Interactive Object Counting

Carlos Arteta[1], Victor Lempitsky[2], J. Alison Noble[1], and Andrew Zisserman[1]

[1] Department of Engineering Science, University of Oxford, UK
[2] Skolkovo Institute of Science and Technology (Skoltech), Russia

Abstract. Our objective is to count (and localize) object instances in an image *interactively*. We target the regime where individual object detectors do not work reliably due to crowding, or overlap, or size of the instances, and take the approach of estimating an object density.

Our main contribution is an interactive counting system, along with solutions for its main components. Thus, we develop a feature vocabulary that can be efficiently learnt on-the-fly as a user provides dot annotations – this enables densities to be generated in an interactive system. Furthermore, we show that object density can be estimated simply, accurately and efficiently using ridge regression – this matches the counting accuracy of the much more costly learning-to-count method. Finally, we propose two novel visualization methods for region counts that are efficient and effective – these enable integral count regions to be displayed to quickly determine annotation points for relevance feedback.

The interactive system is demonstrated on a variety of visual material, including photographs, microscopy and satellite images.

Keywords: Interactive vision systems, object counting, relevance feedback, visual recognition, biomedical image analysis.

1 Introduction

Counting instances of an object in crowded scenes is a tedious task frequently encountered in biology, remote sensing and surveillance data analysis. Towards this goal, several computer vision approaches that use machine learning in order to automate such tasks have been suggested [5, 8, 10, 12]. These methods are however limited to the scenarios where it is possible to collect and to manually annotate a representative set of training images. Systematic differences in the visual appearance or in the geometrical patterns between the training and the test images pose challenges for these methods and may result in systematic counting biases. This can be a serious limitation for practitioners, as e.g. any change of experimental protocol/conditions in a biological experiment might require reannotation and retraining.

Here, we present a system that addresses the counting task through *interactive* machine learning. In our case, the user annotates (i.e. with dots) the objects that belong to a representative but potentially small part of an image and the system propagates the annotations to the rest of the image. The results are then presented to the user, who then has the option to annotate another part of the image where the system has made significant errors. When the user is satisfied with the result, the system provides the count of the objects in the image. Our unoptimized MATLAB implementation takes at most a

D. Fleet et al. (Eds.): ECCV 2014, Part III, LNCS 8691, pp. 504–518, 2014.

few seconds for each iteration of the relevance feedback that includes recomputation of the features, discriminative (re)training, and visualization.

The approaches that cast the counting problem as one of object density estimation [5, 8, 10, 12] have so far been more accurate and also faster than approaches that detect individual instances. For this reason, we base our system around object density estimation. Counting through density estimation works by learning a mapping $\mathcal{F} : \mathcal{X} \mapsto \mathcal{Y}$ between local image features $\mathcal{X}$ and object density $\mathcal{Y}$, which then allows the derivation of an estimated object density map for unseen images (or non-annotated parts of an image in our case). If the learning is successful, integrating over regions in the estimated density map provides an object count within such regions. The accuracy of the density estimation depends crucially on the choice of local image features. Another important aspect of density-based counting is that density per se is not informative for a human user and cannot be used directly to verify the accuracy of the counting (and to provide feedback).

To address this aspect, and to achieve the processing speed demanded by the interactive scenario, we make the following three contributions. First, we propose a simplified approach for supervised density learning based on ridge regression (i.e. simple linear algebra operations). We show that even in the traditional batch mode (i.e. when learning from a set of annotated images and applying to a number of similar images) this approach achieves similar counting accuracy to the constraint-generation based learning in [12] (using the same features), while being dramatically faster to train, which is crucial for an interactive system. Furthermore, we propose two ways to visualize the estimated density, so that counting mistakes are easily identifiable by the user. This allows our system to incorporate user feedback in an interactive regime according to the user's own criterion of goodness. Finally, we propose an online codebook learning [20] that, in real-time, re-estimates low-level feature encoding as the user annotates progressively larger parts of an image. Such feature re-estimation (together with fast supervised density estimation) allows our system to avoid severe under- or over-fitting, whenever a small or a large part of an image is annotated.

In summary, this work presents a system for counting multiple instances of an object class in crowded scenes that *(i)* can be used interactively as it is fast to compute and can re-use the results from previous iterations, *(ii)* uses the simplicity and ease of computation of ridge regression, and *(iii)* presents the density estimation results in intuitive ways such that it is easy to know where further annotations may be required. We show the performance of the method in some of the typical applications for counting and detection methods such as counting cells in microscopy images, and various objects of interest in aerial images.

1.1 Related Work

Interactive Computer Vision: To the best of our knowledge, this is the first time an interactive counting system has been proposed. Nevertheless, interactive learning methods have been popular in computer vision as they provide the users with tools that can immediately and significantly alleviate tedious and time-consuming tasks, without the need to carefully prepare training datasets. The most common examples are the general purpose image segmentation methods [4, 16] which are still widely used today in real

