Small Instance Detection by Integer Programming on Object Density Maps

Zheng Ma Lei Yu Antoni B. Chan
Department of Computer Science
City University of Hong Kong
{zhengma2-c,leiyu6-c}@my.cityu.edu.hk, abchan@cityu.edu.hk

Abstract

We propose a novel object detection framework for partially-occluded small instances, such as pedestrians in low resolution surveillance video, cells under a microscope, flocks of small animals (e.g. birds, fishes), or even tiny insects like honeybees and flies. These scenarios are very challenging for traditional detectors, which are typically trained on individual instances. In our approach, we first estimate the object density map of the input image, and then divide it into local regions. For each region, a sliding window (ROI) is passed over the density map to calculate the instance count within each ROI. 2D integer programming is used to recover the locations of object instances from the set of ROI counts, and the global count estimate of the density map is used as a constraint to regularize the detection performance. Finally, the bounding box for each instance is estimated using the local density map. Compared with current small-instance detection methods, our proposed approach achieves state-of-the-art performance on several challenging datasets including fluorescence microscopy cell images, UCSD pedestrians, small animals and insects.

1. Introduction

Usually, in the computer vision community, object detection and counting are two related but also different research topics. The objective of object detection is to localize individual instances and find their bounding boxes, while object counting aims to estimate the total number of instances. Typically, detection approaches [1, 2] train a sliding window detector for an individual object using a supervised learning algorithm with labeled input images of positive and negative samples. However, such detection approaches tend to be less accurate when objects are overlapping. On the other hand, counting methods [3, 4] tend to do better at estimating the total number of objects in such scenarios, as they model groups of objects locally in a segment [4], or globally in the whole image [3]. However, the output of counting methods is only the number of objects in the image or segment, leaving the locations of each individual object unknown.

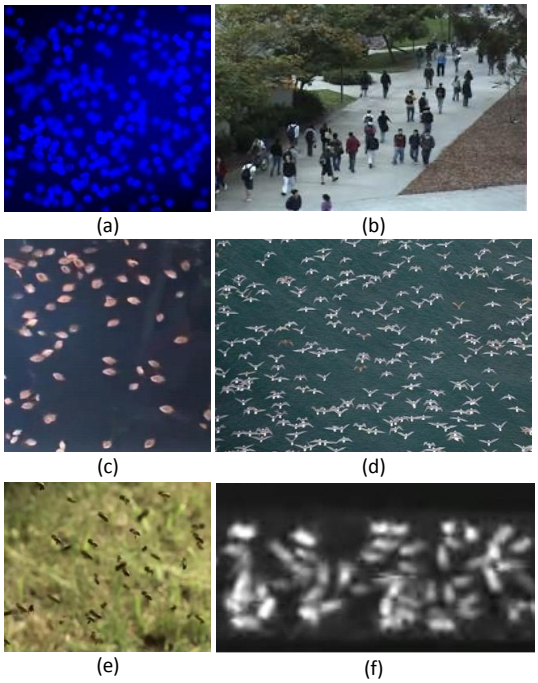


Figure 1. Examples of small object detection: a) synthetic cells; b) pedestrians; c) fish; d) seagulls; e) honeybees; f) flies. Objects are only ∼10-30 pixels tall in the image.

In many real world applications, accurate total counts and locations of objects are equally important. The total number of objects in a region of interest (ROI) can only show a macro-level description of the crowd, which omits the micro-level information for each instance. Usually, the objects are not distributed evenly in an ROI, and the locations of each instance may provide more information for further object behavior analysis. For example, in scenarios such as a crowded street, cells under a microscope, flocks of migrating birds or fishes, and insects moving together, the total count can show a global trend for the whole scene, while the locations of each instance indicate the grouping

structure and the relationship among them. To meet the needs of various vision tasks for crowded scene understanding, solving the counting and detection tasks together in a unified framework can provide more accurate output in terms of both counting and detection.

Most previous detection approaches [1, 2, 5–10] are not suitable for counting-detection task, especially for partially occluded small instances, such as those in Fig. 1. These objects are usually very small in the image or video, perhaps consisting of only a few pixels. A large number of small instances grouped together is neither a pure texture nor a high-resolution object. Compared with a high-resolution object, for groups of small instances, many discriminative details and features are blurred or hidden due to the small object size. Scene perspective makes the problem worse, as the instances with larger distance from the camera tend to be even smaller. For most detection-based methods, the low-resolution training examples of small-instances results in poor detector performance, since there are not enough discriminative features for each individual. Moreover, heavy overlapping between objects moving in a group is very frequent. Although some detection approaches [7–10] aim to handle the occlusion problem by modeling each component of the object, it is challenging to reasonably divide small objects (e.g. tiny flies in a glass tube, small fishes) into discriminative local parts. Finally, different poses and viewpoints make the object's appearance look very different (e.g. flying birds and honeybees). All these factors are challenges of using the traditional detection-based methods for small-instance detection.

