \{\breve{z}_i(k) : i < \breve{Z}(k)\}$ be the set of $\breve{Z}(k)$ predicted states. $\mathcal{M}(k)$ is the set of observed bounding boxes at time k, and $\breve{\mathcal{M}}(k) = \{\breve{m}_i : \breve{m}_i = C\breve{z}_i(k), i < \breve{Z}(k)\}$ is the set of predicted observations.

Given this linear model, and given that observations are correctly matched to the tracks, a Kalman filter bank solves the multiple target tracking problem. In Section III-A, we discuss the matching problem. When observations are not matched with an existing track, a new track must be created in the Kalman filter bank. Given an observation $m \in \mathbb{R}^4$, representing a bounding box, we initialize a new Kalman filter with state $z = (C^T C)^{-1} C^T m$, the pseudo-inverse of $m = Cz$, and initial error covariance $P = C^T RC + Q$. In Section III-B, we discuss criteria for tracks to be deleted. After matching and deleting low confidence tracks, the tracking algorithm has a set of estimated bounding boxes, $\hat{\mathcal{M}}(k) = \{\hat{m}_n = C\hat{z}_n(k) : n < \hat{Z}(k)\}$. The final result must be a set of pixels identified as foreground, $\hat{\mathcal{Z}}(k) \subset \mathfrak{I}$, and we need to convert m_i from vector form to coordinates of the corners of the bounding box to generate $\hat{\mathcal{Z}}(k)$, which is used to evaluate performance at time k in Section IV. Using superscripts to denote elements of a vector, m_n^1 and m_n^2 are the x and y coordinates of the center of the box. m_n^3 and m_n^4 are the width and height. To convert the vector back to a subset of $\mathfrak{I}$, let $m_n^- = (m_n^1 - \frac{m_n^3}{2}, m_n^2 - \frac{m_n^4}{2}) \in \mathfrak{I}$ and $m_n^+ = (m_n^1 + \frac{m_n^3}{2}, m_n^2 + \frac{m_n^4}{2}) \in \mathfrak{I}$. If any coordinate lies outside the limits of $\mathfrak{I}$, we set that coordinate to the closest value within $\mathfrak{I}$, to clip to the image plane. Let $\nu_n = \{(i,j) : m_n^- \leq (i,j) \leq m_n^+\}$. Finally, $\hat{\mathcal{Z}}(k) = \bigcup_{n < \hat{Z}(k)} \nu_n \subset \mathfrak{I}$, the set of pixels within the estimated bounding boxes.

A. Gale-Shapley Matching

Matching observations to tracks makes multiple-target tracking a difficult problem: in its full generality, the problem requires re-computation of the Kalman filter over the entire time history as previously decided matchings may be rejected with the additional information, preventing recursive solutions. To avoid this complexity, sub-optimal solutions are sought. In this section, we describe a greedy, recursive approach that, for a single frame, matches observations to tracks to update the Kalman filter bank.

While some algorithms, e.g. mean-shift [10], use information gathered about the appearance of the foreground object to aid in track matching, our algorithm does not: we assume that individuals are indistinguishable. Here, observation-to-track matching is performed entirely within the context of the probability distribution induced by the Kalman filters. We make use of the Gale-Shapley matching algorithm [13], the solution to the "stable-marriage" problem.

In what follows, we describe the matching problem at time k. Formally, we are given $\mathcal{M}$, the set of detected foreground object bounding boxes, and $\breve{\mathbb{Z}}$, the set of predicted states. Let $|\mathcal{M}| = M$ and $|\breve{\mathbb{Z}}| = Z$. Introduce placeholder sets $\mathcal{M}_\emptyset$ and $\mathbb{Z}_\emptyset$ such that $|\mathcal{M}_\emptyset| = Z$ and $|\mathbb{Z}_\emptyset| = M$. Further, $\mathcal{M} \bigcap \mathcal{M}_\emptyset = \emptyset$ and $\breve{\mathbb{Z}} \bigcap \mathbb{Z}_\emptyset = \emptyset$. These placeholder sets will allow tracks and observations to be unpaired, implying a continuation of a track with a missed observation [33], or the creation of a new track. Define extended sets as $\mathcal{M}^+ = \mathcal{M} \bigcup \mathcal{M}_\emptyset$ and

$\mathbb{Z}^+ = \breve{\mathbb{Z}} \bigcup \mathbb{Z}_\emptyset$. Note that $|\mathcal{M}^+| = |\mathbb{Z}^+|$, a prerequisite for applying the Gale-Shapley algorithm [13]. Let $G \equiv |\mathcal{M}^+|$.

