Imperial College London
Department of Computing

Automatic Cell Tracking in Noisy Images for Microscopic Image Analysis

Pedro Damian Kostelec

September 2014

Supervised by Ben Glocker

Submitted in part fulfilment of the requirements for the degree of Master of Science in Computing (Artificial Intelligence) of Imperial College London

Abstract

Acknowledgement

I offer my sincerest gratitude to life, ...

Contents

3 Detection of cells

$\boxed{\text{IN PROGRESS}}$

3.1 Cell detection overview $\boxed{\text{DRAFT I}}$

In order to track cells across frames we need to first be able to identify the cells within the images. As described in **??**, there is a broad range of automatic methods for cell detection and segmentation.

The requirements for detecting cells in our application are:

1. Accurate detection in sometimes noisy images. The detector should be able to robustly identify cells in datasets that include noise, motion artefacts (induced by the camera and the moving tissue) and variable lens focus. The detector does not need to detect cells that are out of focus, but should reliable detect those are in focus. Out of focus frames are dealt with in the tracking module.

2. There is no need for accurate segmentation. We are primarily interested in the accurate tracking of cells, and analysis of the cells appearance is of secondary interest. For the purpose of tracking, it is sufficient to be able to reliable localize cells, and extract some features that identify these cells.

3. The algorithm should adapt well to different microscopy modalities. The studied datasets are obtained from various organs within the body of mice that have distinct textures. The detector should

be able to detect cells in any of these datasets with minimal manual adjustments.

Much of the previous work described in **??** was focused on accurate segmentations of cells. Many of the described methods would return very good segmentations, but they would fail when presented with a very noisy dataset of varying contrast, or need major manual tuning of parameters when presented with a new dataset with a different kind of cells.

For the purpose of our application we have chosen the cell detector presented in [1], because it conforms to all three requirements. The machine learning method is able to handle noisy datasets, and the model can easily be retrained when a new, previously unseen, type of images needs to be analysed. The method does not focus on accurate segmentation of cells, but on the robust localization. The major drawback of this method was performance. The authors reported run-times of 30 seconds to detect cells on 400-by-400 pixel images on an i7 CPU. This slow performance would make it unusable for tracking large image sequences. However, we were able to optimize the MATLAB code for feature computation so that the detection process takes only a fraction of the original run-time.

The detection process runs in three steps. First, robust candidate regions are extracted from an image. The method uses an efficient MSER region detector to find a large number of candidate regions. Cite: . Second, each of these regions is assigned a value which is produced by a classifier, which indicates the appropriateness score that this this region is a cell. Finally, the non-overlapping subset of these candidate regions with high similarity to the annotated cells in the training data is selected via dynamic programming.

The detector can be trained on a small number of training images, where each cell is annotated with a single dot.

The code used in our method is based on the original code from [1]. Some of our modifications, especially in the feature computations, allow the detector to run significantly faster. Additional work was required to combine the cell detector with the tracking module described in chapter 4.

The remained of this chapter is divided as follows. In section 3.2 we describe how the candidate regions are obtained. Section 3.3 shows the inference procedures that selects the optimal subsets of candidate regions. In section 3.4 we formulate the learning method and in we describe the choice of features in section 3.5. Finally, in section 3.6 we briefly outline the main changes that improved the performance of the original algorithm.

3.2 Detection of candidate regions DRAFT I

The first stage of the cell detection process is the extraction of a large number of cell candidate regions. A region is extremal if the image intensity everywhere inside of it is highter than the image intensity at its outer boundary. For a grayscale image I it is the set of connected components of the thresholded image $\mathcal{I}_{>\mathrm{t}} = \{\mathcal{I} > \mathrm{t}\}$ for some threshold t. In practice we consider only regions that are maximally stable, as defined in [2], obtained using an efficient maximally stable region detector (MSER).

extremal region

maximally stable extremal region

An important property of extremal regions is *nestedness*, which means that two extremal regions detected on the same image can be either nested or non-overlapping.