Rather than individual-centric detection methods, counting approaches [3, 4, 11] avoid individual-level detection, and instead model all the objects as a whole at the image-level [3, 11] or model small groups of objects at the blob-level [4]. These methods achieved very good results in terms of counting, at the price of losing the local information for each individual. Even the blob-level counting algorithm cannot be used for the detection task, since given the count estimate of a large crowded blob it is still difficult to find the locations of each small instance within it. Recently, [12] achieves state-of-the-art counting performance by estimating an object density map from an input image. The object density map indicates the distribution of the objects within the image, and integrating over an ROI in the density maps yields an estimate of the object count within the ROI.

Inspired by [12], we propose a novel detection framework for small-instances using object density maps, which takes full advantage of counting methods and avoids the potential drawbacks of using traditional detection methods on small-instances. This is a new joint counting and detection framework, which can output both counting and detection results. The contributions of this paper are four-fold. First, we develop a 2D integer programming method that recovers 2D object locations from an object density map. Our framework is unique in that there is no non-maximum suppression or detector threshold to select. Second, to take full advantage of the accurate counting results from [12], a global count constraint is added to the integer programming objective function. The constraint regularizes the detection results by suppressing false positive errors and increasing recall when there is heavy occlusion. Third, given a detected object location, we propose a method to estimate the bounding box of the object using the density map. Last, the proposed detection method achieves state-of-the-art results on several quite different and challenging small-object datasets, including pedestrians, cells, fish, flies, honeybees, and seagulls.

2. Related Work

Most previous methods fall into two classes: individual-centric detection or global/blob-centric counting. In [1, 2, 5, 6] a supervised model is trained with labeled individual object samples, and the learned detector is applied to a sliding window over the image to form the confidence map. Since the detector might fire on the same object twice, non-maximum suppression is typically applied to the confidence map. However, this can decrease recall on crowded scenes when several small objects are close to each other. Beyond non-maximum suppression, [7–10] apply stochastic matching between object parts and instances. These methods require expensive stochastic inference in the image space, and are limited to situations with small numbers of objects. In contrast, our method uses integer programming on the density space, and works well for crowded scenes with large numbers of objects.

Typical counting methods use a regression model or neural network to learn the mapping of global or blob-level features to the number of people in the whole image [3, 11] or blob [4]. These methods focus on object counting but not localization of the objects. [13] combines a head detector, Fourier analysis, and interest point counts to estimate the number of people in image patches of extremely crowded scenes. However, in their framework, the detection method is only used to improve the local patch counting results. [12] estimates the object count using pixel-wise density maps. The crowd density at each pixel is regressed from the feature vector, and the number of objects in an ROI is obtained by integrating over the crowd density map.

The boundary between detection and counting is blurred by [14] (and its precursor [15]), which proposed a unified framework for estimating both the number of objects in a local group and their individual locations. Their approach is based on extremal regions. Low-level features are extracted from these candidate regions, and a structured-output SVM is used to predict the instance counts for the regions (each class corresponds to a count). Given the number of people

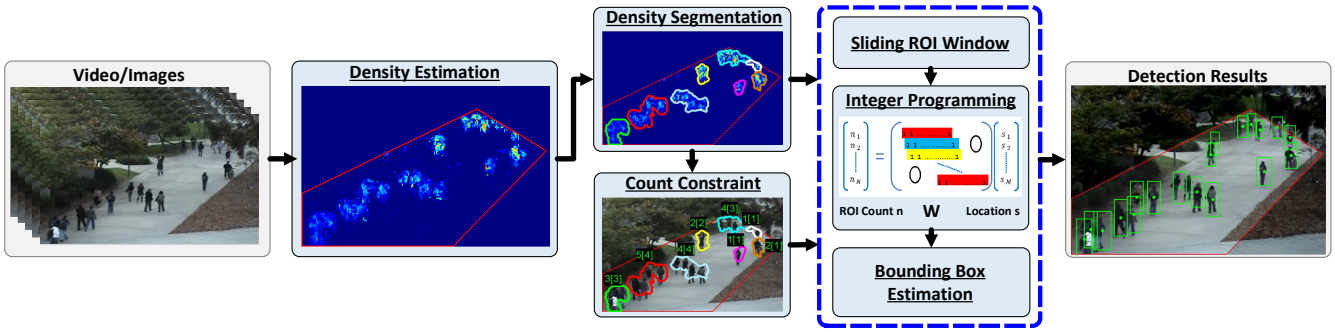
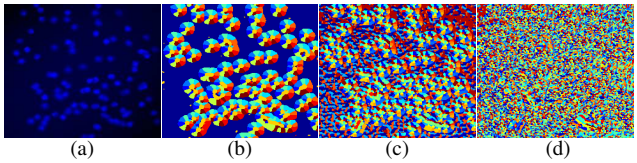


Figure 2. The proposed small-instance object detection framework. The object density map is estimated from the input image or video frame. The density map is divided into several local regions. For each region, the global count is calculated. Then, integer programming with a global count constraint is used to recover the locations of each object from the density map. Finally, the object bounding boxes are estimated from the density map.