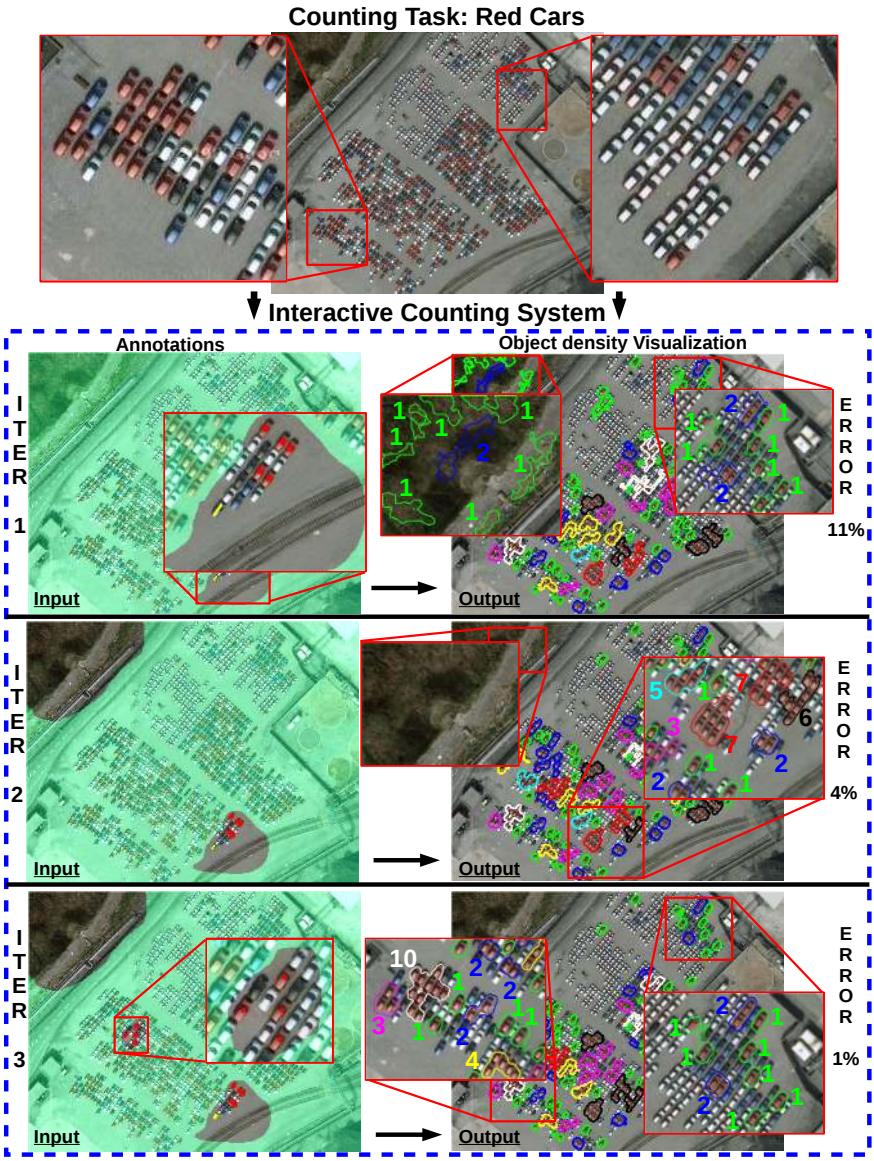


Fig. 1. *(Interactive Framework Overview)* Given an input image (or set of images) containing multiple instances of an object, our framework learns from regions with dot-annotations placed by the user in order to compute a map of *object density* for the non-annotated regions. The left column shows the annotated pixels provided by the user – all regions in the annotation images *outside* the green mask (i.e. with or without dot-annotations) are used as observations in the regression. The right column shown the intuitive visualization tool of the density estimation that allows the user to inspect the results and add further annotations where required in order to refine the output of the counting framework.

applications. Interactive image segmentation methods have also been widely adopted in medical imaging [9, 19, 20] where not only is it difficult to prepare datasets, but also users need to have the possibility of refining results in real-time. More recently, interactive methods have been applied to object detection [25] and learning of edge models for image partitioning [21], which is also highly relevant for biological image analysis. We expect this paper to bring the general benefits of interactive learning to counting problems.

Counting through Object Density Estimation: Counting objects in images without explicit object detection is a simplified alternative for cases where only the number of objects is required; it has proved to be especially useful in those tasks too challenging for object detectors such as crowded scenes. Initial efforts in this class of methods attempted to learn global counts through regressors from image-level [6, 11] or region-level [5, 13, 17] features to the total number of objects in it. However, these approaches ignored the local arrangements of the objects of interest. Towards this end, [12] proposed learning pixel-level object *density* through the minimization of the $MESA$ distance, a cost function specially tailored for the counting problem. Following [12], [8] proposed a simplified version of the object density estimator based on random forest, which represented an improvement in training time. Generating a pixel-level object density estimation not only has the advantage of estimating object counts in any region of the image, but can also be used to boost the confidence of object detectors as shown in [15]. Our approach follows the ideas in [8, 12] and further simplifies the learning to a few algebraic operations using a ridge regression, thus enabling an interactive application. Note that the learning approach proposed here should not be confused with the ridge regression used as baseline in [12], which belongs to the category of image-level regressors.

2 Interactive System Overview

Given an image $\mathcal{I}$, the counting proceeds within a feedback loop. At each iteration, the user marks a certain portion of an image using a freehand selection tool (we refer to pixels being included into such regions as *annotated pixels*). Then the user *dots* the objects, by clicking once on each object of interest within this annotation region. In the first iteration, the user also marks a diameter of a typical object of an image by placing a line segment over such a typical object.

