Counting People Crossing a Line Using Integer Programming and Local Features

Zheng Ma, *Member, IEEE*, and Antoni B. Chan, *Senior Member, IEEE*

***Abstract*— We propose an integer programming method for estimating the instantaneous count of pedestrians crossing a line of interest (LOI) in a video sequence. Through a line sampling process, the video is first converted into a temporal slice image. Next, the number of people is estimated in a set of overlapping sliding windows on the temporal slice image, using a regression function that maps from local features to a count. Given that the count in a sliding window is the sum of the instantaneous counts in the corresponding time interval, an integer programming method is proposed to recover the number of pedestrians crossing the LOI in each frame. Integrating over a specific time interval yields the cumulative count of pedestrians crossing the line. Compared with current methods for line counting, our proposed approach achieves state-of-the-art performance on several challenging crowd video data sets.**

***Index Terms*— Crowd counting, integer programming, local feature.**

I. Introduction

THE GOAL of crowd counting is to estimate the number of people in a region of interest (ROI) (ROI counting) or passing through a line of interest (LOI) (LOI counting) in video. Crowd counting has many potential real-world applications, including surveillance (e.g., detecting abnormally large crowds and controlling the number of people in a region), resource management (counting the number of people entering and exiting), and urban planning (identifying the flow rate of people around an area). Beyond people, these counting methods can also be applied to other objects, such as animals passing through a particular boundary, blood cells flowing through a blood vessel under a microscope, and the rate of car traffic. Therefore, crowd counting is a crucial topic in video surveillance and other related fields. However, it is still a challenging task because of several factors: 1) in crowded scenes, occlusion between pedestrians is common, especially for large groups in confined areas and 2) the perspective of the scene causes people to appear larger and move faster when they are close to the camera. These problems are prominent especially in oblique camera views (where the camera looks down at an angle), which are typical of outdoor surveillance cameras.

Manuscript received May 1, 2015; revised July 23, 2015; accepted September 29, 2015. Date of publication October 9, 2015; date of current version September 30, 2016. This work was supported by the Research Grants Council, China, under Grant CityU 110610 and Grant CityU 123212. This paper was recommended by Associate Editor J. M. Martinez.

The authors are with the City University of Hong Kong, Hong Kong (e-mail: mazheng1985@gmail.com; abchan@cityu.edu.hk).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2489418

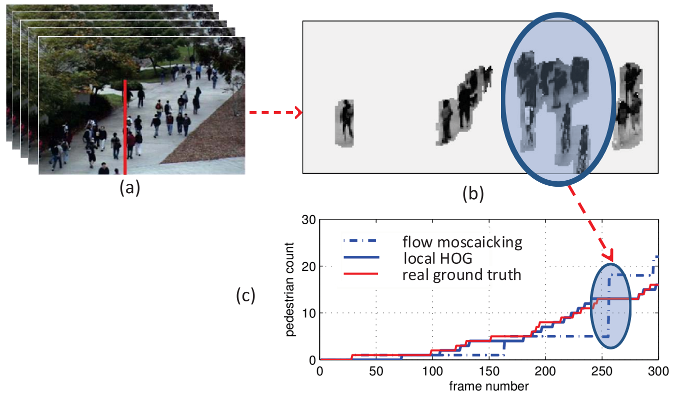


Fig. 1. Line-counting example. (a) Crowd scene and LOI. (b) Temporal slice of the scene. (c) Flow-mosaicking [1] result where a large blob leads to a big jump in the cumulative count. In contrast, our method can predict instantaneous counts better, yielding better cumulative predictions.

Most previous approaches [2]–[6] focus on solving the ROI counting problem and are based on the counting-by-regression framework, where features extracted from the ROI are directly regressed to the number of people. By bypassing intermediate steps, such as people detection, which can be error prone on large crowds with severe occlusion, these counting-by-regression methods achieve accurate counts even on sizable crowds. In this paper, we focus on LOI counting, where the goal is to count the number of people crossing a line (or visual gate) in the video [see Fig. 1(a)]. In particular, the aim is to estimate both the *cumulative* count, i.e., the total count since the start of the video and the *instantaneous* count, i.e., the count at any particular time or short temporal window. The instantaneous count is similar to detecting when a person crosses the line. A naive approach to LOI counting is to apply ROI counting on the regions on each side of the LOI and take the count difference. However, this LOI count will have errors when people enter and exit the ROIs at the same time, since the number of people in the regions remains the same.

Current LOI counting approaches [1] are based on extracting and counting crowd blobs from a temporal slice of the video (e.g., the y-t slice of the video volume). However, there are several drawbacks of these blob-centric methods.

1) Because the blob is not counted until it has completely crossed the line, large blobs (e.g., containing more than 10 people) yield big jumps in the cumulative count, which leads to poor instantaneous count estimates [see Fig. 1(c)].
2) The counts in these large blobs are not accurate due to severe occlusions [1].

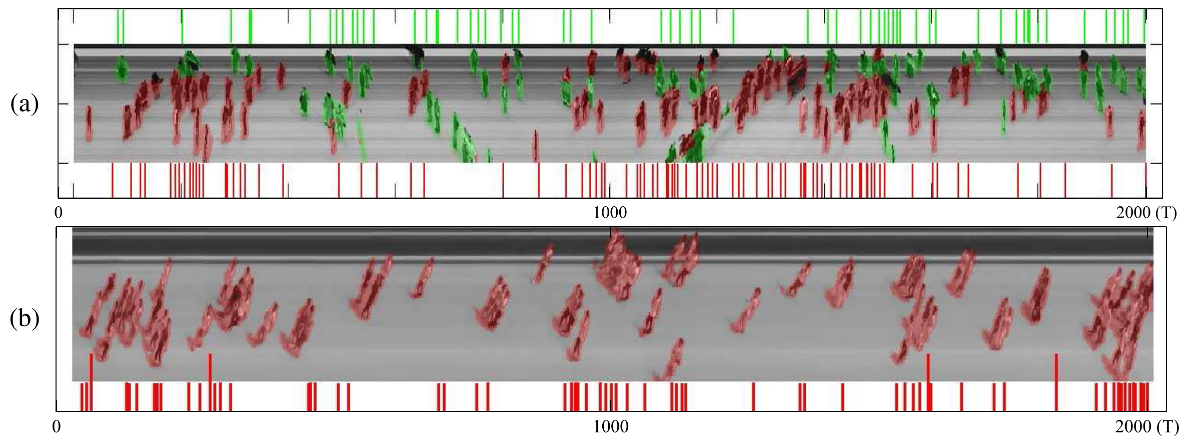


Fig. 2. Results of instantaneous count estimation on (a) UCSD and (b) LHI data sets. The image is a temporal slice of the video on the LOI. The red and green segments correspond to crowds moving in different directions, and the instantaneous count estimates appear above and below the image.

3) Evaluation methods for blob-based methods are based on the ground-truth people in the blob, not the actual people passing the line—hence, it is difficult to assess errors due to segmentation failure of the blob.

Moreover, these methods typically require spatiotemporal (ST) normalization to handle the differences in pedestrian size due to the camera perspective and pedestrian velocity. Current perspective normalization methods [2], [7] require marking a reference person in different positions in the video. For arbitrary videos (e.g., from the Internet), these normalization techniques cannot be applied if no suitable reference exists.

To address the above problems, we propose a novel line-counting algorithm that estimates *instantaneous* people counts using local-level features and regression without perspective normalization (see Fig. 2). The contributions of this paper are threefold. First, to overcome the drawbacks of blob-centric methods, we propose an integer programming approach to estimate the instantaneous counts on the LOI, from a set of ROI counts in the temporal slice image. The cumulative counts of our method are smoother and more accurate than blob-centric methods. Second, we introduce a local histogram-of-oriented-gradients (LHOG) feature, which is robust to the effects of perspective and velocity and yields accurate counts even without ST normalization. Third, we experimentally demonstrate that our method can achieve state-of-the-art results for both cumulative and instantaneous LOI counts on three challenging data sets.

The remainder of this paper is organized as follows. Section II reviews related work in ROI and LOI counting. The line-counting framework based on integer programming is proposed in Section III. Section IV presents the experimental results of our LOI counting framework on synthetic counting data, while Section V validates our framework on three challenging data sets. Finally, Section VI presents detailed experiments on various components of the framework.

II. Related Work

Counting-by-regression methods focus on either counting people in an ROI or counting people passing through an LOI. For ROI counting, features are extracted from each crowd segment in an image, and a regression function maps between the feature space and the number of people in the segment. Typically low-level global features are extracted from the crowd segment, internal edges, and textures [1], [2], [4], [6]. The segment area is a prototypical feature that can indicate the total number of pedestrians in the segment. Reference [2] shows that there is a near linear relationship between the segment area and the number of pedestrian, as long as the feature extraction process properly weights each pixel according to the perspective of the scene. Low-level features can also be extracted from each crowd blob, i.e., an individually connected component in the segment, which contains several pedestrians [5], [6]. Regression methods include Gaussian process regression (GPR) [8] or Bayesian Poisson regression (BPR) [4], which are both kernel methods that can estimate nonlinear functions. Reference [9] introduces a cumulative attribute space for learning a regression model on sparse and imbalanced data. Under the assumption that the source and target data share a similar manifold representation, [10] demonstrates that the lack of labeled data in a new scene can be helped by transferring knowledge from other scenes, thus minimizing the effort required for crowd counting in the new scene. Reference [11] proposes an alternative approach to ROI counting using pixel-wise density learning. The crowd *density* at each pixel is regressed from the feature vector, and the number of pedestrians in an ROI is obtained by integrating over a region. ST group context has also been considered in [12] to further improve the counting performance.