(a) (b) (c) (d)

Figure 3. (a) a cell image; (b-d) output features when the codebook size is 10, 40, and 256. The superpixel size depends on the number of codewords, with larger codebooks yielding smaller superpixels.

in a local region, the locations of each individual are estimated using K-means clustering on the image coordinates of all pixels in that region. While the detection performance of [14] is much better than other existing detection methods, the counting performance is not better than [12]. In contrast to [14], which focuses on group-level localization, our approach is formulated as a detection problem for individual instances, and is based on using integer programming to directly recover the individual object locations from the estimated density map. Our method obtains better detection performance than [14], while still maintaining the high counting accuracy of [12]. Also, unlike [14], our approach also estimates the bounding box of each object.

Finally, [16] proposed a method for counting the number of people passing a fixed line in a video – integer programming was used to estimate the number of people in each vertical line of the temporal slice image from the regression-based counts in a large temporal sliding window. In contrast to [16], in this paper we use integer programming to estimate the spatial locations of objects in an image or video frame from an object-density map. We also estimate the bounding boxes of each object.

3. Detection Framework

In this section, we present our proposed object detection framework for small instances, which is illustrated in Fig. 2.

3.1. Feature Extraction

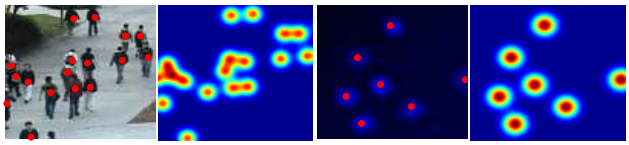
Given the input image, dense pixel-wise features are extracted in a local image patch, which are centered at each image pixel. In this paper, dense SIFT [17] or random forest features [12] (for pedestrians) are used. Sending the feature directly to the density learning process, which learns the mapping of local feature to a density at a single pixel, is not wise, because of the high dimension of the feature vectors (e.g. the length of SIFT is 128). Therefore, following [12], we first do feature quantization. A codebook of K codewords is learned on 20 training images using the K-means algorithm. Then the local features are quantized using the learned codebook. Finally, the feature vector of each pixel is the canonical unit vector $e_j \in \mathbb{R}^K$, where j is the index of the codeword assigned to that location.[1] Thus, all the pixels in a small local region may share the same feature value, resulting in a superpixel segmentation of the input image (e.g., see Fig. 3). The segment sizes depend on the codebook size and the level of local similarity.[2]

3.2. Density Map Estimation

To estimate the object density map, we use the method from [12], which learns the mapping from the extracted image feature to the density value for each pixel. To learn the mapping requires a ground-truth density image. In the training images, the ground-truth object locations are labeled manually as a dot at the center of each object. This type of annotation is more efficient and less labor intensive, compared with rectangular box annotations. Using the ground-truth locations, [12] defines the ground truth density for training image $\mathcal{I}_i$ as (e.g., Fig. 4)

$$F_i^0(p) = \sum_{P \in \mathbf{P}_i} \mathcal{N}(p; P, \sigma^2 I), \forall p \in \mathcal{I}_i \qquad (1)$$

[1] This is equivalent to a bag-of-words representation with 1 sample.
[2] In our experiments, we use a codebook size of $K = 256$.



(a) (b) (c) (d)

Figure 4. Examples of ground truth densities. (a & c) input images with object location annotations; (b & d) the corresponding ground truth densities.

where p is a pixel location of image $\mathcal{I}_i$, $\mathcal{N}(p; P, \sigma^2 I)$ is a 2D Gaussian distribution centered at P, and $\mathbf{P}_i$ is the set of ground-truth object locations in image $\mathcal{I}_i$. The covariance σ^2 is set as a small value (e.g., 1 to 4), which does not significantly affect the performance in practice [18].

The object density at each pixel is estimated by

$$F_i(p; w) = w^T x_p^i, \forall p \in \mathcal{I}_i, \tag{2}$$

where $x_p^i \in \mathbb{R}^K$ is the feature vector for pixel p of image $\mathcal{I}_i$, and $w \in \mathbb{R}^K$ is the parameter vector. Since the feature vectors we use here are the canonical unit vectors, each weight w_j can be interpreted as the density value for codeword j.

The parameter w is learned by minimizing the sum of the mismatches between the ground truth and the estimated density functions over the training set

$$w^* = \underset{w}{\operatorname{argmin}} \|w\|^2 + \lambda_d \sum_{i=1}^{N} \mathcal{D}(F_i^0(\cdot), F_i(\cdot|w)), \tag{3}$$

where λ_d is a hyperparameter controlling the regularization. The function $\mathcal{D}(F_i^0(\cdot), F_i(\cdot|w))$ measures the distance between the estimated density and its corresponding ground truth density. In [12], this distance is selected as the maximum error over all box regions in the image,

$$\mathcal{D}_{\text{MESA}}(F_1, F_2) = \max_{B \in \mathbf{B}} \left| \sum_{p \in B} F_1(p) - \sum_{p \in B} F_2(p) \right|, \tag{4}$$

where $\mathbf{B}$ is the set of all box subarrays of I. After reformulating the loss function properly, the solution turns into a regression problem with w estimated by a QP-solver [19].