At each iteration, given the set of dotted pixels $\mathcal{P}$ placed by the user on top of objects of interest in $\mathcal{I}$, our system aims to (1) build a codebook $\mathcal{X}$ of low-level features, (2) learn a mapping $\mathcal{F} : \mathcal{X} \mapsto \mathcal{Y}$ from the entries in the codebook $\mathcal{X}$ to an object density $\mathcal{Y}$, (3) use the learned mapping $\mathcal{F}$ to estimate the object density in the entire image $\mathcal{I}$, and (4) present the estimated object density map to the user through an intuitive visualization. The estimated object density map produced is such that integrating over a region of interest gives the estimated number of objects in it (e.g. integrating over the entire map gives an estimate of the total number of objects of interest in $\mathcal{I}$). By using the density visualization, the user can easily spot significant errors in the object density estimate, and can proceed to provide further annotations to refine the results in a next iteration of the process. An example with three iterations is shown in Figure 1. In practice, only a small

portion or few small portions of a potentially large image have to be inspected in order to validate the correctness of the learned model or to identify parts with gross errors.

The first design requirement for an interactive counting system, such as the above, is that the codebook needs to be fast to compute, and adapt its size according to the amount of annotations in order to prevent under- or over-fitting. To address these problems, we propose in Section 3 a simple progressive codebook learning procedure, which builds a kd-tree that can grow from its current state as the user provides further annotations. The second requirement is a fast computation of the mapping $\mathcal{F}$. Our proposal is a pixel-level ridge regression presented in Section 4, which has a closed-form solution. Thus, the mapping $\mathcal{F}$ can be computed extremely fast through a few algebraic operations on sparse matrices. Finally, the system needs to present its current estimates to the user in such a way that identifying errors can be done through a quick visual inspection, which is not possible with the raw object density map and/or the global count. Therefore, we propose in Section 5 two methods to visualize object density maps by generating local "summaries" of it.

We show the performance of the interactive counting system through a series of examples in the experimental section, and videos of the interactive process are provided in [1].

3 Progressive Codebook Learning

Initially, we represent each pixel p of an image $\mathcal{I}$ with a d-dimensional real-valued vector $z_p \in \mathbf{R}^d$. The idea of building a codebook on top of these low-level features is to subdivide the feature space into k cells, so that the typical density for appearance patterns falling into each cell is roughly constant. Ideally, we want to strike a balance between two conflicting goals. Firstly, we want to partition the feature space finely enough to avoid underfitting. Secondly, we want each of the partitions to have at least several pixels that belong to the area annotated by the user in order to avoid overfitting. The latter requirement leads to the idea of interactive re-estimation of the feature space partition as more annotations become progressively available. This can be done very efficiently with the following algorithm.

We initialize the feature space partitioning by assigning all image pixels to the same partition. We then proceed recursively by splitting the partitions that contain more than N "annotated" pixels assigned to them ("annotated" here means belonging to the user-annotated area). Here, N is a meta-parameter selected by the user, which we set to 200 in our experiments. In more detail, the algorithms proceed as follows:

1. In the i-th iteration, find the partitions with more than N descriptors z_p assigned to it (only annotated pixels are taken into account).
2. For each of those partitions, find the feature dimension t of maximum variance (among the d dimensions), as well as the median of the values of all annotated pixels corresponding to this dimension.
3. Split such a partition into two according to whether a pixel value at the dimension t is greater or smaller then the median.
4. Repeat until every partition has less than N annotated pixels assigned to it.

The proposed algorithm thus constructs the kd-tree (w.r.t. the annotated pixels). Note, however, that we also maintain the partition assignments of the unannotated pixels (and the resulting partitions can be unbalanced w.r.t. the unannotated pixels). We finally note that there is no need to store the resulting kd-tree explicitly because the algorithm maintains the assignments of pixels to the leaves of the kd-tree (i.e. partitions). The partitioning algorithm is resumed whenever new annotations are added by the user. At this point, the codebook can grow from its current state by continuing the splitting (and is not re-learned from scratch).

Once the codebook has been learned, each pixel p in the image $\mathcal{I}$ is represented by a sparse k-dimensional vector x_p, where all entries are zero except the one corresponding to the partition to which the image descriptor z_p was assigned ("one-hot" encoding). The representation x_p is then used as the pixel features within the learning framework proposed in Section 4. Once again, we emphasize that the vector x_p changes (and becomes more high-dimensional) between the learning rounds as more user annotations become available.

4 Counting with Ridge Regression

We now introduce our simple alternative to [12] for learning an object density estimator. Our method here is similar to (and, arguably, simpler than) [8]. Most importantly, compared to [12], our approach reduces the training time from several dozens of seconds to a few seconds (for heavily annotated images), thus enabling the interaction.

Similarly to [8, 12], we define the ground truth density from the set of user dot-annotations $\mathcal{P}$ as the sum of delta functions centred on each of the annotation dots:

$$F^0(p) = \sum_{p' \in \mathcal{P}} \delta(p - p') \ , \tag{1}$$

where p is the (x, y) position of a pixel.