The number of obtained candidate regions is high, and an inference step determines which belong to cells.

Extremal regions are invariable to affine transformations of image intensities. They are stable, which means that their support is virtually unchanged over a range of thresholds, they can be detected at multiple scales and can be enumerated very efficiently. This makes them ideal for detecting candidates regions that correspond to cells in microscopy imaging. Figure 3.1 shows an example result of running the MSER detector on a sample set of microscopy cell images.

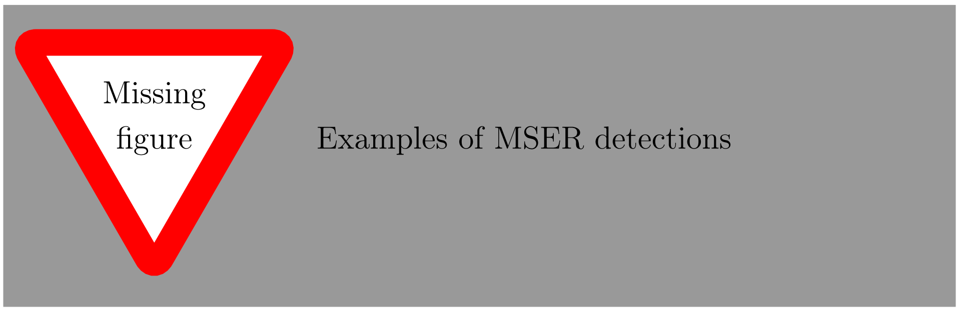

Figure 3.1: Mser detection examples

3.3 Inference under the non-overlap constraint $\boxed{\text{DRAFT I}}$

The inference procedure selects a subset of candidate extremal regions that corresponds to cells. Let V_i be a value assigned to each candidate region R_i indicating the appropriateness of the region to correspond to a cell. The inference procedures selects a subset of the non-overlapping candidate regions such that the sum of the values of selected regions is maximized. To make the problem computationally feasible, the regions are organized into tress according to the nestedness property. Each tree represents a set of overlapping extremal regions. The problem can then be formalized as followes. Let $\mathbf{y} = \{y_1, y_2, \dots y_N\}$ be binary variable such that $y_i = 1$ if the region R_i is selected. Let $mathcalY$ be the set of non-overlapping regions. The maximization problem can be defined as:

$$F(\mathbf{y}) = \max_{\mathbf{y} \in \mathcal{Y}} \sum_{i=1}^{N} y_i V_i.$$

Further details on how the exact solution is obtained can be found in [1].

Although cells in the studied datasets occasionally overlap, detecting only non-overlapping cells is a good balance between manual work required to annotate datasets, training time and accuracy of the detections. The described framework has been expanded to learn to detect partially overlapping regions [3]. However, that model requires a longer training time, and a larger set of manually annotated images to accurately detect blobs of overlapping cells. Finally, the global data association technique used in the cell tracker module should be able to correctly associate cells if the overlap time is short, such as in trajectories that are crossing and then continuing in opposite directions.

3.4 Learning the classifier $\boxed{\text{DRAFT I}}$

The value V_i assigned to each region R_i is a classifier score that indicates the similarity of the extremal region to a cell. The classifier is trained using dot-annotated training images $\mathcal{I}^1, \mathcal{I}^2, \mathcal{I}^3, \ldots$ as follows. Let $R_1, R_2, \ldots, R^j_{N^j}$ indicate the set of N^j candidate regions extracted from the training image $\mathcal{I}^j$. The value V^j_i of each region is computed as the dot product between a feature vector $\mathbf{f}^j_i$ computed for each region and a weight vector w: $V^j_i = \mathbf{f}^j_i \cdot w$.