3.3. Localization by Integer Programming

We propose a 2D integer programming method to recover the locations of objects in the estimated density map. Given the input density map $F(p; w)$ for an image $\mathcal{I}$, the objective is to find the object indicator map $S(p), \forall p \in \mathcal{I}$, where the value $S(p)$ indicates the number of objects present at location p (e.g., 0 means there is no object at location p, and 1 means there is 1 object at p). Here, we denote the vector $\mathbf{s} \in \mathbb{Z}_+^{|\mathcal{I}|}$ as the vectorized matrix S, where $|\mathcal{I}|$ is the number of pixels in the image $\mathcal{I}$.

Next, we define a set of N sliding windows over the density map. The sliding windows are centered at a pixel in the image, and move at a fixed step vertically or horizontally.[3] The sliding window size is set as the average object size. Each window is represented as a mask vector $\mathbf{w}_i \in \{0, 1\}^{|\mathcal{I}|}$, where the entries are 1 for pixels in the ROI of the window, and 0 otherwise.

Using the density map, the number of objects in the sliding window $\mathbf{w}_i$ is estimated as

$$n_i \approx \mathbf{w}_i^T \mathbf{f}, \tag{5}$$

where the vector $\mathbf{f} \in \mathbb{R}^{|\mathcal{I}|}$ is the vectorized density map $F(p; w)$. On the other hand, the number of objects in the same window according to the object indicator map is

$$n_i = \mathbf{w}_i^T \mathbf{s}. \tag{6}$$

Hence the optimal $\mathbf{s}$ can be obtained by minimizing the difference between (5) and (6) over all the sliding windows. Here we use the L1-norm as the distance measure in order to make the solution robust to outliers. In addition, we also add a global count constraint term to ensure that the total number of detected objects $n_c = \mathbf{1}^T \mathbf{s}$ is close to the number predicted by the density map $n_c \approx \mathbf{1}^T \mathbf{f}$, where $\mathbf{1}$ is the vector of ones. The final optimization problem is

$$\mathbf{s}^* = \underset{\mathbf{s} \in \mathbb{Z}_+^{|\mathcal{I}|}}{\operatorname{argmin}} \sum_{i=1}^{N} |\mathbf{w}_i^T \mathbf{s} - n_i| + \lambda |\mathbf{1}^T \mathbf{s} - n_c| \tag{7}$$

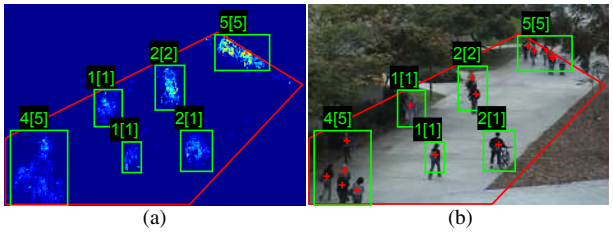
$$= \underset{\mathbf{s} \in \mathbb{Z}_+^{|\mathcal{I}|}}{\operatorname{argmin}} \|\mathbf{W}\mathbf{s} - \mathbf{n}\|_1 + \lambda |\mathbf{1}^T \mathbf{s} - n_c|, \tag{8}$$

where λ is the regularization parameter, and we define a matrix $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_N]^T$ containing all the sliding window mask vectors, and the vector of counts $\mathbf{n} = [n_1, \cdots, n_N]^T$. Note that $\mathbf{n}$ is calculated from the density map and $\mathbf{W}$ is fixed for an image, and hence finding $\mathbf{s}$ is a signal reconstruction problem with non-negative integer constraints on its entries. The formulation in (7) is a linear integer programming problem, which we solve using CPLEX [20].

3.4. Search Space Reduction

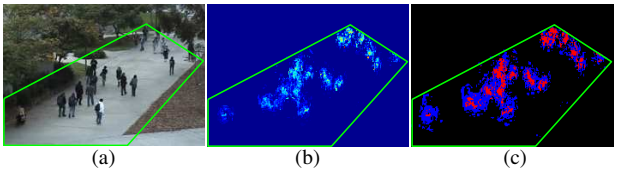
To reduce the search space (number of variables) in the 2D integer programming problem, the input image is divided into several sub-images according to the estimated density map (see Fig. 5). Then, detection is performed on these regions separately. This will significantly reduce the size of matrix $\mathbf{W}$, which depends on the input image size. For each region, we also remove some entries in $\mathbf{s}$ that have low corresponding density values (see Fig. 6), only keeping the locations associated with high-density values as the candidate object locations.

[3] In our implementation, the step size was 2 pixels.



(a) (b)

Figure 5. (a) Local regions of the density map (green boxes). The global count for each region according to the density map and the ground-truth counts (in square brackets) are shown above each region. The global counts are used as constraints to improve the detection results. (b) The corresponding image with ground-truth locations marked as red crosses.



(a) (b) (c)

Figure 6. (a) The input image with ROI (labeled by green line); (b) The estimated density map; (c) The density map is divided into three parts: density > 0.007 (red), $0 <$ density ≤ 0.007 (blue) and density $= 0$ (black). The high-density values make up 26% of the non-zero values, while contributing 60% of the total count, and are distributed in the center of the segmentation.