We now describe the preference relation necessary for the Gale-Shapley algorithm. Let $m_i \in \mathcal{M}$ and $\breve{z}_j \in \breve{\mathbb{Z}}$. $\breve{z}_j$ is the predicted state of track j. The Kalman filter estimates an error covariance for the predicted state: $P_j \succ 0$. We are interested in comparing observations, not states, so the estimated error covariance of the predicted observation, $\breve{m}_j = C\breve{z}_j$, is $CP_jC^T + R$, from the linear system described at the start of Section III. The Mahalanobis distance between two observations under this error covariance matrix is

$$d(m_i, \breve{m}_j) = \sqrt{(m_i - \breve{m}_j)^T (CP_jC^T + R)^{-1} (m_i - \breve{m}_j)}$$

To make a preference relation, we exponentially weight the distance: $\omega_{ij} = \exp(-d(m_i, \breve{m}_j))$, $\omega_{ij} \in (0, 1)$. As the distance approaches 0, $\omega_{ij} \to 1$. Making use of Assumption I-C6, we place constraints on the distance: for some threshold $\gamma_{min} \in (0, 1)$, if $\omega_{ij} < \gamma_{min}$ (equiv. the distance is too great), then we deem the matching impossible, by Assumption I-C6. The symmetric preference relation $\phi : \mathcal{M}^+ \times \mathbb{Z}^+ \to \mathbb{R}$ is as follows:

$$\phi(m_i, \breve{z}_j) = \begin{cases} 0 & m_i \in \mathcal{M}_\emptyset \text{ or } \breve{z}_j \in \mathbb{Z}_\emptyset \\ \omega_{ij} & \omega_{ij} \geq \gamma_{min} \\ -1 & \omega_{ij} < \gamma_{min} \end{cases} \tag{2}$$

Equation 2 indicates that if a track $\breve{z}_j$ or observation m_i is to be unpaired, the preference relation between $\breve{z}_j$ and m_i is 0. If the Mahalanobis distance is too large, the preference relation is -1, so not pairing the two is preferred. Otherwise, the preference is precisely the exponentially weighted Mahalanobis distance between the predicted observation $\breve{m}_j$ and m_i.

Then, the Gale-Shapley algorithm with $\mathbb{Z}^+$ as the proposing set pairs each $z \in \mathbb{Z}^+$ with exactly one $m \in \mathcal{M}^+$, resulting in a stable matching. That is, if observation i is paired with track j, and another observation n is paired with track k, if $\omega_{ij} < \omega_{ik}$, then $\omega_{ik} < \omega_{nk}$, so while observation i would benefit from matching with track k, track k would lose, so no re-matching is accepted. Gale and Shapley prove that their algorithm generates a stable matching, and that it is optimal for $\mathbb{Z}^+$ in the sense that, if w_j is the final score associated with $z_j \in \mathbb{Z}^+$ after matching, then $\sum_j \omega_j$ is maximized over the set of all possible stable matchings [13]. Thus, tracks are paired with the best possible candidate observations.

We refer to the final matching as the set $\mathfrak{M} \subset \mathbb{Z}^+ \times \mathcal{M}^+$, where $|\mathfrak{M}| = G$. $\mathfrak{M}$ is the input to the Kalman Filter bank as in Figure 1. Then, each pair $(z, m) \in \mathfrak{M}$ is used to update the Kalman filter bank: depending on the pairing, this creates a new track, or updates an existing track with or without an observation. The Kalman update step generates $\hat{\mathbb{Z}}(k)$ and $\breve{\mathbb{Z}}(k+1)$. $\hat{\mathbb{Z}}(k)$ is used to generate $\hat{\mathcal{M}}(k)$ and $\hat{\mathcal{Z}}(k)$ as described at the beginning of Section III, and $\breve{\mathbb{Z}}(k+1)$ is used as input for the next iteration of the Gale-Shapley Matching algorithm.