LOI counting estimates the number of people in a temporal slice image (e.g., the y-t slice of the video volume), the result of which represents the number of people passing through the line within that time window. However, with the basic temporal slice, people moving fast will have fewer pixels than those moving slowly, thus confounding the regression function. Flow mosaicking [1] corrects for this by changing the thickness of the line based on the average velocity of the pixels in the crowd blob, resulting in a flow mosaic. The perspective normalization of [7] is used, and the count in each blob is estimated from low-level features. The blob count can only be estimated after the blob has passed the line,

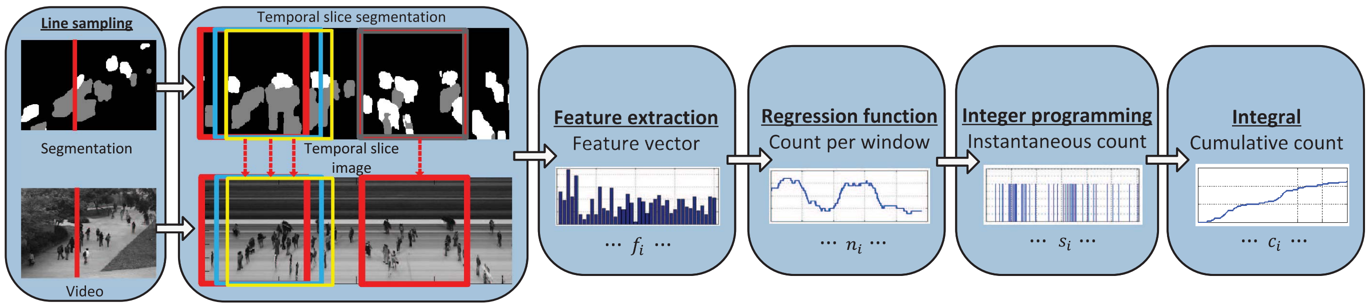


Fig. 3. Proposed line-counting framework. A temporal slice image is formed by sampling on the LOI in a video. Features are extracted from a temporal sliding window, and the number of people in each TROI is estimated using regression. The instantaneous counts on the line are recovered from the TROI counts using integer programming. Finally, the cumulative count is obtained by integrating the instantaneous counts.

and hence large jumps in the cumulative count can occur and instantaneous counts (indicating when each person passes the line) are not possible. In contrast to flow mosaicking [1], our proposed approach performs ROI counting on windows in the temporal slice image and uses integer programming to recover the instantaneous count on the line. In addition, flow mosaicking [1] performs temporal normalization by sampling the LOI using a variable-width line. Because the same line-width must be applied to the whole blob, blobs containing both fast and slow people will not be normalized correctly. In contrast, we use a fixed-width line and do per-pixel temporal normalization, which can better handle large crowd blobs with people moving at different speeds.

Finally, counting can also be performed using people detection methods [13]–[15], which are based on individual-centric features, i.e., features describing the whole person, such as the histogram-of-oriented-gradients (HOG) descriptor of a whole person [13]. The deformable part-based model (DPM) [15] also builds an HOG descriptor of a whole person, using a more flexible layout model for the spatial relationship between HOG parts at different scales. While this results in a model that is better adapted to varying poses of a single person, it can have problems in detecting partially occluded people in groups. In contrast, by removing the layout model, our local HOG representation is better able to handle occlusions.

Visual tracking can also be used for LOI counting. In [16], Kanade-Lucas-Tomasi Feature Tracker (KLT) tracker is used to estimate the tracklets of pedestrians for further crowd behavior analysis. However, the tracking trajectories become noisy and disconnected when the occlusion is high as in crowded scenes (e.g., Grand Central Station).

A preliminary version of our work was first presented in [17]. This paper contains additional improvements in the LOI counting framework and significantly more experimental results:

1) new L1-norm objective function for LOI counting, which improves the processing speed at the cost of a small drop in accuracy for high-density crowds;
2) instead of using one fixed-size ROI temporal window, a new scheme to use multiple window sizes that can improve counting accuracy;
3) new experiments on a synthetic data set, which shows how LOI counting accuracy is affected by crowd density and noisy ROI counts;
4) new large experiment on the Grand Central data set (8000 video frames and eight counting lines);
5) comparisons with other methods of counting, such as DPM pedestrian detection and KLT tracking;
6) in-depth experiments testing different configurations of each component of the framework.

III. Line-Counting Framework

In this section, we introduce our line-counting framework, which is shown in Fig. 3. Given an input video sequence, the video is first segmented into crowds of interest, e.g., corresponding to people moving in different directions. A temporal slice image and temporal slice segmentation are formed by sampling the LOI over time. Next, a sliding window is placed over the temporal slice, forming a set of temporal ROIs (TROIs). Features are extracted from each TROI and the number of people in each TROI is estimated using a regression function. Finally, an integer programming approach is used to recover the instantaneous count from the set of TROI counts. The cumulative counts are obtained by summing the instantaneous count over time.

A. Crowd Segmentation

Motion segmentation is first applied to the video to focus the counting algorithm on different crowds of interest (e.g., moving in opposite directions). We use a motion model [18] of a mixture of dynamic textures to extract the regions with different crowd flows. The video is divided into a set of ST video cubes, from which a mixture of dynamic textures is learned using the Expectation–Maximization algorithm [18]. The motion segmentation is then formed by assigning video patches to the most likely dynamic texture component. Static or very slow moving pedestrians will not be included in the motion segmentation, which is desirable, since the counting algorithm should ignore people who have stopped on the line, in order to avoid double counting.

B. Line Sampling and Temporal ROI

We use line sampling with a fixed line-width to obtain the temporal slice image. As shown in Fig. 3, the input video image and its corresponding segmentation are sampled at the LOI in each frame. Formally, let I_t be the video frame at time t, and $I_t(x, y)$ be the pixel value at location (x, y). The LOI is defined by the y-coordinates of its lower and upper

TABLE I
ST NORMALIZATION FOR LOW-LEVEL FEATURES

Group	Features	Dim.	Weighting strategy
segment features (10)	area	1	$w_p w_v$
	perimeter	1	$\sqrt{w_p w_v}$
	perimeter-area ratio	1	$\sqrt{w_p w_v}$
	perimeter edge orientation	6	$\sqrt{w_p^2 \cos^2\theta + w_v^2 \sin^2\theta}$
	number of blobs	1	N/A
edge features (8)	edge length	1	$\sqrt{w_p w_v}$
	edge orientation	6	$\sqrt{w_p^2 \cos^2\theta + w_v^2 \sin^2\theta}$
	edge Minkowski	1	$\sqrt{w_p w_v}$
texture features (12)	texture homogeneity	4	$\sqrt{w_p w_v}$
	texture energy	4	$\sqrt{w_p w_v}$
	texture entropy	4	$\sqrt{w_p w_v}$

extents $\{y_{lo}, y_{hi}\}$ and its x-coordinate x_L. The sampled image slice at time t is the vector

$$S_t = [I_t(x_L, y_{lo}), I_t(x_L, y_{lo}+1), \ldots, I_t(x_L, y_{hi})]^T. \quad (1)$$

The sampled image slices are collected to form the temporal slice image, where each column in the slice image corresponds to the LOI at a given time, $S = [S_1, S_2, \ldots, S_M]$, where M is the number of frames. Similarly, the corresponding frames in the segmentation are sampled on the LOI to form the temporal slice segmentation. To obtain the TROIs, a sliding window of length L is moved horizontally across the slice image, using a step size of one pixel

$$\text{TROI}_i = [S_i, S_{i+1}, \ldots, S_{i+L-1}], \quad 1 \le i \le M - L + 1. \quad (2)$$

For nonvertical LOIs, we first rotate the input image so that the LOI will be vertical and then perform the line sampling. This removes artifacts in the temporal slice images that are caused when sampling along a pixelated diagonal line.

C. Feature Extraction

Features are extracted from each crowd segment in each TROI. We consider both low-level global and local features.

1) Global Features: We use the 30 global features from [3], which achieved good performance for ROI counting. These features measure various properties of the segment and its internal edges and texture (see Table I). Chan and Vasconcelos [3] demonstrated that there is an almost linear relationship between the number of people and the features like the area of the crowd segment and the length of its internal edges, assuming proper normalization. Local nonlinearities can be modeled with texture features.

2) Local HOG Features: Fig. 4(a) shows an example of a temporal slice image with a crowd walking in two directions. Due to the camera tilt angle, which is nearly 45°, the occlusion of pedestrians is heavy, with torsos or legs not visible in many cases. Rather than using the standard HOGs [13], which is a descriptor of a whole person, we consider a smaller LHOG descriptor that can represent parts of the person independently. As a result, in crowded scenes, meaningful descriptors can still be extracted from partially occluded people.

An LHOG descriptor is calculated from a gray-level square image patch and consists of one block of the standard HOG feature composed of four spatial cells.[1] In each spatial cell, the orientation of the gradient is evenly divided into 9 bins over 0–180° (with unsigned gradient), and the gradient magnitudes are then accumulated into their respective orientation bins, resulting in a 36-D feature (4 cells × 9 bins).[2] Fig. 4(c) presents examples of the LHOG features representing the head–shoulders, side, or legs and feet of people in a crowd [patches in Fig. 4(b)].

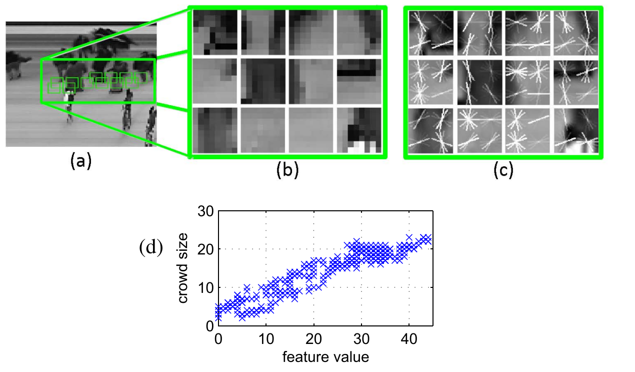


Fig. 4. Example LHOG. (a) Temporal slice image. (b) Image patches. (c) LHOG features. (d) One bin of the BoW histogram versus crowd size.

For each TROI and crowd segment, a set of LHOGs is densely extracted and then summarized into a single feature vector using the bag-of-words (BoW) model. The BoW codewords are the cluster centers resulting from K-means clustering of the LHOGs extracted from the training set. For a given crowd segment, LHOGs are assigned to the closest codewords according to Euclidean distance, and the feature vector is a histogram where each bin represents the number of times an LHOG codeword appears in the crowd segment.

As an example, Fig. 4(d) plots the value of one bin of the histogram versus the number of people in the crowd segment. The bin value varies linearly with the number of people, which suggests that the bag of words of the LHOG can be a suitable feature for crowd counting. Finally, we do not apply histogram normalization methods (e.g., TF and TF-IDF). Normalization will obfuscate the absolute number of codewords in the segment, making histograms from large crowds similar to those from small crowds, which confounds the regression function.

D. Spatiotemporal Normalization

Because the temporal slice image is generated using a fixed-width line, the width of a person will change with its velocity. In particular, people moving slowly across the LOI will appear wider than those moving fast [see Fig. 5(a)]. Hence, temporal normalization is required during feature extraction to adjust for the speed of the person. A temporal weight map $w_v(x, y)$ is formed from the tangent velocity of each LOI pixel, estimated with optical flow[3] [20] [see Fig. 6(b)]. Faster moving people

[1] We also considered rectangular image patches (e.g., 8×16) and found that the 8×8 image patches yield the best performance in the experiments.

[2] We considered weighting the gradient magnitudes using a spatial Gaussian kernel (similar to scale-invariant feature transform [19]), but this did not improve the counting accuracy.

[3] The optical flow on the LOI is computed from two adjacent video frames.