3.5. Bounding Box Estimation

Once the object locations have been estimated, their bounding boxes are estimated from the density image. Let $\{l_j\}_{j=1}^{D}$ be the estimated locations of the objects. Ideally, for the object at l_j, the optimal bounding box B_j should be placed so that the density map integrates to 1. However, the density map is uneven and noisy, and there are occlusions between objects. To mitigate the effects of this noise, we formulate the cost function for finding the optimal bounding box B_j to include a prior term, which depends on a reference box B_j^0 for location l_j,

$$B_j^* = \underset{B_j}{\operatorname{argmin}} \left\| \sum_{p \in B_j} F(p; w) - C_b \right\| + \lambda_b \Delta(B_j, B_j^0) \quad (9)$$

where $\Delta(B_j, B_j^0)$ measures the total difference between the dimensions of the estimated bounding box and the reference box. Parameter λ_b controls the weight of the prior, and C_b is the target density value, whose typical value is between 1 and 0.8 resulting in a looser or tighter bounding box. The reference box B_j^0 is set as the average ground-truth bounding box at location l_j (see Fig. 7), e.g., by using the perspective information of the scene [3].

4. Experiments

In this section, we test our proposed small-instance object detection framework on six challenging datasets containing quite disparate objects. We compare our detection

(a) (b)

Figure 7. (a) Estimating bounding boxes from the density map. The estimated box is green, the prior box is dashed-red, and the estimated box without using prior information is dashed-blue. (b) The corresponding image.

results with other small-instance detection methods, such as the counting-detection approach from [14] (Region-SVM), and the SIFT-SVM detection baseline introduced in [12].

We also consider five other baseline methods that localize objects using the density map. The first method uses a sliding window (SW) to find regions of the density map that sum to 1. The size of the sliding window is set as the average size of the object. A confidence map is generated where, for each window B, the confidence value is $e^{-|\sum_{p \in B} F(p;w)-1|}$, i.e., windows with densities that sum to 1 have higher confidence. The objects are then detected by applying non-maximum suppression to the confidence map. The second baseline method (LM) assumes each local maximum in the density map indicates the location of an object. In particular, the density map is blurred by a Gaussian kernel, then the local maximum values are captured as the output detection results. The final 3 baseline methods perform clustering on the image coordinates in each density region, and the resulting cluster centers are the object locations. We use K-means (KM), Gaussian mixture models (GMM), and mean-shift (MS) clustering. For KM and GMM, the number of clusters is set using the estimated count on the region.

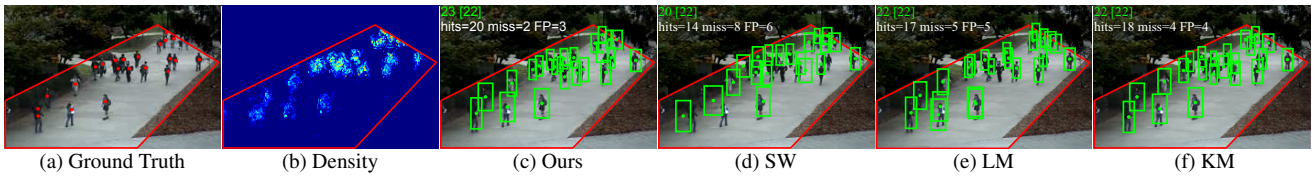
The detection performance is evaluated using precision, recall, and F_1 score as defined in [15]. Precision P is the fraction of detections that are matched with ground-truth locations within matching distance d, recall R is the fraction of ground-truth locations that are paired with detections within matching distance d, and $F_1 = 2PR/(P+R)$. The estimated locations and the ground-truth locations are matched 1-to-1 using the Hungarian algorithm, and the matching distance d is set as [14] (i.e., the minimum object radius). When there is a large perspective change in the image (e.g. UCSD dataset), we weight the matching distance by the perspective map of the scene.

4.1. Pedestrian Dataset

We first test on the UCSD pedestrian dataset [21], which is captured by a stationary digital camcorder with an angled viewpoint over a walkway at UCSD. There are 2000 frames in this video (frame size of 238×158 at 10 fps) supplemented with an approximate perspective map and the re-

Table 1. Detection results on UCSD pedestrian dataset. $\lambda = 0$ means no global count constraints are used.

Method	Split											
	'max'			'downscale'			'upscale'			'min'		
	R%	P%	F_1%	R%	P%	F_1%	R%	P%	F_1%	R%	P%	F_1%
Region-SVM [14]	-	-	89.53	-	-	89.99	-	-	89.21	-	-	86.64
SIFT-SVM [12]	64.16	73.37	68.46	60.97	78.99	68.82	66.57	64.46	65.50	61.76	68.07	64.76
SW / density map	76.69	81.56	79.05	78.19	81.33	79.73	81.48	76.38	78.85	77.07	79.99	78.50
LM / density map	81.01	86.58	83.70	82.85	84.75	83.79	81.18	88.80	84.82	85.39	79.89	82.55
KM / density map	85.34	81.75	83.50	88.07	89.19	88.63	83.21	80.88	82.03	87.98	86.81	87.39
GMM / density map	81.55	78.78	80.14	85.84	86.52	86.18	79.33	77.18	78.24	85.57	84.15	84.85
MS / density map	58.30	78.97	67.08	56.29	89.52	69.12	58.23	79.30	67.15	52.33	91.05	66.47
Ours / density map ($\lambda = 0$)	89.22	94.15	91.62	87.65	95.48	91.40	86.89	94.71	90.63	89.71	92.23	**90.95**
Ours / density map	92.39	91.18	**91.78**	91.35	91.95	**91.65**	90.71	91.07	**90.89**	91.97	89.37	90.65



(a) Ground Truth (b) Density (c) Ours (d) SW (e) LM (f) KM

Figure 8. Examples of detection results on the UCSD dataset using various methods based on the object density map.