We want to learn the mapping $\mathcal{F} : \mathcal{X} \mapsto \mathcal{Y}$ from local image features to a ground truth object density using ridge regression. Let us assume that each pixel p in a training image $\mathcal{I}$ is represented with a sparse vector $x_p \in R^k$ from a learned codebook. At the same time, each pixel is associated with a real-valued ground truth object density $y_p \in R$ according to $F^0(p)$. The ridge regression finds a k-dimensional vector of coefficients w that minimizes the following objective:

$$||Xw - Y||^2 + \lambda ||w||^2 \rightarrow \min_w \tag{2}$$

Here, X is the matrix of predictors (feature vectors) with each row containing x_p, Y is a vector of corresponding density values from the ground truth density F^0 and λ controls the balance between prediction error and regularization of w. Although computationally simple and efficient, the fitting procedure (2) tries to match the ground truth density values exactly in every pixel, which is unnecessary and leads to severe overfitting. Instead, as was argued in [12], the estimated densities Xw should match the ground truth densities Y when integrated over extended image regions (i.e. we do not care about very local deviations between Xw and Y as long as these deviations are unbiased and sum approximately to zero over extended regions).

Based on this motivation, [12] replaces the L2-distance between Xw and Y in (2) by the so called MESA-distance which looks at integrals over all possible box regions. While this change of distance dramatically improves the generalization, the learning with MESA-distance is costly as it employs constraint generation and has to solve a large number of quadratic programs. Here, we propose another, much simpler, alternative for the L2-distance in (2). Namely, we minimize a smoothed version of the objective by convolving the difference between the ground truth density and the estimated density with a Gaussian kernel:

$$||G * (Xw - Y)||^2 + \lambda ||w||^2 \rightarrow \min_w \tag{3}$$

Here, $G*$ denotes (with a slight abuse of notation) the Gaussian smoothing over the image plane (i.e. the column vector has to be reshaped back to the image dimensions and smoothed spatially). Typically, we use a sufficiently large isotropic covariance to ensure that the local unbiased deviations between Xw and Y are smoothed (so that "excesses" and "deficits" cancel each other). We found that the performance of the estimated w on the test set is not sensitive to large variation in the covariance (as long as the covariance parameter σ within G is greater than say half a typical object diameter). As we will show below, such smoothing is crucial for good performance of the counting.

Because of the linearity of the convolution, we can rewrite (3) as:

$$||(G * X)w - G * Y)||^2 + \lambda ||w||^2 \rightarrow \min_w , \tag{4}$$

where $(G * X)$ denotes Gaussian smoothing applied to each column of X.

Importantly, (4) can be regarded as ridge regression between the smoothed version of the feature maps $G * X$ and the smoothed ground truth density. The latter can be seen as the sum of Gaussian kernels centered on the user annotations:

$$F^0(p) = \sum_{p' \in \mathcal{P}} \mathcal{N}(p', \sigma) \tag{5}$$

Similarly, the smoothed (but still sparse) matrix of predictors $G * X$ can be obtained by convolving independently each dimension of the feature vectors (i.e. each column of X), that is, spatially blurring each of the feature channels.

Using the vertically concatenated smoothed maps $X_s = [G * X]$ and $Y_s = [G * Y]$ respectively, w can be expressed using a standard ridge regression solution formula:

$$w = (X_s^T X_s + \lambda \Gamma^T \Gamma)^{-1} X_s^T Y_s , \tag{6}$$

where Γ denotes the identity matrix and λ is a regularization parameter.

Finally, for a non-annotated region of image $\mathcal{I}$, the density value at pixel p can be obtained using the estimated w through a simple linear transform of the non-smoothed feature vectors x_p (i.e. in the same way as in [12]):

$$F(p) = w^T x_p \tag{7}$$

It can be seen that learning the mapping vector w only involves simple matrix operations (mostly on sparse matrices) and Gaussian smoothing. Thus, w can be learned on-the-fly and with a low memory footprint.

We have found that the generalization performance is improved slightly if the non-negativity of the estimated w is enforced (which for non-negative x_p results in physically meaningful non-negative object densities at test time). Since it is computationally more expensive to include a non-negativity constraint within ridge regression in a principled manner (compared to the closed-form solution provided by unconstrained ridge regression), we use a simple trick of iteratively re-running ridge regression, while clipping the components of w having negative values to zero after each iteration.

4.1 Experimental Validation of Ridge Regression Counting

We now evaluate the counting method based on ridge regression and determine how it compares to previous work in the area that uses traditional batch processing (i.e. non-interactive). In particular, we compare to the most related approach [12] (using the same features, the same datasets, and the same experimental protocols as in [12]).

Table 1 shows the experimental results on the USCD pedestrian dataset [5], which consists of 2000 frames of dot-annotated pedestrians from a surveillance camera video. Table 2 shows the results on the synthetic cell dataset presented in [12]. The dataset

Table 1. Mean absolute errors for people counting in the UCSD surveillance camera video [5]. The columns correspond to four training/testing splits of the data ('maximal','downscale','upscale','minimal') proposed in [17]. The average number of people per image in each of the testing sets is shown in brackets on the top row. $\star$ indicates the methods tested with the same set of features. The proposed method (bottom line) matches on average the performance of the previous best method for the same features.