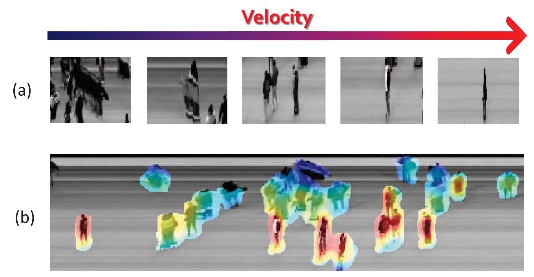


Fig. 5. Temporal slices of pedestrians with different velocities. (a) Slow people (left) have a wide appearance, while fast people (right) have a thin appearance. (b) Tangent velocity of crowd moving though the LOI. The cold colors represent slow people, while the warm colors indicate fast people.

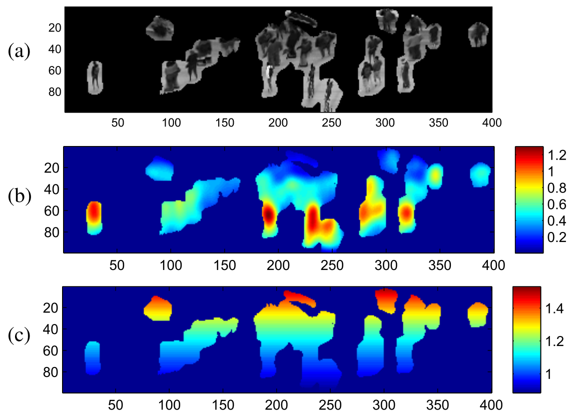


Fig. 6. (a) Temporal slice image. (b) Temporal and (c) spatial weighting maps.

have higher weights, since their features will be present for less time. In addition to the temporal normalization, the features must also be normalized to adjust for perspective effects of the angled camera. We follow [2] to generate the spatial perspective weight map $w_p(x, y)$ [see Fig. 6(c)].

Both weighting maps are applied when extracting low-level features from the image, yielding an ST normalization, summarized in Table I. Specifically, for the area feature, each pixel is weighted by $w_p w_v$, and for edge and texture features, the weighting of $\sqrt{w_p w_v}$ is applied on each pixel. The edge and perimeter orientation features are sensitive to a particular edge angle $\theta \in \{0°, 30°, 60°, 90°, 120°, 150°\}$, and hence a weight of $(w_p^2 \cos^2\theta + w_v^2 \sin^2\theta)^{1/2}$ is used to readjust the contributions between w_v and w_p. For example, for a horizontal edge (90°), only the temporal weight is applied, since there is no component of the edge in the spatial direction.

To normalize LHOG, at each location in the image, we change the size of the image patch by scaling the height and width by w_p and w_v. The extracted image patches are then rescaled to a common reference size (8 × 8). However, normalization of LHOG is not necessary; our experimental results show a similar performance between LHOG with and without ST normalization, which indicates the robustness of the descriptor to perspective and velocity variations.

E. Temporal ROI Count Regression

For each TROI, the count in each crowd segment is predicted using a regression function that directly maps between

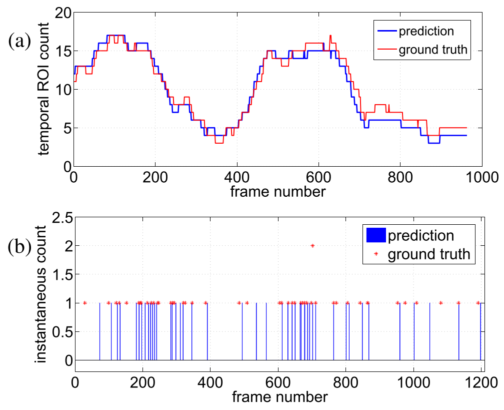


Fig. 7. (a) TROI counts over time. (b) Recovered instantaneous count estimates using integer programming.

the feature vector (input) and the number of people in the crowd segment (output). Since pedestrian counts are discrete non-negative integer values, we use BPR [3], which is an extension of GPR [8], that directly learns a regression function with non-negative integer outputs. BPR models the noisy output of a counting function with a Poisson distribution where the log-mean parameter is a linear function of the input vector. A Gaussian prior is placed on the weights of the linear function, and the model can be kernelized similar to GPR to obtain nonlinear log-mean functions. We use the combination of Radial Basis Function (RBF) and linear kernels, which yielded the best performance compared with the single RBF kernel, linear Bhattacharyya histogram intersection, and Chi-squared-RBF kernels. Fig. 7(a) shows an example of the predicted counts for the TROIs, along with the ground truth.

F. Instantaneous LOI Count Estimation

In the final stage, the instantaneous counts on the LOI are recovered from the TROI counts using an integer programming formulation. The ith TROI spans time i through $i + L - 1$, where L is the width of the TROI. Let $\hat{n}_i$ be the estimated count in the ith TROI, and s_j be the instantaneous count on the LOI at time j. According to the instantaneous counts, the TROI count n_i is the sum of the instantaneous counts s_j within the temporal window (see Fig. 8)

$$n_i = s_i + s_{i+1} + \cdots + s_{i+L-1} = \sum_{k=0}^{L-1} s_{i+k}. \tag{3}$$

Defining the vector of TROI counts $n = [n_1, \ldots, n_N]^T$ and $s = [s_1, \ldots, s_M]^T$, where N is the number of TROIs and M is the number of video frames, we have

$$n = As \tag{4}$$

where $A \in \{0, 1\}^{N \times M}$ is an association matrix with entries

$$a_{ij} = \begin{cases} 1, & j \le i < j + L \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Both the count estimates $\hat{n} = [\hat{n}_1, \ldots, \hat{n}_N]^T$ and A are known, and hence finding s is a signal reconstruction problem, with non-negative integer constraints on the counts s_j. We next consider this reconstruction problem using two error functions.

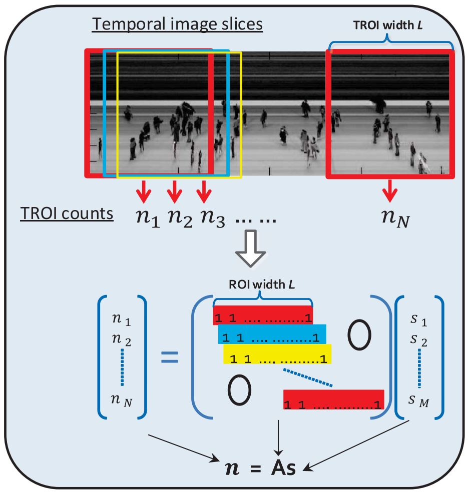


Fig. 8. Relationship between the TROI count n_i and the instantaneous count s_j. The width of the TROI is L, and there are N TROIs.

1) Least Squares Reconstruction Error: We consider recovering the instantaneous counts s using an *integer programming* problem with a sum-squared reconstruction error (L2-norm)

$$s^* = \arg\min_s \ \|As - \hat{n}\|^2 \quad \text{s.t. } s_j \in \mathbb{Z}^+ \quad \forall j \tag{6}$$

where $\mathbb{Z}^+$ is the set of non-negative integers. We solve (6) using the CPLEX optimization toolbox [21]. Fig. 7(b) presents an example of the instantaneous counts recovered from the TROI counts in Fig. 7(a) with integer programming. The predicted instantaneous counts are close to the ground-truth people crossing the line.

2) L1-Norm Reconstruction Error: The L2-norm used in the least squares reconstruction error is known to be prone to large estimation error if there are outliers. In the presence of outliers (e.g., very noisy TROI count estimates), the L1-norm can lead to a more robust estimator

$$s^* = \arg\min_s \|As - \hat{n}\|_1 = \arg\min_s \sum_{i=1}^{N} |a_i s - \hat{n}_i|$$
$$\text{s.t. } s_j \in \mathbb{Z}^+ \quad \forall j \tag{7}$$

where a_i is the ith row of A. The L1 formulation in (7) can be turned into a standard linear integer programming problem (see Supplementary Material), which can be solved with CPLEX [21].

G. Multiple Temporal Window Lengths

The LOI counting framework can be extended to handle TROIs generated with multiple window lengths. Using multiple window lengths can improve the accuracy for line counting, by providing more count measurements over varying window sizes at the same location, which helps to better localize people in large crowds (see the example in Fig. 9).

Let $\mathscr{L} = \{L_1, \ldots, L_K\}$ be a set of window lengths. For each window length L_k, TROIs are extracted from the temporal slice image. The number of people in each TROI is predicted using count regression, resulting in the count vector $\hat{n}^{(k)}$. The association matrix $A^{(k)}$ for length L_k is

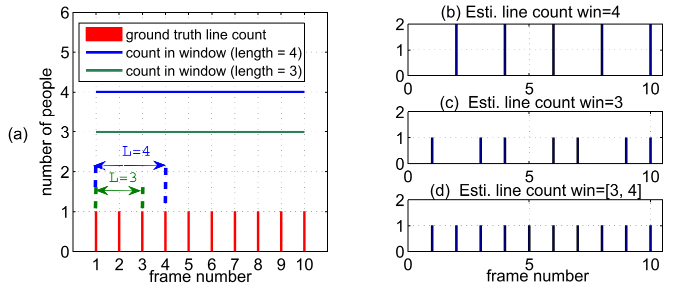


Fig. 9. Example of using multiple window lengths. (a) Large crowd where one person crosses the line in each frame and the TROI counts for window lengths 3 and 4. The instantaneous counts estimated from windows of (b) length 4 or (c) length 3 have errors, while (d) using both windows together yields the correct result.

then formed using (5). To incorporate the multiple windows together, the count vectors and association matrices are concatenated together

$$A = \begin{bmatrix} A^{(1)} \\ \vdots \\ A^{(K)} \end{bmatrix}, \quad \hat{n} = \begin{bmatrix} \hat{n}^{(1)} \\ \vdots \\ \hat{n}^{(K)} \end{bmatrix} \tag{8}$$

and then the instantaneous counts s are obtained by solving the L2 or L1 optimization problems in (6) or (7).

IV. Experiments on Synthetic Data

In this section, we test the ability of our integer programming framework to recover the instantaneous and cumulative counts through experiments on synthetic data.

A. Experiment Setup

The procedure for generating synthetic line counts and TROI counts is seen in Fig. 10. We first generate a synthetic time series of instantaneous line counts. We set the length of the time series to 1200 frames, and 40 random frames are selected to place the instantaneous counts (1 person).[4]

From the ground-truth instantaneous counts, we then generate the ground-truth TROI counts n_i, by summing the instantaneous counts over a temporal sliding window of length $L = 238$. Next, a noisy TROI count $\hat{n}_i$ is produced by adding rounded Wiener noise to each ground-truth TROI count, $\hat{n}_i = n_i + \text{Round}(v_i)$. The random variable v_i is a zero-mean Wiener process, which is simulated as $v_i = v_{i-1} + (\rho/\sqrt{N})\delta_i$, where ρ is the scale factor and $\delta_i \sim \mathcal{N}(0, 1)$. In the following experiments, we randomly select the scale factor $\rho \sim \mathcal{N}(1.5, 1)$ and generate a random noise sequence $\{v_i\}_{i=1}^N$ such that $|E_{\text{ROI}} - 1/N \sum_{i=1}^N |\text{Round}(v_i)|| \le 0.1$, where parameter E_{ROI} is the target TROI noise level. The resulting noisy TROI count will have absolute error (AE) within 0.1 of E_{ROI}. The synthetic TROI counts produced using the rounded Wiener noise tend to be higher or lower than the ground truth for extended periods of time, which is similar

[4]This setting is similar to the University of California, San Diego (UCSD) data set [2], where there are 47 and 40 pedestrians in the test set for the right and left directions, respectively. For the Lotus Hill Institute (LHI) data set [22], there are 44 pedestrians in the test set for the right direction.