The learning procedure has to learn the weight vector w so that the inference procedure tends to pick regions with a single annotated cell $n^j_i = 1$. In order to consider the non-overlap constraint of regions a structured SVM [4] is used that directly optimizes the performance of the inference on the training set. The loss function that the learning framework tries to optimize counts the number of deviations from the one-to-one correspondence between the annotated dots and the selected regions:

$$L(\mathbf{y}^j) = \sum_{i=1}^{N^j} y^j_i |n^j_i - 1| + U^j(\mathbf{y}^j),$$

where n_i^j indicates the number of annotated dots in region R_i^j and $U^j(\mathbf{y}^j)$ counts the number of annotated dots that do not have a corresponding selected region R_i^j. Further details about the convex objective we try to minimize and how it can be optimized using a cutting-plain algorithm can be found in [1].

3.5 Feature selection $\boxed{\text{DRAFT I}}$

The effictiveness of the machine learning method to detect cells depends on the quality of features computed for the regions. Good features have a lot of discriminative power, so that they can effectively distinguish regions corresponding to cells to regions that don't correspond to cells. In order to find the best feature vector we have evaluated the performance of the classifier with several combinations of these features:

1. The area A of the region represented by a 10-dimensional binary vector with the entry $\lceil logA \rceil$ set to 1.

2. 10-dimensional histogram of intensities within the region.

3. The position of the descriptor in the image in terms of x-y coordinates of a centroid fitted to the descriptor.

4. Two 6-dimensional histograms of differences in intensities between the region border and a dilation of it for two different dilation radii.

5. A shape descriptor represented by a 60-dimensional histogram of the distribution of the boundary of the region on a size-normalized polar coordinate system.

6. The orientation of the descriptor after attempting to normalize its orientation.

7. The proportion of edge-pixels in the region.

Each of these features has different discriminative power, and takes some time to compute. The application of the cell detector requires that images are processed within a time limit. For this reason, we have trained and tested the algorithm with all the $2^7 - 1$ possible combinations of these features.

We have also developed a tool that helps select the most appropriate set of features given specific constraints, for example a maximum computation time, minimal precision and recall values, etc. A graph generated by the function is shown in figure 3.2.

3.6 Speeding up the algorithm NEW

review this section

The main drawback of the algorithm presented by Arteta [1] is the poor performance. The original algorithm took about 30 seconds to detect cells in a 400x400 image. Because we will be processing hours of microscopy video it was important to reduce the detection time as much as possible. The major performance improvements where achieved by addressing three things.

The algorithms needs to extract a set of feature on every single detecetd MSER. First, we have first fine-tuned the MSER detector to detect less cells, more robustly.

Second, we have identified features that are slow to compute and improved their algorithmic behaviour. One such feature is the Contour Points Distribution Histogram implemented in *cpdh.m*. The function was performing excessive calls to slow functions to extract region characteristics, and was rewritten to call these function less often, without affecting its value.

Second, several MATLAB builtin function were modified to remove excessive parameter checking, which in several cases represented an overhead of over 30%. These parameter checks are welcome when developing the algorithm, but once the algorithm is complete, several of these checks can be safely removed.

Rewrite: Not sure I can write this, since the matlab code is copyrighted

These opimizations resulted in a significant performance boost. Instead of 30 seconds, the algorithm can now detect cells on the same images in about .