	'max' [28.25]	'down' [24.35]	'up' [29.68]	'min' [28.25]
Global count [11]	2.07	2.66	2.78	N/A
Segment+Count [17]	1.53	1.64	1.84	1.31
Density estimation (MESA) [12] $\star$	1.70	1.28	1.59	2.02
Density estimation (RF) [8]	1.70	2.16	1.61	2.20
Density estimation (proposed) $\star$	1.24	1.31	1.69	1.49

consists of 200 synthetic images of cells in fluorescence microscopy, with an average of 174 $\pm$ 64 cells per image. Between 1 and 32 images are used for training and validation,while testing is performed on a set of 100 images. The procedure is repeated for five different sets of N images, and the mean absolute counting errors and standard deviations are reported for different values of N. We additionally include the result of the method that does not blur the feature channels, and thus uses (2) rather than (3) as a learning criterion (in the special case of one-hot encoding and $\lambda = 0$, it simply corresponds to averaging the GT density value corresponding to each codeword in the dictionary). The inferior performance of this baseline highlights the importance of the smoothing in (3).

It can be seen from Table 1 and Table 2 that using simple ridge regression never results in a significant drop in performance compared to the more complex approach in [12], and thus, we use it as part of our interactive system (described next) without

Table 2. Mean absolute errors for cell counting in the synthetic cell dataset [12]. The columns correspond to the different sizes of the training and validation sets. $\star$ indicates the methods that use the same features. The proposed method (bottom line) matches the performance of the previous best method for the same features. The version that does not smooth the difference between the estimated and the GT density when evaluating the solution, performs considerably worse. The method [8] achieves lower counting error in this case, however it uses different and much stronger features learned in a supervised fashion.

	N = 1	N = 2	N = 4	N = 8	N = 16	N = 32
Img-level ridge reg. [12] $\star$	67.3 ± 25.2	37.7 ± 14.0	16.7 ± 3.1	8.8 ± 1.5	6.4 ± 0.7	5.9 ± 0.5
Dens. estim. (MESA) [12] $\star$	9.5 ± 6.1	6.3 ± 1.2	4.9 ± 0.6	4.9 ± 0.7	3.8 ± 0.2	3.5 ± 0.2
Dens. estim. (RF) [8]	N/A	4.8 ± 1.5	3.8 ± 0.7	3.4 ± 0.1	N/A	3.2 ± 0.1
Dens. estim. (no smooth) $\star$	13.8 ± 3.6	11.3 ± 3.1	10.6 ± 2.3	9.9 ± 0.7	10.6 ± 1.1	10.2 ± 0.4
Dens. estim. (proposed) $\star$	9.6 ± 5.9	6.4 ± 0.7	5.53 ± 0.8	4.5 ± 0.6	3.8 ± 0.3	3.5 ± 0.1

any compromise on the counting accuracy. Crucially for sustaining interactivity, our simplification yields a dramatic speedup of the learning procedure.

To avoid confusion, we note again that ridge regression was proposed as a baseline for counting in [12], as shown in Table 2, and tested with the same set of features we have used, but resulting in much poorer performance. This is because, the regression in that baseline was learned at the image level, i.e. it treated each of the training images as a single training example, which resulted in a severe overfitting due to a limited number of examples. This differs from learning w to match the pixel-wise object densities (it can be seen as an extreme case of infinitely wide Gaussian kernel G). Finally, we note a connection between the proposed approach and [8] that also performs smoothing of the output density function (at the testing stage) using structured-output random forests.

5 Object Density Visualizations

The visualization of the object density estimate plays a key role in our interactive counting system as it assists the user to identify the parts of the image where the system has estimated the counts with large errors, and thus, where to add further annotations. While the predicted densities are sufficient to estimate the counts in any region, the accuracy of these densities cannot be controlled by the user without further post-processing due to the mismatch between the continuous nature of the densities and the discrete nature of the objects. To address this problem, we propose two density visualization methods, which convert the estimated density into representations that are intuitive for the user. The first method is based on non-overlapping extremal regions and is algorithmically similar to [3], and aims to localize the objects from the density estimate. The second method is based on recursive image partitioning, and aims to split the image into a set of small regions where the number of objects can be easily eyeballed and compared to the density-based estimates.

For both visualization techniques, we start by generating a set of candidate regions with a nestedness property such that two regions R_i and R_j are either nested (i.e. $R_i \subset R_j$ or $R_j \subset R_i$) or they do not overlap (i.e. $R_i \cap R_j = \emptyset$). Therefore, the set of candidate regions in each case can be arranged into trees [2]. Both visualization approaches

represent the densities by showing summations over a subset of those candidate regions and both approaches optimize the choice of this subset. In particular, each region R_i has a score V_i associated to it that indicates the *integrality* of the region, or how well the region encloses an entire object or a cluster of objects. The idea is that we want to show the user the regions that have near-integer integrals of the density over them. Given the integrality scores, both approaches compute the final representation by picking a non-overlapping subset of regions that maximize the sum of such integrality-driven scores.

In more detail, we begin by defining S_i to be the integral of the estimated density map over the region R_i, and I_i to be the approximation of S_i to its nearest integer. The score V_i for region R_i is then defined as:

$$V_i = (1 - (S_i - I_i))^2 \tag{8}$$

For N candidate regions, we introduce the indicator variables $\mathbf{y} = \{y_1, y_2, ..., y_N\}$, where $y_i = 1$ implies that R_i has been selected. Additionally, $\mathbf{y}$ must satisfy the constraint of only containing non-overlapping regions. That is, $\mathbf{y} \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of all sub-sets of non-overlapping regions such that if $R_i \cap R_j \neq \emptyset$ then $y_i.y_j = 0$.