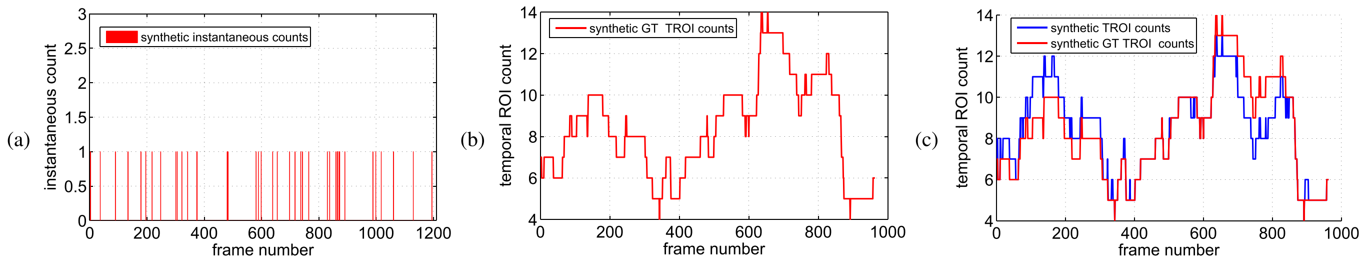


Fig. 10. Generating synthetic counts. (a) Ground-truth instantaneous line counts. (b) Ground-truth TROI counts. (c) Synthetic TROI counts with noise ($E_{\text{ROI}} = 0.7$).

to the errors produced by the actual TROI count prediction [see Fig. 7(a)].

From the noisy TROI counts, the integer programming method in Section III-F is used to recover an estimate of the instantaneous line counts. The cumulative line count is then the sum of the estimated instantaneous line count over time. Let $\hat{c}_{a,b}$ denote the estimated cumulative count between frames a and b, i.e., $\hat{c}_{a,b} = \sum_{t=a}^{b} \hat{s}_t$, where $\hat{s}_t$ is the estimated instantaneous count at time t.

The counting results are evaluated in three ways. First, the cumulative counts from the start of the video are evaluated with the AE between the estimated counts and the ground-truth count, averaged over *all frames*

$$\text{AE} = \frac{1}{M}\sum_{i=1}^{M} |c_{1,i} - \hat{c}_{1,i}| \qquad (9)$$

where $\hat{c}_{1,i}$ and $c_{1,i}$ are the estimated and true cumulative counts between frame 1 and i, respectively, and M is the number of frames. Since AE is based on the overall cumulative counts starting from the beginning of the video, it may give more penalty to errors that occur in the beginning of the sequence than at the end. To mitigate this effect, we also consider the windowed AE (WAE), which is the cumulative counting error within a window of length T,[5] averaged over all windows

$$\text{WAE} = \frac{1}{M-T+1} \sum_{i=1}^{M-T+1} |c_{i,i+T-1} - \hat{c}_{i,i+T-1}| \qquad (10)$$

where the cumulative counts are now over the temporal window spanning frames i to $i+T-1$. When the size of the window is the same as the count sequence length $T = M$, then WAE is the error of the cumulative count in the last frame.

The performance of the instantaneous count prediction is measured using an F-distance curve. The ground-truth instantaneous counts and the predictions are matched pairwise using the Hungarian algorithm to find pairs with minimal temporal distances. An F-distance curve is formed by sweeping a threshold temporal distance d and recording the F-score for the retrieval of pairwise matches with distance less than d. In particular, the precision P is the fraction of predictions that are paired within distance d, the recall R is the fraction of ground-truth instantaneous counts that are paired within distance d, and $F = 2\text{PR}/(P+R)$. The curve represents

[5]T is distinct from the TROI window length L used in Section III-F.

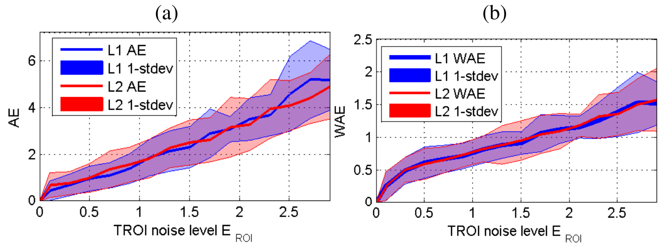


Fig. 11. (a) AE and (b) WAE@100 versus the TROI noise level (E_{ROI}) for L1 and L2 formulations on a synthetic data set. The solid lines show the means along with one standard deviation (shaded).

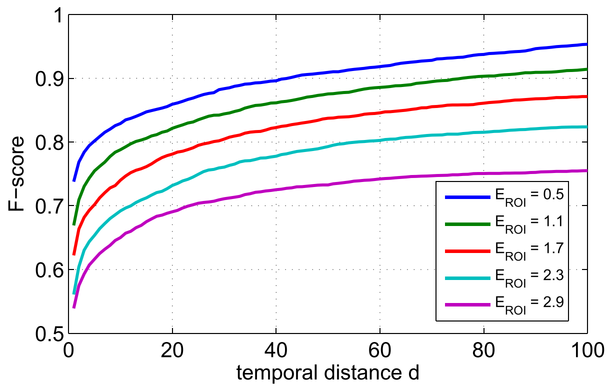


Fig. 12. Average F-distance curve for the instantaneous counting results for different levels of TROI noise (E_{ROI}).

the accuracy (F-score) of detecting a person crossing the line within distance d of the ground-truth crossing. Average errors are reported from 100 random synthetic count sequences.

B. Experimental Results

Fig. 11 plots the AE and the WAE ($T = 100$; denoted by WAE@100) versus the ROI noise level E_{ROI}. This curve describes the relationship between the TROI noise level and the LOI cumulative counting error. For example, when the error in TROI counts (E_{ROI}) is 2.1 people, then the AE for the LOI cumulative count is 3.26 people, while the count error is 1.18 people over windows of length 100 (WAE@100). Empirically, the AE and WAE vary linearly with the TROI noise level, and the L1 and L2 formulations have similar errors. Also note that the cumulative count can be recovered perfectly when no noise is present.

Fig. 12 shows the F-distance curves, which measure the instantaneous counting accuracy, for different TROI noise levels E_{ROI}. When the TROI noise level is $E_{\text{ROI}} = 0.5$, the F-distance curve shows that our method has an F-score

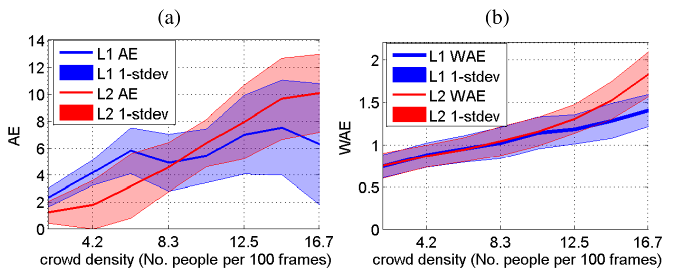


Fig. 13. Comparison of (a) AE and (b) WAE@100 versus crowd density for L1 and L2 formulations on a synthetic data set. The TROI noise level is set to $E_{\text{ROI}} = 0.5$.

of 0.86 for correctly identifying pedestrians crossing the line within $d = 20$ frames (around 2 s for a frame rate of 10 frames/s). The results suggest that integer programming is a possible way to recover LOI counts from TROI counts even from very noisy input TROI counts.

In another experiment, we fix the TROI noise level and vary the crowd density by changing the total number of ground-truth people in a synthetic sequence of length 1200. We set the TROI noise level to $E_{\text{ROI}} = 0.7$. For a given crowd density, 100 synthetic sequences are generated and the L1-norm and L2-norm reconstruction errors are used to recover the instantaneous and cumulative counts. The AE and WAE@100 for different crowd densities are plotted in Fig. 13. In general, as the crowd density increases, the counting error using integer programming also increases. When the crowd density is relatively lower (less than 17 people per 100 frames), the AE of L2-norm is smaller than L1, while WAE is comparable. However, in crowded scenes (more than 25 people per 100 frames), L1-norm achieves better AE and WAE than L2-norm. This is mainly due to overfitting behavior of the L2-norm when there are outliers. The overfitting tends to happen more often when the crowd size is large, since there are more instantaneous counts that can be moved around to reduce the larger residuals between neighboring TROIs.

V. Experiments on Crowd Counting

In this section, we present experiments using the proposed LOI counting algorithm on three crowd data sets.

A. Experiment on UCSD and LHI Data Sets

We first present experiments on two crowd video data sets, the UCSD people counting data set [2] and the LHI pedestrian data set [22]. An example frame from the UCSD data set is shown in Fig. 14(a). The video is captured by a stationary digital camcorder with an angled viewpoint over a walkway at UCSD. The data set contains 2000 video frames (frame size of 238 × 158 at 10 frames/s). The LHI data set contains three types of video, categorized by the camera tilt angle. In our experiments, we use the 3-3 video with a 40° camera tilt angle, which is the most challenging video in LHI due to the large amounts of occlusion. An example frame is displayed in Fig. 14(b), and the frame size is 352 × 288.

1) Experiment Setup: For UCSD, we follow the experimental protocol in [2], where the training set consists of 800 frames (frames 600 to 1399), and the remaining 1200 frames are used as the test set for validation. For LHI, the training set

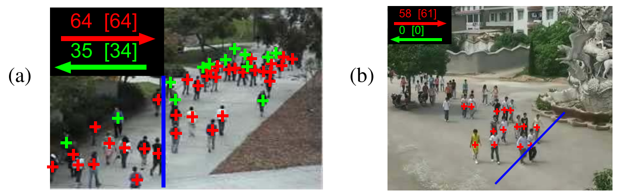


Fig. 14. Examples of input video with LOI for the (a) UCSD data set and (b) video 3-3 of the LHI data set.

is the first 800 frames and the following 1200 frames are the test set. The LOI positions are also shown in Fig. 14. The ground-truth time that each person crossed the LOI was labeled manually. For UCSD, the crowd was separated into two components moving in opposite directions on the walkway (right and left), using the motion segmentation method described in Section III-A. For LHI, the crowd is only moving in the right direction. We estimate the instantaneous and cumulative counts on the LOI using our proposed framework with temporal window length $L = 238$. We also tested multiple windows, $\mathscr{L} = \{50, 100, 150, 200\}$. We use global low-level [2], [3] or LHOG features, with and without ST normalization. The regression model is learned from the training set (UCSD or LHI) and predictions made on the corresponding test set. The hyperparameters of the regression model are estimated automatically by maximizing the marginal likelihood of the training set. All other parameters are fixed for all videos. For comparison, we also predict the cumulative counts using the flow mosaicking [1]. Both methods are run on the same motion segmentation and optical flow images.