B. Heuristic Confidence Model

We employ a heuristic confidence model to discern people from spurious detections such as reflections from skylights. We maintain a confidence level $c_i \in [0, 1]$ for each tracked object $z_i \in \hat{\mathbb{Z}}(k)$, which is a weighted mix of information from the error covariance of the Kalman filter, the size of the object, and the amount of shape deformation of the contour of the object (provided by OpenCV). Typically, undesirable objects are small, move slowly, and have a nearly constant contour.

In the following, we drop the dependence on time k for simplicity and denote time $k+1$, with a superscript $+$.

Consider an estimated state $\hat{z} \in \hat{\mathbb{Z}}$, with error covariance P. Let $c^{dyn} = \exp(-\det(P)/\gamma_{det})$, with parameter γ_{det}. Intuitively, as the determinant of P increases, the region around $\hat{z}$ which is likely to contain the true state expands, implying lower confidence in the estimate. Let $c^{sz} = 1$ if the bounding box width and height are both large enough, $c^{sz} = 0.5$ if one dimension is too small, and $c^{sz} = 0$ if both are too small, relative to parameters $\underline{w}$ and $\underline{h}$ representing the minimum width and height. The third component, c^{sh}, is derived from the Hu moment (using OpenCV functionality), measuring the difference between the contour of the object at time $k-1$ and time k. Let ν_{dyn}, ν_{sz}, ν_{sh} be parameters in $[0, 1]$ such that $\nu_{dyn} + \nu_{sz} + \nu_{sh} = 1$; these are weighting parameters for different components of the confidence model. Then, given a parameter β,

$$c^+ = (1 - \beta)c + \beta(\nu_{dyn}c^{dyn} + \nu_{sz}c^{sz} + \nu_{sh}c^{sh})$$

When a track is first created at time k, $c(k) = 0$. After the first update, if at time $r > k$, $c(r) < \varphi$, another parameter, the track is discarded.

IV. Results

Performance is measured according to precision, $\mathfrak{p}$, recall, $\mathfrak{r}$, and the F_2 measure $\mathfrak{F}_2$, introduced in Section I-D. These are evaluated with respect to manually labeled ground-truth sequences, which determine $\overline{F}(k)$.

We evaluate the performance of our proposed algorithm in comparison with three methods in OpenCV 2.1. We compare our algorithm against tracking algorithms in OpenCV using a nonparametric statistical background model similar to what we propose, `CV_BG_MODEL_FGD` [22]. We compare against three "blob tracking" algorithms, which are tasked with segmentation and tracking: `CCMSPF` (connected component and mean-shift tracking particle-filter collision resolution), `CC` (simple connected components with Kalman Filter tracking), and `MS` (mean-shift). These comparisons, in Figure 3, indicate a significant performance improvement over OpenCV across the board. We also explore the effect of the additional feedback loop we propose, by comparing our "dynamic" segmentation and tracking algorithm with a "static" version, which utilizes only the top row of the block diagram in Figure 1. In the "static" version, the background model is not updated selectively, and no dynamical information is used. Figure 4 illustrates a precision/recall tradeoff. In both comparisons, we see an $\mathfrak{F}_2$ gain similar to the recall gain, so recall is not shown in the former and $\mathfrak{F}_2$ in the latter comparisons, due to space limitations. These and many more comparisons, along with annotated videos of algorithm output, are available at `automation.berkeley.edu/ACC2012Data/`.