The global maximization objective is defined as follows:

$$F(y) = \max_{y \in \mathcal{Y}} \sum_{i=1}^{N} y_i (V_i + \lambda) \tag{9}$$

where λ is a constant that prevents from selecting the trivial solution (one biggest region containing the whole image) and biases the solution towards a set of small regions. The objective (9) is optimized efficiently by using dynamic programming due to the tree structure of the regions as in [3]. We now discuss the details of the two approaches and the difference between them.

Visualization Using Non-overlapping Extremal Regions. Extremal regions are the connected components on the binary images resulting from thresholding a gray image $\mathcal{I}$ with any arbitrary threshold τ. A key property of the extremal regions is the nestedness as described above. Therefore, the set of extremal regions of an image can be arranged into a tree (or a forest) according to the nestedness.

Following [3], we use extremal regions as candidates for object detection. In this case, extremal regions are extracted from the estimated object density map, and the ones selected by the optimization (9) should delineate entire objects or entire clusters of objects (Figure 2-c).

In practice, we collect these candidate regions using the method of Maximally Stable Extremal Regions (MSER) [14]. This method only keeps those extremal regions that are stable in the sense that they do not change abruptly between consecutive thresholds of the image (i.e. on regions with strong edges). During the inference, we exclude the regions which have an integral of density smaller than 0.5 from consideration as we have found that allowing any extremal region to be selected can result in very cluttered visualizations. Instead, this visualization aims to show only regions containing entire objects.

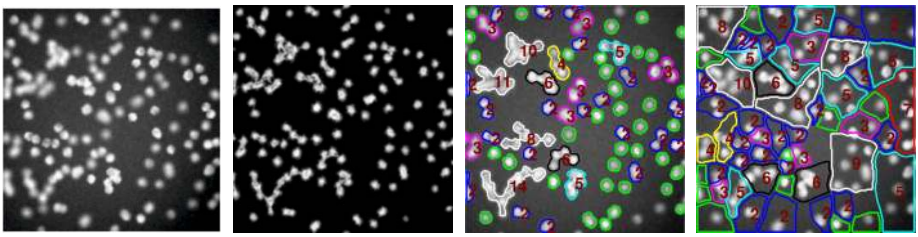
Visualization Using Hierarchical Image Partitioning. In this approach, we build a hierarchical image partition driven by the density. To obtain the partition, we iteratively

apply spectral graph clustering, dividing image regions into two (akin to normalized cuts [18]). Unlike the extremal region visualization and unlike the traditional use of normalized cuts, we encourage the boundaries of this partition to go through regions of low density, thus creating a tile of regions that enclose entire objects (Figure 2-d). To achieve this, we build a 4-connected weighted graph $G = (V, E)$ with the adjacency matrix W defining the weights of the edges based on the estimated density map $F(p)$ as $w_{p,q}$=0.5 $(F(p)+F(q))$ for $(p, q) \in E$.

The normalized cuts then tend to cut through the parts of the image where the density is near-zero, and also as usual have a bias towards equal-size partitions (which is desirable for our purpose).

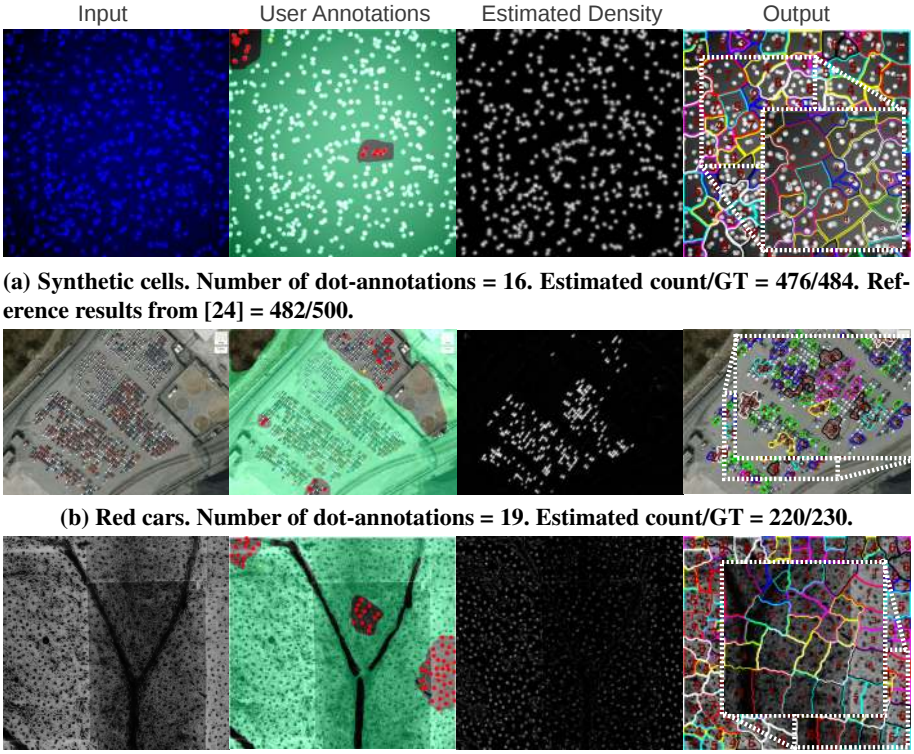
Once the tree-structured graph is built, the inference selects the set of non-overlapping regions through the maximization of the sum of the integrality scores of the regions, as explained above. Additionally, we enforce at inference time that every pixel in the estimated density map must belong to one of the selected regions (i.e. that the selected subset of regions represent a cover). Therefore, the entire density distribution is "explained". Accordingly, all regions from the hierarchical partitioning of the image are considered, including those with near zero density integrals.