We also compared with KLT tracker [23] as a baseline for line counting using standard visual tracking algorithms. The KLT trajectories are locally clustered in each frame, and the number of people crossing the LOI is calculated as the number of trajectories intersecting a bounding box around the line. Note that the KLT tracker does not require training, while our algorithm needs scene-specific training.

The counting results are evaluated with AE (9) and WAE (10). For flow mosaicking, which is blob based and inherently cannot produce smooth cumulative counts, we also consider a blob ground truth that updates only when the predicted count changes, i.e., when a blob is counted. The performance of the instantaneous count prediction is measured using an F-distance curve, as introduced in Section IV-A.

2) Counting Results: The counting results on UCSD and LHI are presented in Table II, with the cumulative and instantaneous counts plotted in Fig. 15.[6] First comparing the different feature sets on the UCSD data set, the LHOG feature achieves comparable results with the global low-level features (AE 0.604 versus 0.534 and WAE@100 0.723 versus 0.793) for the left direction. In the right direction, LHOG obtains significantly less error than the global features (AE 0.6883 versus 1.5067 and WAE@100 0.511 versus 0.703). Since the right direction contains larger crowds, this suggests that LHOG is better at counting the partially occluded people. Furthermore, the counting error with LHOG only increases

[6]Videos of the line-counting results on UCSD and LHI data sets can be found at http://visal.cs.cityu.edu.hk/research/linecount-demo/.

TABLE II
CUMULATIVE COUNTING RESULTS ON UCSD AND LHI DATA SETS. FLOW MOSAICKING IS DENOTED BY FlMsk

ST norm.	Method	Features	UCSD Left		UCSD Right		LHI Right	
			AE	WAE@100	AE	WAE@100	AE	WAE@100
Yes	Ours	LHOG	0.6040	0.7231	**0.6883**	0.5105	**0.8208**	0.8252
		LHOG-mix	0.7220	**0.5621**	0.7245	**0.5017**	0.9020	**0.8240**
		segment	1.2233	0.8647	4.5367	1.2625	1.2608	1.0167
		edge	3.8417	1.3642	1.7517	1.1090	1.5008	1.2521
		segment, edge, texture	**0.5342**	0.7929	1.5067	0.7030	1.0350	0.9201
		LHOG, segment, edge, texture	0.7000	0.8856	0.9600	1.0725	0.9525	0.8547
	FlMsk	area, edge length	1.7233	1.2679	8.2400	2.5876	3.3400	1.7956
	FlMsk (blob GT)	area, edge length	1.3108	-	8.3767	-	2.4058	-
No	Ours	local HOG	**0.6083**	0.7548	**0.7100**	0.5313	**0.8250**	0.8283
		LHOG-mix	0.7358	**0.5886**	0.7383	**0.5231**	0.9028	**0.8238**
		segment, edge, texture	0.9958	1.1580	2.4158	1.2534	1.0625	0.9237
	FlMsk	area, edge length	1.8583	1.4199	11.0108	2.9691	3.5267	1.8610
	FlMsk (blob GT)	area, edge length	1.4458	-	11.1492	-	2.4675	-
	KLT tracker	-	5.7542	1.0300	3.0858	1.1474	5.4958	1.2825

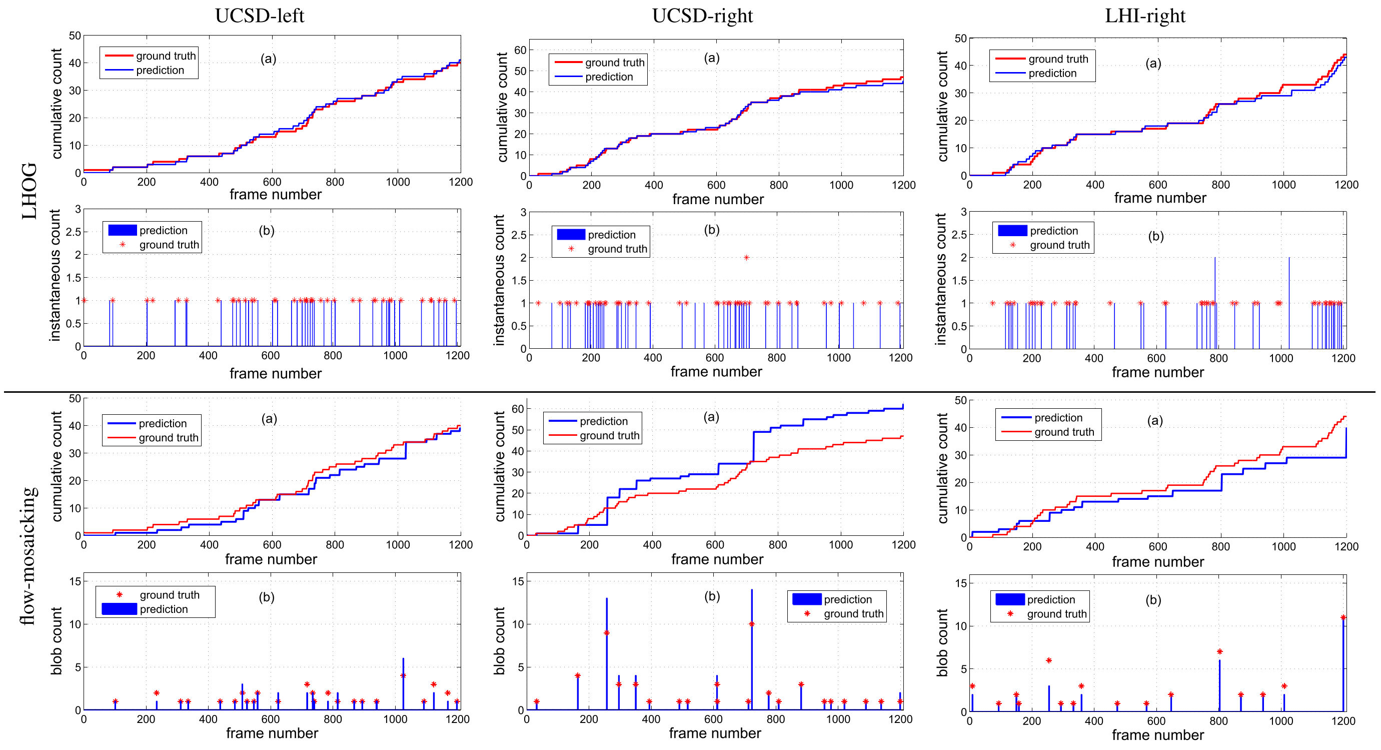


Fig. 15. Counting results on UCSD and LHI data sets using LHOG and integer programming (top) and flow mosaicking (bottom). (a) Cumulative counts. (b) Instantaneous counts for LHOG or blob counts for flow mosaicking.

slightly when ST normalization is not used (increase of <0.03 for AE or WAE on UCSD dataset and <0.005 on LHI dataset). On the other hand, the error for the global features increases significantly, e.g., for the right direction, from 1.507 to 2.416 for AE and from 0.703 to 1.253 for WAE@100. This demonstrates that LHOG is more robust to perspective and velocity effects than the global features. Concatenating the LHOG BoW and global features does not yield to improved performance, possibly due to overfitting or incompatibility of the features. Finally, using multiple windows (denoted by LHOG-mix) can improve the WAE@100 compared with using just a single window, but at the expense of increased AE.

Our LOI counting framework using LHOG has lower AE than the flow-mosaicking method (for both the ground truth and blob ground truth). The flow-mosaicking method has a particularly large error (AE 8.240 and WAE@100 2.588) in the UCSD-right direction. In crowded scenes with large blobs, the flow mosaicking method tends to have high error, which is also shown in the count plots for UCSD-right and LHI-right [see Fig. 15 (bottom)]. Overall, the KLT tracker has lower WAE@100 than the flow-mosaicking one on the UCSD/LHI data sets (average WAE@100 of 1.15 versus 1.19), but also higher AE. KLT-tracker can perform reasonably well in these videos because the pedestrians are large enough for the tracker to find stable features. However, the performance of KLT is still worse than that of our method (WAE of 0.69).

Fig. 16(a) presents the WAE for various temporal window lengths, and Fig. 16(b) shows the corresponding average

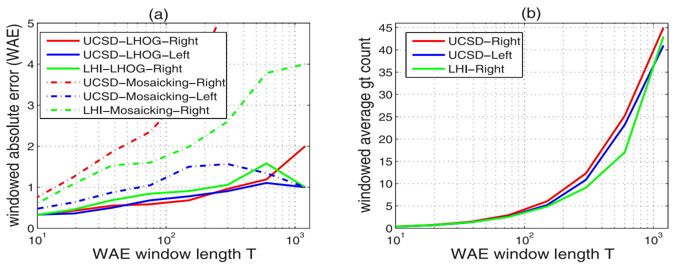


Fig. 16. (a) WAE versus WAE window length T. (b) Average count versus WAE window length T.

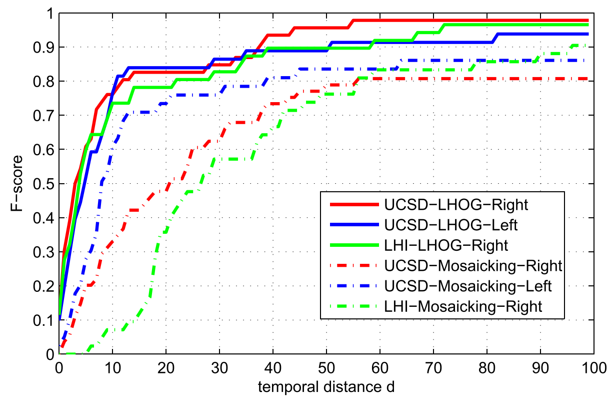


Fig. 17. F-distance curves on UCSD and LHI data sets.

number of ground-truth people. For our method, the WAE is relatively stable regardless of the length of the window evaluated, whereas that of the flow-mosaicking method increases as the window length T and the number of people increases.

The recovered instantaneous counts are presented in Fig. 2, and the accuracy is evaluated using the F-distance curves in Fig. 17. For correctly identifying pedestrians crossing the line within 2 s, our method has F-scores of 0.82, 0.84, and 0.90 on UCSD-right, UCSD-left, and LHI, respectively. For comparison, the flow-mosaicking method has F-scores of 0.48, 0.73, and 0.76. Our method can generate more accurate instantaneous counts than the flow-mosaicking method, which is a blob-centric method.