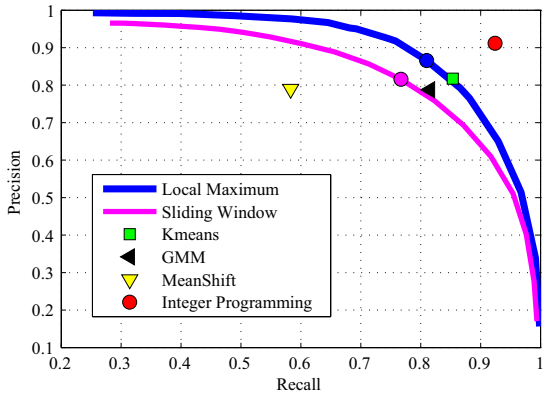


Figure 9. Precision-recall curves of detection approaches using density maps on UCSD ('max' split).

gion of interest. Due to the viewpoint of the camera, there is a large perspective effect in this video. Thus the furthest pedestrians are only a few pixels tall. Moreover, pedestrians walking in a group frequently occlude each other, and the video frames are in low resolution. These challenges make the detection task very hard. Therefore the UCSD dataset is usually used as a benchmark for people counting rather than detection.

We use the four training/testing splits in [4, 12]:

1. 'maximum' – train with 160 frames and test on 1200;
2. 'downscale' – train on the most crowded frames (80 frames) and test on less crowded frames (1600 frames);
3. 'upscale' – train on the less crowded frames (60 frames) and test on more crowded frames (1700);
4. 'minimum' – use only 10 training frames from 'maximum', while testing on the same test set.

The 'downscale' and 'upscale' settings test the generalization of the detection method on crowd levels not seen in the training set, where as the 'max' and 'min' settings test the accuracy and practicality of the method.

Table 2. Detection results on synthetic cell dataset. N denotes the number of training images.

N	Method	R%	P%	F_1%
32	Region-SVM [14]	94.62	94.58	**94.60**
	SIFT-SVM [12]	80.37	85.44	82.83
	SW / density map	71.87	62.76	67.01
	LM / density map	76.47	91.51	83.32
	KM / density map	89.77	94.21	91.93
	GMM / density map	87.62	92.44	89.96
	MS / density map	87.52	81.61	84.46
	Ours / density map ($\lambda = 0$)	92.45	94.57	93.50
	Ours / density map	92.60	94.77	93.67
5	Ours / density map	90.47	93.71	92.06
-	Ours / GT density map	97.20	98.14	**97.67**

The detection results on UCSD are presented in Table 1. Fig. 8 shows example results of different localization methods, and Fig. 9 plots the Precision-Recall curves and points resulting in maximum F1 scores. Our detection F_1 scores on all four splits are higher than Region-SVM [14]. Note that when there are only 10 frames for training ('min' split), [14] has fewer samples to train the structured-SVM, and its F_1 score decreases to 86.64. In contrast, our detection method remains at high-level of 90.95. Finally, except for the 'min' split where the estimated density map is noisier, adding the global count constraint to the objective function increases F_1. In particular, the recall is increased at the expense of some precision – when the count is low, the constraint encourages detections when the density map is noisy.

4.2. Cell Dataset

The synthetic cell dataset [12] includes 200 cell images with an average number of 171 ± 64 cells per image. Partial cell occlusion and image saturation, which are very com-

Table 3. Detection results on small object datasets. The large/small training sets have 32/5 images for {Fly, Honeybee, Fish}, and 1/$\frac{1}{4}$ image for Seagull.

Training set	Method	Fly			Honeybee			Fish			Seagull		
		R%	P%	F_1%	R%	P%	F_1%	R%	P%	F_1%	R%	P%	F_1%
Large	SIFT-SVM [12]	74.55	85.69	79.73	78.99	84.31	81.57	89.82	87.27	88.53	93.86	92.61	93.23
	SW / density map	64.67	62.58	63.61	50.74	48.74	49.72	76.42	65.74	70.68	75.60	71.50	73.49
	LM / density map	85.24	83.82	84.53	87.45	68.72	76.96	88.18	93.68	90.85	85.26	90.33	87.73
	KM / density map	64.77	95.20	77.09	77.38	76.41	76.89	81.28	98.74	89.16	87.07	80.39	83.60
	GMM / density map	64.33	94.55	76.57	87.85	63.24	73.54	80.70	98.80	88.84	86.83	80.17	83.37
	MS / density map	65.92	79.90	72.24	76.91	74.27	75.57	90.83	94.22	92.50	85.49	80.58	82.96
	Ours / density map ($\lambda = 0$)	81.39	93.71	87.12	78.39	91.39	84.39	91.73	92.34	92.03	91.51	96.78	94.07
	Ours / density map	82.47	93.34	**87.57**	78.72	92.14	**84.91**	92.11	95.12	**93.59**	91.47	98.88	**95.03**
Small	SIFT-SVM [12]	38.83	37.06	37.92	12.08	58.25	20.01	90.72	82.93	86.65	72.21	98.72	83.41
	Ours / density map	78.18	94.52	**85.58**	81.68	86.62	**84.08**	81.91	94.37	**87.70**	80.22	98.92	**88.59**

Table 4. Evaluation of detection bounding boxes in terms of F_1%.