Compared to the visualization using the extremal regions, the visualization based on recursive partition does not tend to outline object boundaries, but represents the underlying ground truth density with greater fidelity due to the fact that the whole image ends up being covered by the selected regions (Figure 2).



(a) Input | **(b)** Density estimation Total count = 180 | **(c)** ER-visualization Total count = 163 | **(d)** SC-visualization Total count = 185

Fig. 2. *(Density Visualization)* In order to assess the density estimation (b) of the original image (a), we propose two visualization methods. The first method (c) is based on non-overlapping extremal regions (ER) and aims to localize objects in the estimated density map (more intuitive but biased towards undercounting). The second method (d) is based on hierarchical image partitioning with spectral clustering (SC) and aims to explain the distribution of the density estimate across the entire image (higher fidelity but less intuitive visualization of the density). See text for details. In (c) and (d), the numbers indicate the objects contained within the region. Green regions contain a single object, but the number has been omitted for clarity. Non-outlined regions in (d) have zero counts.



(a) Synthetic cells. Number of dot-annotations = 16. Estimated count/GT = 476/484. Reference results from [24] = 482/500.

(b) Red cars. Number of dot-annotations = 19. Estimated count/GT = 220/230.

(c) Stomata. Number of dot-annotations = 37. Estimated count/GT = 655/676. Reference results from [24] = 716/676.

Fig. 3. *(Example results)* For the experiments shown, a large image (first column) is annotated interactively (second column) until qualitatively reasonable results are produced (fourth column). For all examples, the green masks on the annotation images (second column) indicate regions that have not been annotated by the user. All regions outside the green mask, with or without dot-annotations, are used as observations in the regression (Section 4). Annotated regions without dots can be seen as zero annotations. As expected, the number of annotations required increases with the difficulty of the problem. In cases where the background is complex, such as in aerial images (b), residual density tends to appear all over the image, which can be seen in the visualization. However, this can be easily fixed interactively using zero annotations, which are very fast and simple to add. Many more examples, as well as videos of the interactive process, are provided at [1].

6 Interactive Counting Experiments

We show the qualitative performance of the interactive counting system. The aim is to give a sense of the amount of annotations (and effort) required to obtain an object count that would closely approximate the ground truth (i.e. with an approximate absolute counting error of 10% or less). This section is complemented with [1], where a video of the system in use is shown, as well as additional example results including counting elongated objects using a scribble variant of the annotation. Note that, though results are shown here using the same images as input and output, it is possible to propagate the density estimation to other similar images in a batch.

Figure 3 shows example results of the interactive counting system, indicating the number of annotations added and the estimated object count for that amount of annotation. Part of the examples (Figures 3-a,c) have been taken from the benchmark dataset of [24], and we use their results *as reference* in Figure 3. We do not attempt to do a direct comparison of performance with [24] due to the fact that for the cases where a single image is given, our interactive method requires annotations on this image in order to produce results, and thus, disrupts the possibility of a fair comparison of performance. Moreover, due to the nature of the low-level features, our system crops the borders of the image by half the size of the texture patches (see implementation details), resulting in a possible difference of the ground truth count w.r.t. the original image. The additional examples (Figure 3-b and examples in [1]) correspond to aerial images extracted from Google Maps.

The same set of parameters have been used for all the examples shown, with the most relevant ones indicated in the implementation details. Due to space limitations, we use a single visualization method for each of the examples in Figure 3, but it can be seen that they are complementary. Nevertheless, depending on the image, one visualization can be more convenient than the other.

6.1 Implementation Details

Low-Level Features. We compute the initial (low-level) pixel descriptor z_p based on two types of local features on the Lab color-space. First, we use the contrast-normalized lightness values (L channel) of the pixels in a patch of size $n \times n$ centered at p [22]. The patches are rotated such that their dominant gradients are aligned in order to be invariant to the object's rotation in the image. Secondly, we collect the raw L, a and b values of the center pixel. The descriptor $z_p \in \mathbf{R}^d$ is the concatenation of the two local features. Therefore, the dimensionality d of the pixel descriptor for a color image is $n^2 + 3$. In the case of grayscale images, we do the feature computation on the given intensity channel, which results in $d = n^2 + 1$.

Collecting Extremal Regions. Extremal regions are extracted from the estimated density map using the MSER implementation from *VLFeat* [23]. In order to collect enough candidate regions for the inference to select from, we set a low stability threshold in the MSER algorithm.

Building a Binary Tree with Spectral Clustering. Computing the traditional spectral clustering as in [18] can be too slow for our interactive application, and the reason is the expensive computation of eigenvectors. Therefore, in practice we use the method from

Dhillon *et al.* [7] which solves the equivalent problem of weighted kernel k-means thus greatly reducing the computation time. We use the implementation from the authors of [7].