B. Experiments on Grand Central Data Set

We next present counting experiments on the Grand Central data set from [16]. The video is collected from the inside of the Grand Central Station in New York (see Fig. 18). Compared with the previous outdoor videos (UCSD and LHI), this video is more challenging, since the reflection on the floor and the shadows of people introduce noise that affects the segmentation and introduces noise in the features.

1) Experiment Setup: We define eight LOIs, as shown in Fig. 18, which are labeled L1 to L8 and cover the entrances and exits of the scene. We manually label the ground truth of the first 8000 frames of the video (about 5.3 min at 25 frames/s). The training set for each line consists of 1000 frames, with the other 7000 frames for testing. Since the temporal distribution of people is different for each line, the training sets for each line are selected so that they contain a range of crowd sizes (see Table III). The total number

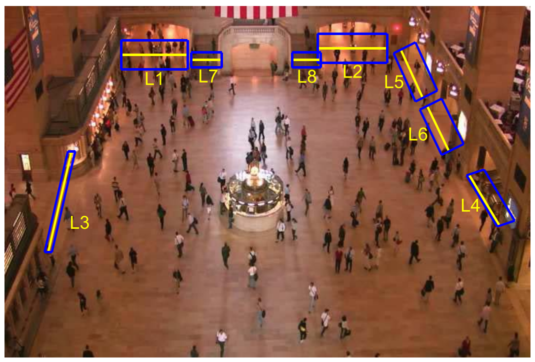


Fig. 18. Frame from the Grand Central data set. The yellow lines represent the LOIs of our algorithm, while the blue rectangle boxes represent the counting areas for the KLT tracker baseline.

TABLE III

TRAINING SET SETTINGS AND NO. OF PEOPLE IN THE GRAND CENTRAL DATASET

Lines	Training frames
L1-L4	2001 to 3000
L5	2001 to 2500, 3001 to 3500
L6	2001 to 2500, 3501 to 4000
L7-L8	4001 to 5000

Data set	Direction	L1	L2	L3	L4	L5	L6	L7	L8
Train set	left/down	37	27	28	3	29	31	1	4
	right/up	29	11	10	60	2	17	9	17
Test set	left/down	184	249	118	32	157	192	7	9
	right/up	209	100	40	484	11	108	54	82
Total	left/down	221	276	146	35	186	223	8	13
	right/up	238	111	50	543	13	125	63	99

of people crossing each LOI and the number of people in training and test data sets are also shown in Table III. The most crowded line is L4, where the right direction contains 543 pedestrians. Since L7-left has only one pedestrian in its training set, the training set of L8 is used as the training set for L7.

For estimating the instantaneous count, we consider one window length, $L = 238$ denoted by LHOG-238, and multiple window lengths. For multiple windows (denoted by LHOG-mix), we use sizes $\mathscr{L} = \{200, 220, 240, 260\}$ for L1 and L2 and $\mathscr{L} = \{50, 100, 150, 200\}$ for L3–L8. L1 and L2 use larger windows than L3–L8 because the people are moving slowly across the line, resulting in stretched bodies in the temporal slice image. Finally, we also estimate the count using the KLT tracking results provided with the Grand Central data set [16].

2) Experimental Results: The cumulative counting results are shown in Table IV, and three representative lines are plotted in Fig. 19 (see the Supplementary Material for all plots). Using multiple windows lengths produced more accurate counts than using a single window length in 14 out of 16 line directions according to WAE@100, and had overall better accuracy averaged over all lines (average WAE@100 of 0.78 versus 0.92).

Counting with KLT has higher average WAE@100 than LHOG-mix (1.47 versus 0.78). The KLT tracker has difficulty tracking the people in lines L1, L2, L7, and L8, because they are far away from the camera, and the people tend to be small

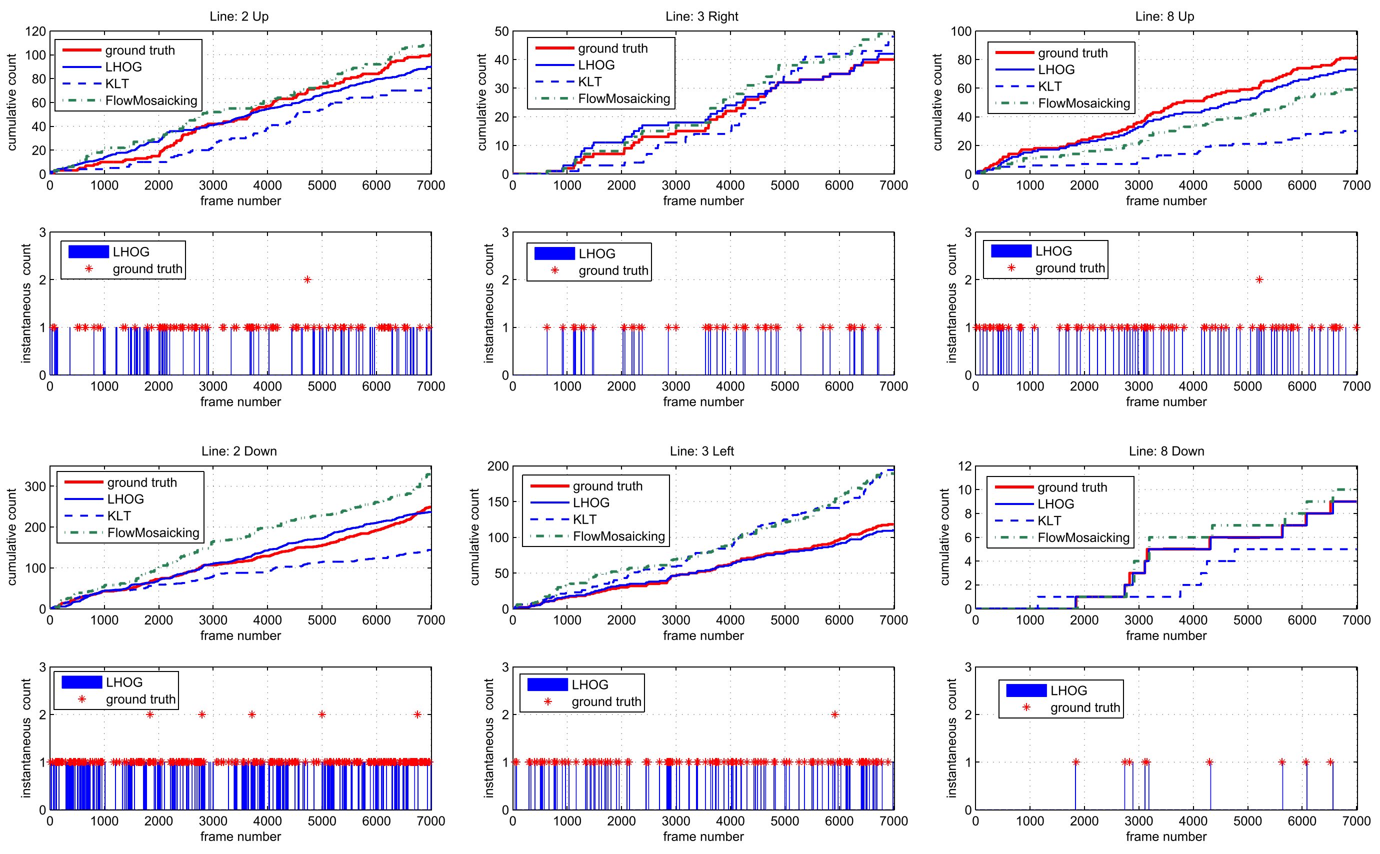


Fig. 19. Line-counting results using LHOG and KLT on the Grand Central data set (lines L2, L3, and L8).

and partially occluded. KLT also has difficulty on L4-right, which is the most crowded line, exhibiting a much higher WAE@100 of 4.0 than LHOG-mix (1.49). Our algorithm also has lower average WAE than the flow-mosaicking method (0.78 versus 1.26). Note that on Grand Central, the flow-mosaicking method performs better than the KLT tracker, most likely because the pedestrians appear smaller on this data set, resulting in more trajectories missed by KLT.

One failure case of LHOG is on L6-left. The temporal distribution of the pedestrian is extremely unbalanced. As a result, the TROI counting function makes more errors, resulting in larger errors in the instantaneous and cumulative count predictions, compared with KLT (AE of 29.1 versus 17.6).

Finally, Fig. 20 plots the F-distance curves averaged over all lines and directions on Grand Central. For our method, the average F-score for detecting people crossing a line within 2 s (50 frames; 25 frames/s) is 0.77, compared with 0.40 for KLT and 0.71 for flow-mosaicking.

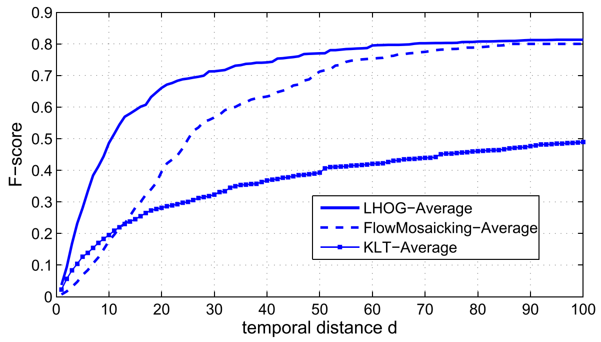


Fig. 20. F-distance curve averaged over all lines on the Grand Central data set.

C. Counting Results Using People Detection Methods

We next test people detection methods for line counting on UCSD and LHI.

1) Setup: We use two people detectors, HOG [13] and DPM [15], to detect and count people in the temporal slice image. The standard detection framework applies a detector with a fixed-size image input to an image pyramid in order to detect people at multiple scales. To adapt the detection framework to work on the temporal slice image, we modify the image pyramid to separately scale the height and the width of the temporal image. The image height is scaled to handle changes in a person's height due to perspective, while the image width (i.e., the temporal dimension) is scaled to handle changes in a person's width due to its velocity (see Fig. 5). In addition, we use vertical LOIs (Fig. 21) so that people in the temporal slice image are not distorted too much. Both detectors are trained on the temporal slice images of UCSD and LHI. The training samples are labeled from the temporal slice image after ST normalization. Finally, the threshold of the detector is learned on the training set, while the counting results are evaluated on the test set.