Method	Fly	Honeybee	Fish	Seagull
SIFT-SVM [12]	79.71	64.45	87.61	92.74
Ours	**86.37**	**81.65**	**90.88**	**92.84**

mon for fluorescence cell microscopy images, blur the cell boundaries resulting in several single cells merging into one group. Following [12], the first 100 cell images are used for training, while the second 100 images are for testing. In this experiment, the density map function is trained on the first 32 or the first 5 training images. The results are shown in Table 2. Our results are comparable to (but slightly worse than) the Region-SVM [14]. In addition, the F_1 score remains high (92) when only using 5 training images.

When using the ground-truth density, the F_1 for our detector is 97.67, which upper bounds our detection framework. The difference between detection scores when using the ground-truth and estimated density maps suggests that the estimated density maps for cell images are somewhat noisy. Improving the density map estimator would improve our detection results. Finally, from the viewpoint of a joint detection-counting framework, counting with the density map [12] is more accurate than [14].

We also test our method on a real cell dataset [22], which includes 6 embryonic cell fluorescence images (500×500 pixels). The F_1 score for our algorithm is 87.09, compared to 87 for Region-SVM [14] and 86 for [22].

4.3. Other Small Object Datasets

We collect four datasets of small objects from images/videos on the Internet (e.g.YouTube or Google).

Fly Dataset: contains 600 video frames with an average of 86 ± 39 flies per frame (648×72 @ 30 fps). 32 images are used for training (1:6:187) and 50 images for testing (301:6:600).

Honeybee Dataset: contains 118 images with an average of 28 ± 6 honeybees per image (640×480). The dataset is divided evenly for training and test sets. Only the first 32 images are used for training.

Fish Dataset: contains 387 frames of video with an average of 56 ± 9 fish per frame (300×410 @ 30 fps). 32 images are used for training (1:3:94) and 65 for testing (193:3:387).

Seagull Dataset: contains three high-resolution images (624×964) with an average of 866 ± 107 seagulls per image. The first image is used for training, and the rest for testing.

Examples of the detection results are shown in Fig. 10. The detection results of our proposed method and SIFT-SVM [12] are presented in Table 3. On all four datasets, our detection results are better than SIFT-SVM. In addition, the detection performance of SIFT-SVM suffers significantly when a small training set is used, whereas our approach is less affected.

Finally, we evaluate the bounding boxes using the standard overlap ratio (Jaccard index) between the predicted and ground-truth bounding boxes [23]. A detection is considered correct when the ratio is larger than 50%. Table 4 shows the evaluation of the detection bounding boxes using F_1 score. On all 4 datasets, the bounding box results using our method are better than those of the baseline SIFT-SVM.

5. Conclusion

In this paper, we propose joint object detection and counting framework using object density maps that is based on 2D integer programming. In our experiments, we achieve very good results on several challenging datasets of very different small crowded objects, including cells, pedestrians, flies, honeybees, fish and seagulls. These experiments suggest that our framework is robust to different visual appearances, object structures and poses, as well as partial-occlusion that is common with crowded scenes. In addition, our method is less affected by small training sets than traditional sliding window detectors (e.g. SIFT-SVM) and other counting-detection approaches (Region-SVM).