Setting Object-Size Dependent Parameters. Some of the parameters used in the implementation of the interactive counting system are better set with respect to the size of the object of interest. These are the size $n \times n$ of the patches for the low level features and the standard deviation σ for the Gaussian kernel used to smooth the dot-annotations and feature channels for the ridge regression. As discussed in Section 2, we chose to request an additional input from the user, where the approximate diameter of the object of interest is input by drawing a line segment over a single object in the image. The image is then rescaled with the scale factor of the object. For the experiments of the interactive system shown in the experimental section, we use an object size of 10 pixels, patches of 9×9 pixels and $\sigma = 3$ pixels.

Region Visualization. The boundaries of the regions that are chosen to visualize the density are superimposed on top of the original images. Alongside the boundaries, we show the density integrals over the highlighted regions rounded to the nearest integer (recall that regions are chosen so that such integrals tend to be near integer). We also color code the boundaries according to the counts (e.g. green for objects containing one object, blue for two objects, etc.).

7 Summary and Discussion

This paper is a first foray into enabling counting, previously treated as a traditional batch learning problem, to be handled interactively. To do this we have proposed a solution that speeds up the learning of object densities and overcomes the challenge of efficient density visualization. The result is an agile and flexible system which enables quite disparate visual material (spanning both microscopy images of cells and satellite imagery) to be annotated and counted in a matter of seconds.

There is certainly room for improvement: firstly, the features used can be extended to enable more local and contextual information to be captured. Secondly, our current system does not handle perspective geometry and cannot be directly applied to images with objects on a slanted ground plane. The latter, however, can easily be fixed by allowing a projective transformation to be imported or by the user providing additional object size annotations.

Acknowledgements. Financial support was provided by the RCUK Centre for Doctoral Training in Healthcare Innovation (EP/G036861/1) and ERC grant VisRec no. 228180.

References

1. `http://www.robots.ox.ac.uk/~vgg/research/counting/`
2. Arteta, C., Lempitsky, V., Noble, J.A., Zisserman, A.: Learning to detect cells using non-overlapping extremal regions. In: Ayache, N., Delingette, H., Golland, P., Mori, K. (eds.) MICCAI 2012, Part I. LNCS, vol. 7510, pp. 348–356. Springer, Heidelberg (2012)
3. Arteta, C., Lempitsky, V., Noble, J.A., Zisserman, A.: Learning to detect partially overlapping instances. In: CVPR (2013)

4. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: Proc. ICCV, vol. 2, pp. 105–112 (2001)
5. Chan, A.B., Liang, Z.S.J., Vasconcelos, N.: Privacy preserving crowd monitoring: Counting people without people models or tracking. In: CVPR (2008)
6. Cho, S.Y., Chow, T.W.S., Leung, C.T.: A neural-based crowd estimation by hybrid global learning algorithm. Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 29(4), 535–541 (1999)
7. Dhillon, I.S., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors a multilevel approach. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(11), 1944–1957 (2007)
8. Fiaschi, L., Nair, R., Köethe, U., Hamprecht, F.: Learning to count with regression forest and structured labels. In: Proc. ICPR (2012)
9. Grady, L.: Random walks for image segmentation. TPAMI 28(11), 1768–1783 (2006)
10. Idrees, H., Saleemi, I., Seibert, C., Shah, M.: Multi-source multi-scale counting in extremely dense crowd images. In: CVPR (2013)
11. Kong, D., Gray, D., Tao, H.: A viewpoint invariant approach for crowd counting. In: Proc. ICPR (2006)
12. Lempitsky, V., Zisserman, A.: Learning to count objects in images. In: NIPS (2010)
13. Ma, W., Huang, L., Liu, C.: Crowd density analysis using co-occurrence texture features. In: 2010 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), pp. 170–175 (November 2010)
14. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. Image and Vision Computing (2004)
15. Mikel, R., Ivan, L., Josef, S., Jean-Yves, A.: Density-aware person detection and tracking in crowds. In: ICCV (2011)
16. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: interactive foreground extraction using iterated graph cuts. Proc. ACM SIGGRAPH 23(3), 309–314 (2004)
17. Ryan, D., Denman, S., Fookes, C., Sridharan, S.: Crowd counting using multiple local features. In: Proc. DICTA (2009)
18. Shi, J., Malik, J.: Normalized cuts and image segmentation. TPAMI (2000)
19. Singaraju, D., Grady, L., Vidal, R.: Interactive image segmentation of quadratic energies on directed graphs. In: Proc. of CVPR 2008. IEEE Computer Society (June 2008)
20. Sommer, C., Straehle, C.N., Köthe, U., Hamprecht, F.A.: Ilastik: Interactive learning and segmentation toolkit. In: ISBI, pp. 230–233 (2011)
21. Straehle, C., Koethe, U., Hamprecht, F.A.: Weakly supervised learning of image partitioning using decision trees with structured split criteria. In: ICCV (2013)
22. Varma, M., Zisserman, A.: Texture classification: Are filter banks necessary? In: CVPR (2003)
23. Vedaldi, A., Fulkerson, B.: Vlfeat - an open and portable library of computer vision algorithms. In: ACM Multimedia (2010)
24. Verdié, Y., Lafarge, F.: Detecting parametric objects in large scenes by monte carlo sampling. International Journal of Computer Vision, 1–19 (2013)
25. Yao, A., Gall, J., Leistner, C., Gool, L.J.V.: Interactive object detection. In: CVPR, pp. 3242–3249 (2012)