In addition to detection on the temporal image, we also perform standard people detection in the spatial image around the LOI and apply nonmaximum suppression to

TABLE IV
CUMULATIVE COUNTING RESULTS ON THE GRAND CENTRAL DATA SET

Line	Method	Left/Down		Right/Up	
		AE	WAE@100	AE	WAE@100
L1	KLT	39.7744	2.0406	40.6136	2.1378
	flow-mosaicking	19.3056	1.5476	19.1054	2.112
	LHOG-mix	**8.6735**	**1.0465**	**8.5991**	**1.1594**
	LHOG-238	14.7026	1.3889	11.4042	1.2740
L2	KLT	34.3199	2.5267	14.2189	1.3790
	flow-mosaicking	47.9159	2.6651	7.3389	1.3092
	LHOG-mix	**8.8842**	2.3250	**5.8536**	1.2617
	LHOG-238	31.7000	**2.2869**	10.0820	**1.2253**
L3	KLT	28.5777	2.1442	3.7009	0.9050
	flow-mosaicking	32.3197	1.5799	3.4459	0.2533
	LHOG-mix	**2.5308**	**0.3677**	**1.6722**	**0.1947**
	LHOG-238	6.6725	0.6486	4.0207	0.3618
L4	KLT	5.3343	0.6840	42.0110	4.0090
	flow-mosaicking	**4.539**	0.2601	24.3476	3.9478
	LHOG-mix	6.4384	**0.1679**	11.6508	**1.4915**
	LHOG-238	10.9603	0.3271	**10.0320**	1.7779
L5	KLT	9.7004	1.9336	1.8543	0.2046
	flow-mosaicking	7.8677	2.0706	0.8179	0.0756
	LHOG-mix	8.4251	**0.9796**	0.8010	**0.0727**
	LHOG-238	**7.5983**	1.2323	**0.4550**	0.0952
L6	KLT	17.580	1.8402	16.3181	1.4056
	flow-mosaicking	**16.697**	1.8722	**2.7464**	0.9512
	LHOG-mix	29.1323	**1.7961**	5.8866	**0.6073**
	LHOG-238	30.9304	2.1510	3.9623	0.6132
L7	KLT	2.0357	0.1304	10.9519	0.8576
	flow-mosaicking	0.4587	0.0968	**2.0687**	0.4315
	LHOG-mix	**0.0239**	**0.0484**	3.7773	**0.4042**
	LHOG-238	0.3758	0.0525	10.8120	0.5514
L8	KLT	1.6110	0.1884	28.9231	1.1259
	flow-mosaicking	0.6063	0.1075	12.9126	0.8731
	LHOG-mix	**0.0286**	**0.0580**	**5.0090**	**0.5029**
	LHOG-238	0.1853	0.0949	7.0966	0.6334
Avg	KLT	17.3668	1.4360	19.8240	1.5031
	flow-mosaicking	16.2137	1.2750	9.0979	1.2442
	LHOG-mix	**8.0171**	**0.8487**	**5.4062**	**0.7118**
	LHOG-238	12.8907	1.0228	7.2331	0.8165

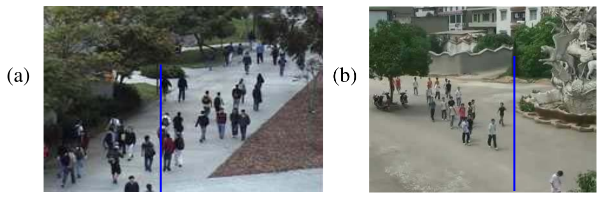


Fig. 21. Line poses for people detection for (a) UCSD and (b) LHI data sets.

obtain the count of people crossing the line (denoted by Histogram of Oriented Gradients (HOG) features with spatial normalization or Deformable Parts Model with spatial normalization (DPM-S)). We also combine the detection maps from the temporal and spatial images to obtain a line count (Histogram of Oriented Gradients (HOG) features with spatio-temporal normalization (HOG-ST) or Deformable Parts Model with spatio-temporal normalization (DPM-ST)).

The x-coordinate of the center of a detection box indicates when a person has passed the LOI and these are collected to form the instantaneous count. To improve the detection results, we use two postprocessing constraints to remove false positive errors. First, we only keep detections whose centers are in the motion segment, in order to remove erroneous detections caused by background clutter. Second, we remove detections that do not fit the perspective geometry of the scene, i.e., those that suggest a person that is too tall or too wide for the given location. Using the crowd motion segments, we obtain the line counts for each direction: right and left and scene (both right and left).

2) Results: The cumulative counting results are presented in Table V (see the Supplementary Material for detection and count plots). On these scenes, DPM obtains a lower cumulative counting error than HOG; the deformable model is better able to handle the distortion of a person's appearance in the temporal slice image. However, the counting-by-detection results have a higher error rate compared with our LHOG regression model.

Deformable Parts Model with temporal normalization (DPM-T) is successful in detecting most people walking alone, while moving at normal speeds (around one pixel per frame). However, the detector has difficulty when the person is moving too fast or too slowly. When a person is moving slowly, its appearance in the temporal slice image will be stretched due to the slow speed and blurred due to the changing pose. When a person is moving too quickly, the appearance will be thin and low quality, because the line sampling process skips slices of the person. Therefore, the appearance of each pedestrian in temporal slice image is not stable, compared with its appearance in a normal image. The two detectors also have difficulty on people walking together in a group, which is due to both partial occlusion between pedestrians and the distortion due to the line sampling process.

Finally, the overall detection results of DPM-S are better than those of DPM-T. However, DPM-S still has a higher error than LHOG regression, except on UCSD Scene where they get similar performance in terms of WAE@100. On average, the combined detection (DPM-ST) has a larger error than only detection on the image. Table VI shows the processing time for the line-counting algorithms on UCSD and LHI. Our framework has a processing time comparable with that of flow-mosaicking (both implemented in MATLAB), and is faster than the people detectors (implemented in C).

VI. EXPERIMENTS ON FRAMEWORK COMPONENTS

In this section, we conduct further in-depth experiments on our proposed line-counting framework.

A. Comparing Framework Components for Line Counting

The experiment results in Section V-A compare our method and the flow-mosaicking [1] at the framework level. These two frameworks use different feature sets, line-sampling and normalization methods, and regression methods. For feature sets, our framework uses LHOG features or 30 global features [2], while flow-mosaicking uses area and edge-length features [1]. To form the temporal slice image, our framework uses fixed-width line sampling in conjunction with ST normalization to adjust for people moving at different speeds. In contrast, the flow-mosaicking uses a variable-width line, with width that adapts to the speed of the blob segment so that people have similar sizes in the temporal slice image; spatial normalization is used to handle perspective. Finally, our framework uses BPR for counting in the TROIs and integer programming to

TABLE V

CUMULATIVE COUNTING RESULTS USING PEOPLE DETECTION ON UCSD AND LHI. *S* AND *T* DENOTE DETECTION IN THE SPATIAL OR TEMPORAL IMAGE

Feature-Method	UCSD Left		UCSD Right		UCSD Scene		LHI Right		Average	
	AE	WAE@100	AE	WAE@100	AE	WAE@100	AE	WAE@100	AE	WAE@100
LHOG regression	**0.608**	**0.755**	**0.710**	**0.531**	**1.848**	1.471	**0.825**	**0.828**	**0.998**	**0.896**
HOG-T detection [13]	4.797	1.767	6.589	2.480	11.384	4.063	5.057	1.812	6.957	2.531
HOG-S detection [13]	7.139	1.413	7.855	1.762	15.377	3.075	5.496	1.281	8.967	1.883
DPM-T detection [15]	2.356	1.128	2.565	1.172	4.908	1.667	4.509	1.728	3.585	1.424
DPM-S detection [15]	1.784	0.833	3.292	1.025	5.506	**1.412**	2.829	0.965	3.353	1.059
DPM-ST detection [15]	3.941	0.831	2.606	1.267	4.431	1.916	5.803	1.254	4.195	1.317

TABLE VI

PROCESSING TIME OF LINE-COUNTING ALGORITHMS ON UCSD AND LHI

Algorithm	Language	Time (ms/frame)
Flow Mosaicking	MATLAB	71.6
KLT-Tracker	C	15
DPM	C	191
HOG	C	180
Ours (LHOG, L2)	MATLAB	102.8
Ours (LHOG, L1)	MATLAB	66.5

recover the instantaneous counts (denoted by BPR + IP). The flow-mosaicking counts people in each blob using quadratic regression.

Here, we compare the performance of individual components within the same counting framework, i.e., one component is changed while the remaining two are fixed. The counting results on UCSD are presented in Table VII. First, for the same feature set (either global, area/edge, or LHOG) and BPR + IP counting, our ST normalization method with fixed-width line sampling is more accurate than using variable-width line sampling (i.e., flow-mosaicking). Because the variable line-width is based on the average speed of the blob, it may distort the people when there are several people moving at different speeds. On the other hand, our ST normalization can better handle this case [see Fig. 6(b)] by effectively applying per-person normalization. Second, for BPR + IP, the global and LHOG features perform better than the area/edge features. Third, using the same feature set and variable-width line sampling, our BPR-IP counting function is more accurate than blob-level regression. Note that we did not test the LHOG features with blob-level regression, since for small blobs, there are too few LHOG patches to build a useful descriptor. In summary, each of the components in our framework individually contributes to the improvement in counting over the one in [1].

B. Comparison of Instantaneous Count Methods

We compare different formulations of recovering the instantaneous counts. First, we investigate the effect of using different output domains when solving the least squares reconstruction of the instantaneous counts (6). In particular, we consider using the output domains of real number (i.e., ordinary least squares), non-negative real numbers, and non-negative integers (integer programming). The counting results of the three approaches are presented in Table VIII (using LHOG features without ST normalization). The integer programming method yields the best result. In practice, for the instantaneous count, we also tend to prefer a non-negative integer value rather than a real value.

We next consider different norms for the reconstruction error, in particular, the L2-norm in (6) and the L1-norm in (7). The test results on UCSD and LHI are presented in Table IX. Averaged over the three data sets, using the L2-norm yields lower AE and WAE@100 than L1-norm. Note that these results are consistent with the results of the synthetic experiments in Fig. 13, since the UCSD and LHI data sets have less than 50 people in the test set. Finally, the average processing time (i7 CPU, 3.40 GHz, 4G memory) needed for the L1 reconstruction (0.91 ms/frame)[7] is about 40 times faster than reconstruction using L2. Hence, with a small loss in performance, L1 reconstruction can be used to decrease the runtime of the line-counting framework.

C. Comparison of Window Lengths

To investigate the performance of multiple temporal windows, we test the performance of using single windows of various lengths and multiple windows consisting of different combinations. Fig. 22 shows the results on L1 and L3 of Grand Central. Compared with only using a single length-50 window, the mixture of multiple windows improves the performance (increases of 9.30%, 11.51%, and 13.81% for sets $\{50, 100\}$, $\{50, 100, 150\}$, and $\{50, 100, 150, 200\}$, respectively).

D. Training Set Size

To analyze the influence of the training set size, we trained the regression model on UCSD and LHI with smaller subsets $\{100, 200, 400\}$ of the original training set, while keeping the test set fixed. Fig. 23 shows the results for different training set sizes. When using half of the original training set, the errors increase by 13.8%/14.7%/1.23% for UCSD-left/UCSD-right/LHI-right. However, even only using 100 frames for training, our results (1.0645/0.9546/0.8792) are still better than the flow-mosaicking and KLT.

