A discriminatively trained fully connected conditional random field model for blood vessel segmentation in fundus images

José Ignacio Orlando, Elena Prokofyeva and Matthew Blaschko

README - Code version 1.0

Hi all! This is a very first attempt to release our code for blood vessel segmentation in fundus images using our method as published in [1]. If you use this code, please cite our papers [1] and [2].

Take into account that this is code that was used only for experimentation. We are working now on a refactoring of this code, which is going to be simpler to use and will allow users to learn their own models within this framework. Stay tuned!.

1 Set up your data sets

Our method requires to read the data from a single folder, structured in the following way:

- <**dataset-name**>
  - **training**: this folder has to contain data used for training.
    - **images**: RGB images to be used for training.
    - **masks**: FOV binary masks.
    - **labels**: binary ground truth masks of the vessels.
  - **validation**: this folder has to contain data used for validation (that is, model selection).
    - **images**: RGB images to be used for model selection.
    - **masks**: FOV binary masks.
    - **labels**: binary ground truth masks of the vessels.
  - **test**: this folder has to contain data used for test (that is, data to evaluate the algorithm).
    - **images**: RGB images to be used for evaluating the learned model.
    - **masks**: FOV binary masks.
    - **labels**: binary ground truth masks of the vessels.

Additionally, you will need to download the DRIVE data set[1], which will be used to estimate the scaling factor. Remember to organize the data set as explained before, with that structure.

[1] `http://www.isi.uu.nl/Research/Databases/DRIVE/`

2 Set up the system

A number of variables are hardcoded with paths and data sets names in the `script_experiments_massi` script, which is the one responsible from running our system. Here we list the variables that you must modify to use our code.

1. `datasetsNames` is a cell array containing a list of data sets that will be segmented. You must put the name of the data sets you will use in this variable (remember, the name has to be the same that the name of the folder containing the data set).
2. `learnC` is a binary value indicating if the value of C is preset or if it is going to be adjusted using the validation set.
3. `crfVersions` is a cell array containing the names of the models that will be used. Possible values are:
   - `up`: only the unary potentials.
   - `local-neighborhood-based`: the local neighborhood based CRF model.
   - `fully-connected`: the fully connected CRF model.
4. `cValue` is a double variable indicating the value of C to be used. If you already set `learnC`= 1, `cValue` will be ignored.
5. `rootDatasets` is a string indicating the path where all the data sets are stored.
6. `rootResults` is a string indicating the folder where all the results will be saved.

The function `getConfiguration_GenericDataset` in `Configuration` folder is used to adjust other parameters. Here we describe each important variable and their responsibilities:

1. `estimateScaleFactor` is a function that estimate the scale factor based on a given data set with FOV masks. You should modify the path to retrieve the images of the DRIVE data set, organized as explained in the previous section.
2. `config.learn.modelSelection` is a binary value indicating if feature selection has to be performed and if the θ_p value has to be adjusted. We strongly advice you to set this variable to 0 and use our configuration.
3. `config.learn.theta_p` is a binary value indicating if θ_p has to be adjusted. If you already set `config.learn.modelSelection` to 0, this value will be ignored.
4. `config.theta_p.initialValue` is an integer indicating the first value of θ_p to analyze.
5. `config.theta_p.lastValue` is an integer indicating the last value of θ_p to analyze.
6. `config.theta_p.increment` is an integer indicating the increment of θ_p. Thus, if you already set `config.learn.theta_p` to 1, our code will use grid search to evaluate values from `initialValue` to `lastValue` with `increment` increments.
7. `config.C.initialPower` is an integer with the initial power to be considered while searching for the best value of C.
8. `config.C.lastPower` is an integer with the last power to be considered while searching for the best value of C.
9. `config.compute_scores` is a binary value indicating if the likelihoods are going to be computed and save. Be careful, as the size of the final files can be increased significantly.

3 Run!

If you successfully arranged the data sets and modified the corresponding variables, then you are ready to run our system by launching the `script_experiments_massive` script.

References

[1] Orlando, J.I., Prokofyeva, E., Blaschko, M.B.: A discriminatively trained fully connected conditional random field model for blood vessel segmentation in fundus images. Biomedical Engineering, IEEE Transactions on (2016)

[2] Orlando, J.I., Blaschko, M.B.: Learning fully-connected CRFs for blood vessel segmentation in retinal images. In: MICCAI. (2014) 634–641