In each experiment, the first 120 frames of the given video sequence are used to initialize the background models. Results are filtered with a gaussian window, using 8 points on either side of the datapoint in question. We evaluate performance on three videos. The first is a video sequence called `StationaryVisitors` where three visitors enter the gallery and then stand still for the remainder of the video. Situations where visitors remain still are difficult for all the algorithms. Second is a video sequence called `ThreeVisitors` with three visitors moving about the gallery independently, a typical situation for our installation. Figure 4 illustrates that this task is accomplished well by a statistical segmentation algorithm without any tracking. Third is a video with 13 visitors, some moving about and some standing still, a particularly difficult segmentation task; this is called the `ManyVisitors` sequence.

V. Conclusions

This paper presents a single-camera statistical tracking algorithm and results from our implementation at the Contemporary Jewish Museum installation called "Are We There Yet?". This system worked reliably during museum hours (5-8 hours a day) over the four month duration of the exhibition under highly variable lighting conditions. We would like to extend our analysis and experiment with other datasets. We welcome others to experiment with our data and use the software under a Creative Commons License. Source code and benchmark datasets are freely available and in OpenCV. For details, visit: `automation.berkeley.edu/ACC2012Data/`

In future versions, we'd like to explore automatic parameter adaptation, for example, to determine the prior probabilities in high-traffic zones such as doorways. We would also like to explore how the system can be extended with higher-level logic. For example, we added a module to check the size of the estimated foreground region; when the lights were turned on or off, and too many pixels were identified as foreground, we would refresh the histograms of the background image probability model, allowing the system to recover quickly. A 2-minute video describing the installation is available at `j.mp/awty-video-hd`, and project reviews and documentation are available at `are-we-there-yet.org`.

VI. Acknowledgements

The first author is indebted to the insight of Jon Barron and Dmitry Berenson, the hard work of Hui Peng Hu, Kevin Horowitz, and Tuan Le, and the generous support of Caitlin Marshall and Jerry and Nancy Godbehere. We thank Gil Gershoni, Gilad Gershoni, the staff of the Contemporary Jewish Museum, Meyer Sound, and the organizers of the ACC Special Session on Controls and Art: Amy LaViers, John Baillieul, Naomi Leonard, D'Andrea Raffaello, Cristian Huepe, Magnus Egerstedt, Rodrigo Cadiz, and Marco Colasso.

References

[1] I.P. Alonso, D.F. Llorca, MA Sotelo, L.M. Bergasa, P.R. de Toro, J. Nuevo, M. Ocaña, and M.A.G. Garrido. Combination of feature extraction methods for svm pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):292–307, 2007.

[2] A.A. Argyros and M.I.A. Lourakis. Binocular hand tracking and reconstruction based on 2d shape matching. In *International Conference on Pattern Recognition*, volume 1, pages 207–210. IEEE, 2006.

[3] John Baillieul and Kayhan Ozcimder. The control theory of motion-based communications: Problems in teaching robots to dance. *American Control Conference*, 2012.

[4] S.S. Blackman. Multiple-target tracking with radar applications. *Dedham, MA, Artech House, Inc., 1986, 463 p.*, 1, 1986.

[5] G. R. Bradski and V. Pisarevsky. Intel's computer vision library: Applications in calibration, stereo segmentation, tracking, gesture, face and object recognition". *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2:796–797, 2000.

[6] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[7] E.J. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis. *Arxiv preprint arXiv:0912.3599*, 2009.

[8] T. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, and V. Eruhimov. Computer vision workload analysis: Case study of video surveillance systems. *Intel Technology Journal*, May 2005.

[9] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271–288, 1998.

[10] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, pages 603–619, 2002.

[11] A. Elgammal, R. Duraiswami, D. Harwood, and L.S. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, 2002.

[12] J. Friedman, T. Hastie, and R. Tibshirani. Special invited paper. additive logistic regression: A statistical view of boosting. *Annals of statistics*, pages 337–374, 2000.

[13] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[14] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. *International Conference on Computer Vision*, 99:1–19, 1999.

[15] Cristian Huepe, Rodrigo Cadiz, and Marco Colasso. Generating music from flocking dynamics. *American Control Conference*, 2012.

[16] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 119–126. ACM, 2003.

[17] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proc. European Workshop Advanced Video Based Surveillance Systems*, volume 1. Citeseer, 2001.