E. LHOG Without Spatiotemporal Normalization

Next, we investigate why LHOG can achieve good results without using ST normalization. Fig. 24 shows a plot of the LHOG codewords (without ST normalization) versus the speeds of image patches assigned to those codewords.

[7]The proposed method works in batch mode on a chunk of video (e.g., 1000 frames). Here, we report the average processing time over all frames.

TABLE VII

COUNTING RESULTS ON UCSD FOR DIFFERENT COMBINATIONS OF FEATURES, LINE SAMPLING, NORMALIZATION, AND REGRESSION METHODS

Features	Line-sampling/Normalization	Regression	Left		Right	
			AE	WAE@100	AE	WAE@100
global features	fixed-width/spatio-temporal	BPR+IP	**0.5342**	**0.7929**	**1.5067**	**0.7030**
	variable-width/spatial [1]	BPR+IP	0.9892	0.8274	3.3350	1.2906
	variable-width/spatial [1]	Blob-level	2.8233	1.2498	3.8800	2.4514
area+edge length	fixed-width/spatio-temporal	BPR+IP	**0.9092**	**0.9074**	**1.4942**	**1.5232**
	variable-width/spatial [1]	BPR+IP	1.300	0.9410	6.7583	1.9991
	variable-width/spatial [1]	Blob-level	1.7233	1.2679	8.2400	2.5876
local HOG	fixed-width/spatio-temporal	BPR+IP	**0.6040**	**0.7231**	**0.6883**	**0.5105**
	variable-width/spatial [1]	BPR+IP	1.1958	0.9186	1.6992	1.0781

TABLE VIII

COUNTING RESULTS ON THE UCSD DATA SET WHEN USING DIFFERENT OUTPUT DOMAINS FOR INSTANTANEOUS COUNTS

Output type	Left		Right	
	AE	WAE@100	AE	WAE@100
real numbers	0.8775	0.8746	1.8813	0.7679
non-negative real numbers	0.6120	**0.7406**	1.4307	0.6307
non-negative integers	**0.6083**	0.7548	**0.7100**	**0.5313**

TABLE IX

CUMULATIVE COUNTING RESULTS USING L2-NORM AND L1-NORM ERRORS FOR RECONSTRUCTING INSTANTANEOUS COUNTS

Norm	UCSD Left		UCSD Right		LHI Right		ms/frame
	AE	WAE@100	AE	WAE@100	AE	WAE@100	
L2	0.6083	0.7548	0.7100	0.5313	0.8250	0.8283	37.2
L1	0.6233	1.0400	1.2658	0.7402	0.8325	0.7611	0.91

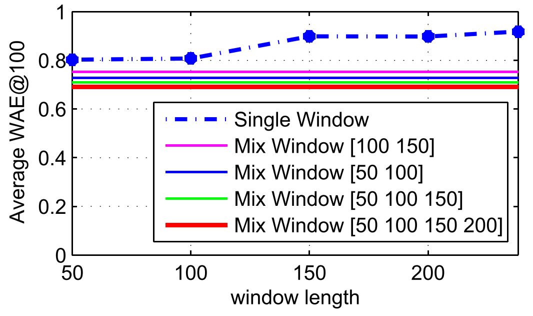


Fig. 22. Average WAE@100 of the left and right directions of L1 and L3 on the Grand Central data set.

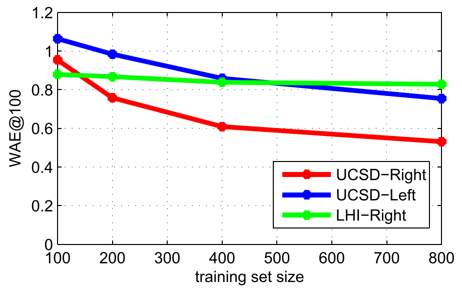


Fig. 23. Counting results on UCSD and LHI versus training set sizes.

Codewords tend to specialize on the appearance of people moving at different speeds. Codewords for slow moving people (e.g., Codeword 1) consist of horizontal edges, as the appearance of a slow-moving pedestrian contains elongated horizontal edges due to sample slices being repeated.

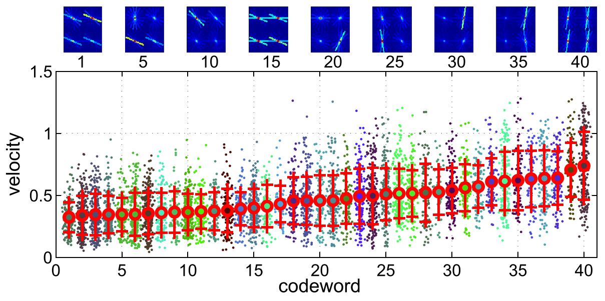


Fig. 24. LHOG codebook learned without ST normalization. The codeword (x-axis) versus the optical flow speed of image patches assigned to that codeword (y-axis). The red circle and the red bar show the average and standard deviation of the speed, respectively. The codeword visualizations are on top.

In contrast, codewords for fast-moving people (e.g., Codeword 40) consist of two vertical edges, which corresponds to the thin appearance of a fast-moving person (see Fig. 5). Hence, the LHOG BoW descriptor without ST normalization is capable of capturing variations in the appearance due to the person's speed, from which a reliable counting function can be learned.

VII. CONCLUSION

In this paper, we have presented a novel line-counting framework, which is based on using integer programming to recover the instantaneous counts on the LOI from TROI counts of a sliding window over the temporal slice image. We validate our framework on three data sets. The results show that compared with global low-level features, the proposed LHOG feature is more robust to the perspective and object velocity variations and performs equally well without using ST normalization. Moreover, compared with blob-centric methods (e.g., flow-mosaicking), our method can generate more accurate instantaneous and cumulative counts, especially in crowded scenes. Further experiments showed that the components in our line-counting framework, in particular fixed-width line sampling with ST normalization, instantaneous counting by integer programming, and LHOG features, each contribute to improving the line-counting accuracy.

There are four potential improvements to be considered for future work. First, the appearance of pedestrians in the temporal slice image becomes distorted during line sampling when using diagonal or horizontal lines. Hence, the features could be made more robust by applying geometric normalization to counteract this distortion. Second, the instantaneous count reconstruction runs in batch mode on all TROI counts. For online estimation, the LOI counts could be obtained by appending the new frame to the previous frames and running the batch method, but this would be inefficient. Efficient online updating of the reconstruction is another topic for future work. Third, devising an automatic method for selecting the best combination of TROI window lengths is an interesting topic for future work. Finally, training of the proposed framework is scene specific since the LHOG and global features are sensitive to the camera viewpoint and LOI orientation. Future work will consider how to transform the LHOG feature when the camera viewpoint changes and how to apply scene transfer algorithms such as those in [10], [12], and [25].

Acknowledgment

The authors would like to thank Y. Cong for the videos from the LHI data set [22].

References

[1] Y. Cong, H. Gong, S.-C. Zhu, and Y. Tang, "Flow mosaicking: Real-time pedestrian counting without scene-specific learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 1093–1100.

[2] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–7.

[3] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and Bayesian regression," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 2160–2177, Apr. 2012.

[4] A. B. Chan and N. Vasconcelos, "Bayesian Poisson regression for crowd counting," in *Proc. IEEE 12th Int. Conf. Comput. Vis. (ICCV)*, Sep./Oct. 2009, pp. 545–551.

[5] D. Kong, D. Gray, and H. Tao, "A viewpoint invariant approach for crowd counting," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, 2006, pp. 1187–1190.

[6] D. Ryan, S. Denman, C. Fookes, and S. Sridharan, "Crowd counting using multiple local features," in *Proc. Digit. Image Comput., Techn. Appl.*, Dec. 2009, pp. 81–88.

[7] F. Lv, T. Zhao, and R. Nevatia, "Camera calibration from video of a walking human," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1513–1518, Sep. 2006.

[8] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[9] K. Chen, S. Gong, T. Xiang, and C. C. Loy, "Cumulative attribute space for age and crowd density estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 2467–2474.

[10] C. C. Loy, S. Gong, and T. Xiang, "From semi-supervised to transfer counting of crowds," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 2256–2263.

[11] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1324–1332.

[12] J. Wang, W. Fu, J. Liu, and H. Lu, "Spatiotemporal group context for pedestrian counting," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 9, pp. 1620–1630, Sep. 2014.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1. Jun. 2005, pp. 886–893.

[14] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors," in *Proc. 10th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 1. Oct. 2005, pp. 90–97.

[15] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.

[16] B. Zhou, X. Wang, and X. Tang, "Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2871–2878.

[17] Z. Ma and A. B. Chan, "Crossing the line: Crowd counting by integer programming with local features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 2539–2546.

[18] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 909–926, May 2008.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[20] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods," *Int. J. Comput. Vis.*, vol. 61, no. 3, pp. 211–231, 2005.

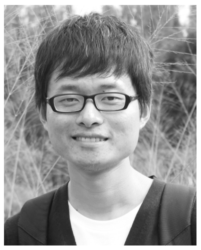
[21] (2013). *IBM ILOG CPLEX Optimizer*. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[22] B. Yao, X. Yang, and S.-C. Zhu, "Introduction to a large-scale general purpose ground truth database: Methodology, annotation tool and benchmarks," in *Proc. 6th Int. Conf. Energy Minimization Methods Comput. Vis. Pattern Recognit. (EMMCVPR)*, 2007, pp. 169–183.

[23] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-91-132, Apr. 1991.

[24] N. Tang, Y.-Y. Lin, M.-F. Weng, and H.-Y. M. Liao, "Cross-camera knowledge transfer for multiview people counting," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 80–93, Jan. 2015.

[25] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 833–841.



Zheng Ma (M'16) received the B.S. and M.S. degrees from Xi'an Jiaotong University, Xi'an, China, in 2007 and 2011, respectively. He is currently pursuing the Ph.D. degree with the City University of Hong Kong, Hong Kong.

His current research interests include computer vision, crowd counting, and object detection.

Antoni B. Chan (SM'15) received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, USA, in 2000 and 2001, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California at San Diego (UCSD), La Jolla, CA, USA, in 2008.

He was a Visiting Scientist with the Vision and Image Analysis Laboratory, Cornell University, from 2001 to 2003, and a Post-Doctoral Researcher with the Statistical Visual Computing Laboratory, UCSD, in 2009. In 2009, he joined the Department of Computer Science, City University of Hong Kong, Hong Kong, where he is currently an Associate Professor. His current research interests include computer vision, machine learning, pattern recognition, and music analysis.

Dr. Chan received the National Science Foundation Integrative Graduate Education and Research Training Fellowship from 2006 to 2008, and the Early Career Award from the Research Grants Council of the Hong Kong Special Administrative Region, China, in 2012.