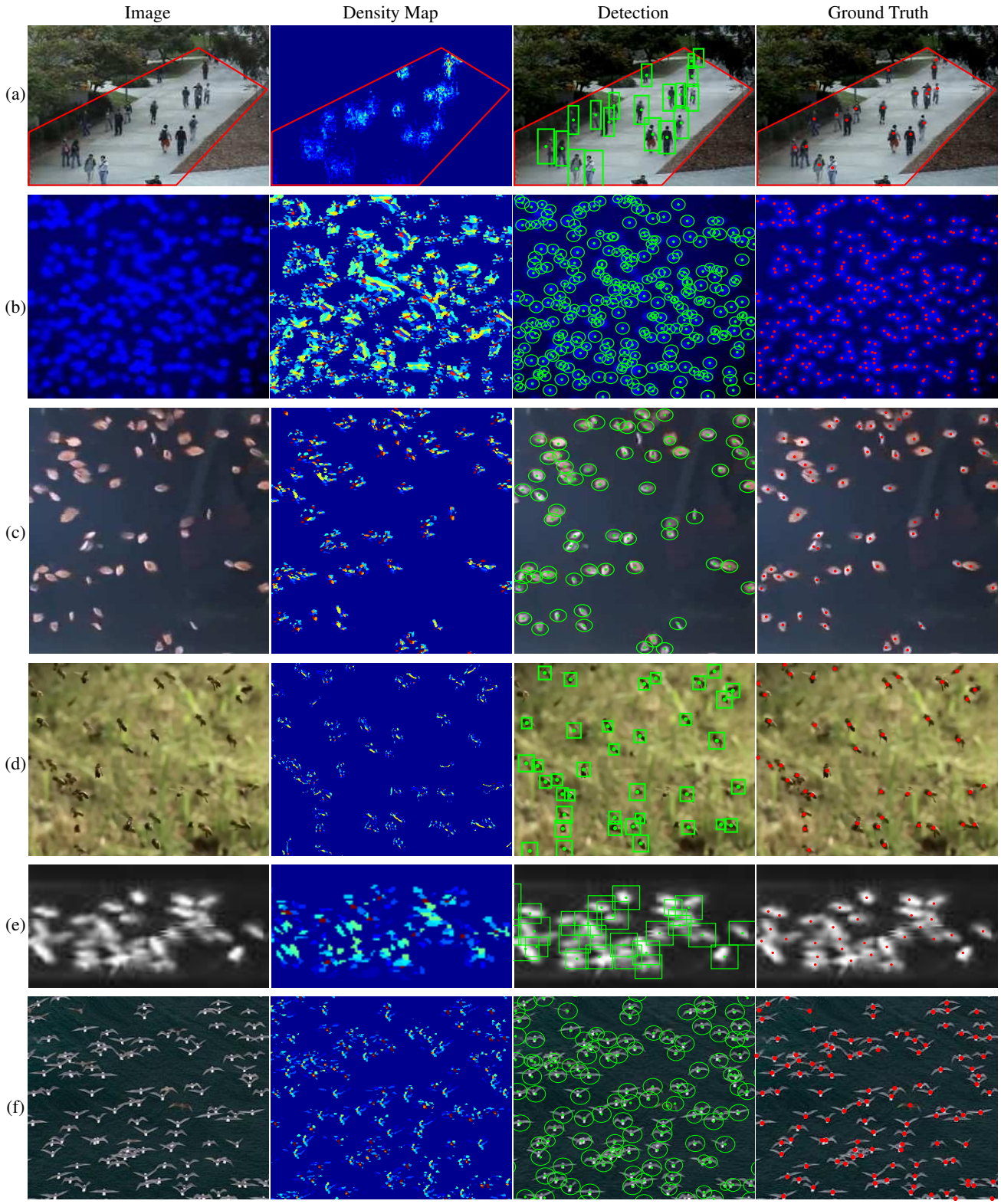


Figure 10. Results of the proposed detection method on (a) UCSD pedestrians, (b) Synthetic cells, (c) Fish, (d) Honeybees, (e) Flies, and (f) Seagulls. The red dots are the ground truth locations and the green boxes/circles are the detection results.

Acknowledgments

The authors thank Victor Lempitsky for the UCSD density maps and code from [12]. This work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 123212 & CityU 110513).

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886 – 93.

[2] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2008, pp. 1 –8.

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

[4] D. Ryan, S. Denman, C. Fookes, and S. Sridharan, "Crowd counting using multiple local features," in *Digital Image Computing: Techniques and Applications*, 2009, pp. 81 –88.

[5] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for multi-class object layout," *International Journal of Computer Vision*, vol. 95, pp. 1–12, 2011.

[6] T. W. Nattkemper, H. Wersing, W. Schubert, and H. Ritter, "A neural network architecture for automatic segmentation of fluorescence micrographs," *Neurocomputing*, vol. 48, no. 1-4, pp. 357 – 367, 2002.

[7] L. Dong, V. Parameswaran, V. Ramesh, and I. Zoghlami, "Fast crowd segmentation using shape indexing," in *Computer Vision, (ICCV). IEEE International Conference on*, 2007, pp. 1–8.

[8] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 259–289, 2008.

[9] B. Wu, R. Nevatia, and Y. Li, "Segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[10] T. Zhao and R. Nevatia, "Bayesian human segmentation in crowded situations," in *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. II–459–66.

[11] D. Kong, D. Gray, and H. Tao, "A viewpoint invariant approach for crowd counting," in *Intl. Conf. Pattern Recognition*, 2006, pp. 1187 –90.

[12] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Advances in Neural Information Processing Systems*, 2010.

[13] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2013, pp. 2547–2554.

[14] C. Arteta, V. Lempitsky, J. Noble, and A. Zisserman, "Learning to detect partially overlapping instances," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2013, pp. 3230–3237.

[15] ——, "Learning to detect cells using non-overlapping extremal regions," in *MICCAI 2012*, 2012, pp. 348–356.

[16] Z. Ma and A. Chan, "Crossing the line: Crowd counting by integer programming with local features," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2013.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[18] V. Lempitsky, "Learning to count objects in images," http://cmp.felk.cvut.cz/cmp/events/colloquium-2010.10.21/, 2010.

[19] MOSEK, "The mosek optimization software." http://www.mosek.com/., 2013.

[20] IBM, "Ibm ilog cplex optimizer," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, 2013.

[21] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2008, pp. 1 –7.

[22] E. Bernardis and S. Yu, "Pop out many small structures from a very large microscopic image," *Med. Image Analysis*, 2011.

[23] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2010.