Measure the number of seconds it takes to process one of these 400x400 images

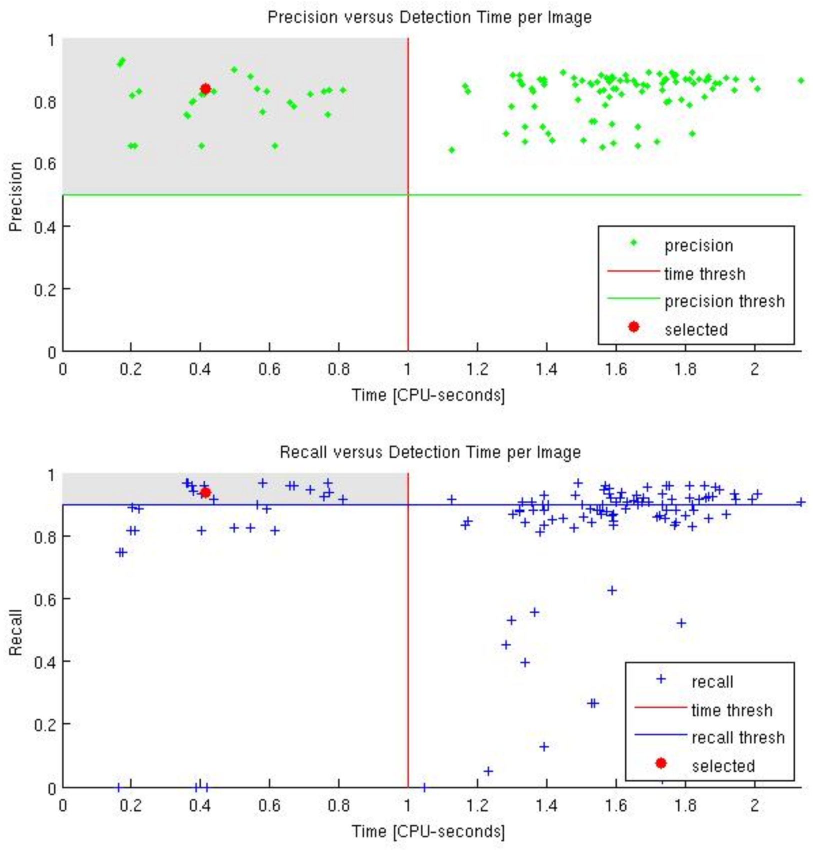

Figure 3.2: The plots helps the user to select an appropriate feature set according to the computation time per image, mean precision and recall. This example filters features sets that compute within 1 CPU-second per image (on average), have at least 0.5 precision and 0.9 recall. The feature set corresponding to the red dots contains features 1, 3 and 4. Most importance was given to high precision followed by low computation time and high recall. The selected feature set computes in about 0.4 CPU-seconds per 512-by-512 pixel image with mean precision of 0.836 and mean recall of 0.9363 (as measured on an i7-2600 CPU wish a clock frequency of 3.40 GHz).

Bibliography

[1] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect cells using non-extremal regions," in *Proceedings of the 15th International Conference on Medical Image Computing and Computer-Assisted Intervention - Volume Part I*, MICCAI'12, (Berlin, Heidelberg), pp. 348–356, Springer-Verlag, 2012. 9, 13, 16, 17, 18, 35

[2] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761 – 767, 2004. British Machine Vision Computing 2002. 14

[3] C. Arteta, V. S. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect partially overlapping instances.," in *CVPR*, pp. 3230–3237, IEEE, 2013. 16

[4] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, (New York, NY, USA), pp. 104–, ACM, 2004. 16

[5] R. Bise, T. Kanade, Z. Yin, and S. il Huh, "Automatic cell tracking applied to analysis of cell migration in wound healing assay," in *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE*, pp. 6174–6179, Aug 2011. 18

[6] R. Bise, Z. Yin, and T. Kanade, “Reliable cell tracking by global data association.,” in *ISBI*, pp. 1004–1010, IEEE, 2011. 21, 25

[7] L. Zhang, Y. Li, and R. Nevatia, “Global data association for multi-object tracking using network flows,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008. 21

[8] C. Huang, B. Wu, and R. Nevatia, “Robust object tracking by hierarchical association of detection responses,” in *Computer Vision - ECCV 2008* (D. Forsyth, P. Torr, and A. Zisserman, eds.), vol. 5303 of *Lecture Notes in Computer Science*, pp. 788–801, Springer Berlin Heidelberg, 2008. 21

[9] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona, “Fiji: an open-source platform for biological-image analysis,” *Nature Methods*, vol. 9(7), pp. 676–682, 2012. 35

[10] S. F. I. o. T. L. Philippe Thévenaz, Biomedical Imaging Group, “Point picker: An interactive imagej plugin that allows storage and retrieval of a collection of landmarks,” May 2014. 35