[18] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[19] A. LaViers and M. Egerstedt. The ballet automaton: A formal model for human motion. *American Control Conference*, 2011.

[20] Amy LaViers and Magnus Egerstedt. Style based robotic motion. *American Control Conference*, 2012.

[21] Naomi Ehrich Leonard, George Forrest Young, Kelsey Hochgraf, Daniel Swain, Aaron Trippe, Willa Chen, and Susan Marshall. In the dance studio: Analysis of human flocking. *American Control Conference*, 2012.

[22] L. Li, W. Huang, I.Y.H. Gu, and Q. Tian. Foreground object detection from videos containing complex background. In *Proceedings of the eleventh ACM international conference on Multimedia*, page 10. ACM, 2003.

[23] L. Li, W. Huang, I.Y.H. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, 2004.

[24] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):530–549, 2004.

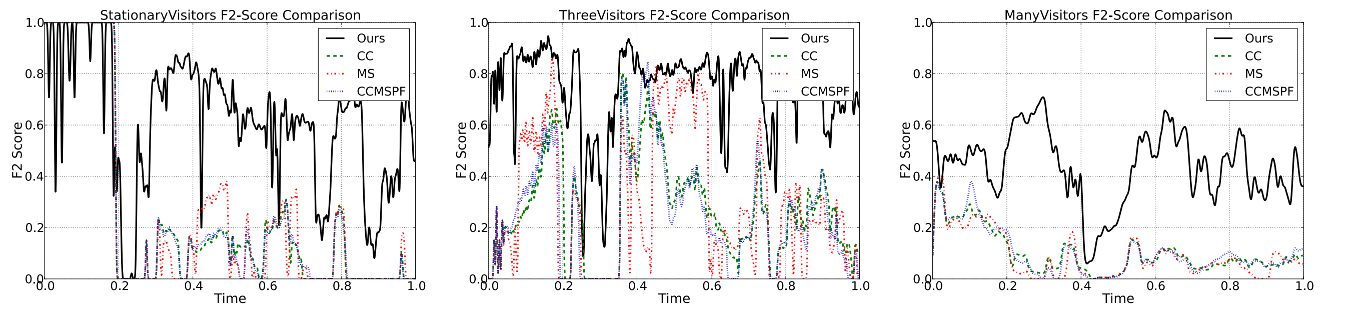

Figure 3. Comparisons with OpenCV. Results indicate a significant improvement in $\mathfrak{F}_2$ score in each case. The recall plots have very similar characteristics and demonstrate our claim of improved recall over other approaches; these plots and more are available on our website. Situations when visitors stand still are a challenge for all algorithms, indicated by drastic drops. When the $\mathfrak{F}_2$ score goes to 0 for OpenCV's algorithms, our algorithm's performance is significantly reduced, although in general, it stays above 0, indicating a better ability to keep track of museum visitors, even when standing still.

