CVUT-CZ

Authors: Tomáš Sixta, Jiahui Cao, Jochen Seebach, Hans Schnittler, Boris Flach

Email: flachbor@cmp.felk.cvut.cz

Platform: Linux

Prerequisites: Python 3

CVUT-CZ: SUMMARY

The tracking-by-detection strategy is the backbone of many methods for tracking living cells in time-lapse microscopy. An object detector is first applied to the input images and the resulting detection candidates are then linked by a data association module. The performance of such methods depends heavily on the quality of the detector, because detection errors propagate to the linking step. To tackle this issue we propose a joint model for segmentation, detection and tracking. The model is defined implicitly in terms of a Markov chain Monte Carlo algorithm and contains a temporal feedback which allows to rerun the detector multiple times and correct detection errors using hints from neighboring frames. The detector used in this challenge is based on a multiresolution convolutional neural network but the model allows integration of an arbitrary detector which makes the method applicable to various domains. The parameters of the model are learned using an objective based on empirical risk minimization.

CVUT-CZ: DETECTION AND SEGMENTATION

Detection and segmentation is performed by a multiresolution convolutional neural network (CNN). The network is composed of three segments, each being a CNN with 3×3 convolution filters and leaky ReLU activations. The first segment is a CNN with 16 layers and the input to the first layer is an input image downscaled by factor 4. The segment is followed by a deconvolution layer, which upscales the output of the last layer by factor two. The upscaled output is then concatenated with the input image downscaled to the corresponding resolution and used as the input to the second segment (CNN with 2 layers). The output of the second segment is again upscaled by a deconvolution layer, concatenated with the (original resolution) input image and used as the input to the final segment (CNN with 2 layers). The last layer of the network predicts for each pixel its probability of being part of a cell (segmentation) and a distance B_i^{NN} to the nearest cell boundary (the distance is truncated by a threshold λ_{dist}). For more detailed description of the network please see https://github.com/tsixta/jnet.

The segmentation output is thresholded at λ_{seg} and every foreground blob is partitioned into one or several detection candidates. The partitioning algorithm has three steps: first it finds clustering seeds, then it precalculates distances between seeds andother pixels and finally it selects a subset of seeds and assigns each pixel to the nearest selected seed. The seeds are local maxima of boundary distance transform of the blob, which is obtained by calculating L_1 distance of each pixel to the nearest boundary. In the second step the distance between two pixels is defined as a cost of a shortest path between these pixels, where the cost of stepping on a pixel i is $\lambda_{\text{dist}} - B_i^{NN}$. This definition helps to create clusters that closely follow cell boundaries predicted by the neural network. The subset of seeds is selected as follows. Initially the blob is partitioned such that the discrepancy between the boundary distance transform B^{NN} predicted by the network and the actual B induced by the created clusters

$$\sum_{i \in \text{blob}} (B_i^{NN} - B_i)^2$$

is minimized. After initialization the tracker is allowed to reconsider the initial partitioning and select a different one, which is more probable with respect to the tracking model.

The neural network detector was trained using fully annotated images from the training sequences (we trained separate network for each dataset). To compensate for small number of annotated images we used data augmentation extensively (random flips and 90° rotations, elastic transforms and for **Fluo-N2DH-GOWT1** also additive intensity transforms).

CVUT-CZ: TRACKING

We formulate the tracking task as inference in an implicitly defined probabilistic model: a Markov chain Monte Carlo (MCMC) algorithm generates several samples, which characterize the underlying distribution, and the final solution is obtained by minimizing the Bayesian expected loss. The samples are constructed in a sequence of local modifications of various types, e.g. start new trajectory, modify shape of an existing trajectory in certain frame, merge two trajectories, set up a mitosis, etc.

The algorithm has two phases. The first phase is intended for initialization and the algorithm behaves in a similar way as a Gibbs sampler. In every iteration it considers all available modification proposals m and samples one of them based on their probability $p_\theta(m)$. The proposals are created using initial detection candidates only.

In the second phase the algorithm fine-tunes the sample. It does not consider all available proposals but instead for each type preselects a proposal with highest probability $p_\theta(m)$ among proposals of that type. This may include reruning the detector – in some cases the initial detection step mistakenly merges one cell with another or splits a cell into several parts, but detects it correctly in a neighboring frame. This serves as a hint for the sampler to reconsider partitioning of the corresponding blob and come up with appropriate modification proposals. One of the preselected proposals is then sampled based on their probability $p_\theta(m)$. Parameters θ of the probabilistic model were learned automatically using an objective function based on empirical risk.

CVUT-CZ: POST-PROCESSING

We enhanced the final tracking results by filling holes in tracked cells. Furthermore due to the restricted field of interest in **Fluo-N2DL-HeLa** we pruned the final results by removing cells whose centroids were less than 12.5 pixels from the image boundary.