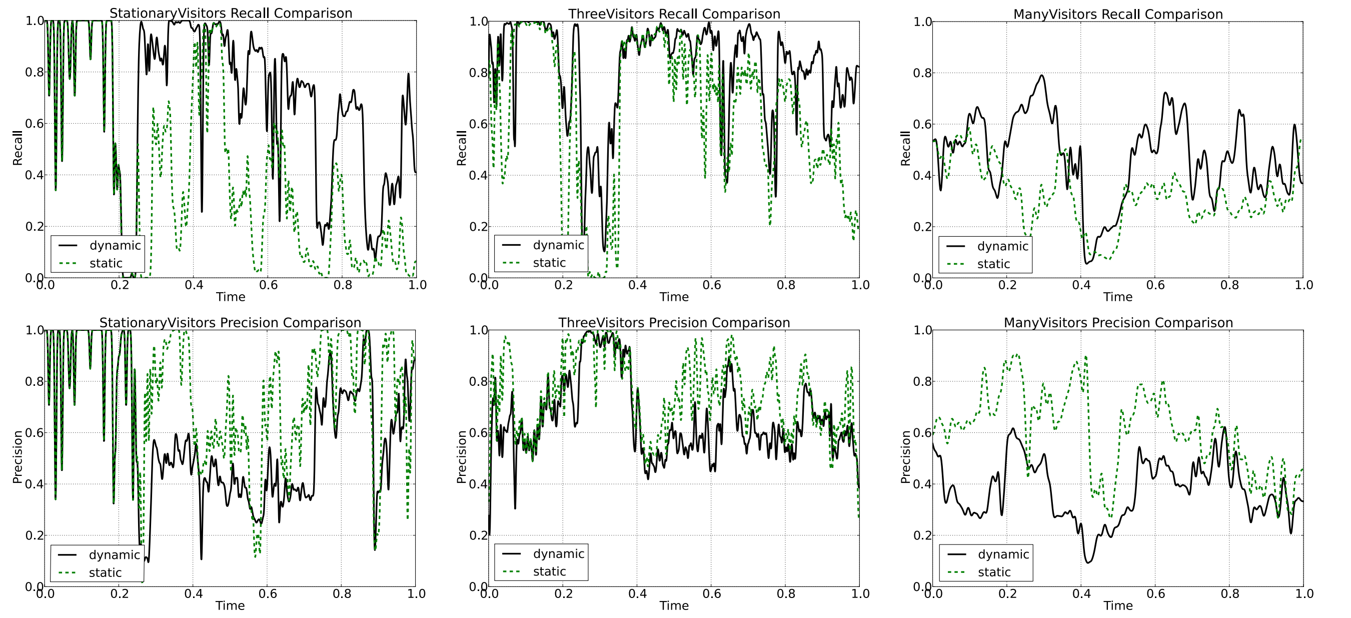

Figure 4. Comparisons between "dynamic" and "static" versions of our algorithm. While the dynamic feedback loop improves the overall $\mathfrak{F}_2$ score, illustrated on our website, we illustrate here that the approach improves recall at the price of precision. The `StationaryVisitors` sequence illustrates the high gains in recall with the dynamic algorithm when visitors stand still. In more extreme cases, as in `ManyVisitors`, this difference is exaggerated. The `ThreeVisitors` sequence shows very similar performance for both algorithms, indicating selectively updated background models are less useful when visitors are continuously moving.

[25] O. Masoud and N.P. Papanikolopoulos. A novel method for tracking and counting pedestrians in real-time using a single camera. *IEEE Transactions on Vehicular Technology*, 50(5):1267–1278, 2002.

[26] F. Meyer and S. Beucher. Morphological segmentation. *Journal of visual communication and image representation*, 1(1):21–46, 1990.

[27] M. Nájar and J Vidal. Kalman tracking for mobile location in nlos situations. *IEEE Proceedings on Personal, Indoor and Mobile Radio Communications*, 3:2203–2207, 2003.

[28] K. Nummiaro, E. Koller-Meier, and L. Van Gool. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, 2003.

[29] K. Okuma, A. Taleghani, N. Freitas, J.J. Little, and D.G. Lowe. A boosted particle filter: Multitarget detection and tracking. *European Conference on Computer Vision*, pages 28–39, 2004.

[30] D.L. Olson and D. Delen. *Advanced data mining techniques*. Springer Verlag, 2008.

[31] P.W. Power and J.A. Schoonees. Understanding background mixture models for foreground segmentation. In *Proceedings Image and Vision Computing New Zealand*, volume 2002. Citeseer, 2002.

[32] Angela Schoellig, Clemens Wiltsche, and Raffaello D'Andrea. Feed-forward parameter identification for precise periodic quadrocopter motions. *American Control Conference*, 2012.

[33] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S.S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, 2004.

[34] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[35] L. Vincent. Morphological area openings and closings for greyscale images. *Proc. NATO Shape in Picture Workshop*, pages 197–208, September 1992.

[36] L. Vincent. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, 1993.

[37] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 2001.

[38] P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[39] C. Yang, R. Duraiswami, and L. Davis. Fast multiple object tracking via a hierarchical particle filter. *International Conference on Computer Vision*, 1, 2005.

[40] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[41] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.