TOPICS IN VIRTUAL REALITY: MATHEMATICAL METHODS FOR VISUAL COMPUTING

EXERCISE 4 - GRAPH CUT

Handout date: Monday 12 November 2018
Submission deadline: Monday 26 November 2018, 17:59
Demo date: TBA

General Rules

Plagiarism note. Copying code (for example, from other students, external sources and previous years) is strictly prohibited! We will be using automatic anti-plagiarism tools, and any violation of this rule will lead to expulsion from the class.

Late submissions will not be accepted, except in case of serious illness or emergency. In that case please notify the teaching assistants.

Software. All exercises of this course use the MATLAB programming language. The MATLAB distribution is available from `http://kftp.kaist.ac.kr/` for KAIST students. See our MATLAB tutorial slides for hints or specific functions that could be useful for your implementation.

What to hand in. Send a .zip file of your solution by email to the TA and lecturer (as a download link, **not** in attachment). The emails will be processed automatically. The *zip file* must be called "MathMethods18-Ex*-StudentID-firstname-familyname.zip" (replace * with the assignment number, e.g. MathMethods18-Ex01-20180503-John-Smith.zip). The *email subject* must be the same (except ".zip") (e.g. MathMethods18-Ex01-20180503-John-Smith). The .zip file MUST contain *a single folder* called the same as the email subject (e.g. "MathMethods18-Ex01-20180503-John-Smith") with the following data inside:

- A folder named "code" containing your MATLAB code.
- A README file (in pdf format) containing a description of what you have implemented and instructions for running it, as well as explanations/comments on your results.
- Screenshots of all your results with associated descriptions in the README file.

Grading. The homeworks count for 50% of your total grade. Your homework submission will be graded according to the quality of the images/results produced by your program, the conformance of your program to the expected behaviour of the assignment, and your understanding of the underlying techniques used in the assignment. The submitted code must produce exactly the same images included in your submission.

To ensure fairness of your grade, you will be asked to briefly present your work to the teaching assistants. Each student will have about 5 minutes to demo their submission and explain in some detail what has been implemented, report potential problems and how they tried to go about solving them, and point the teaching assistants to the code locations where the various key points of the assignments have been implemented. See above for the scheduled demo date for this particular assignment.

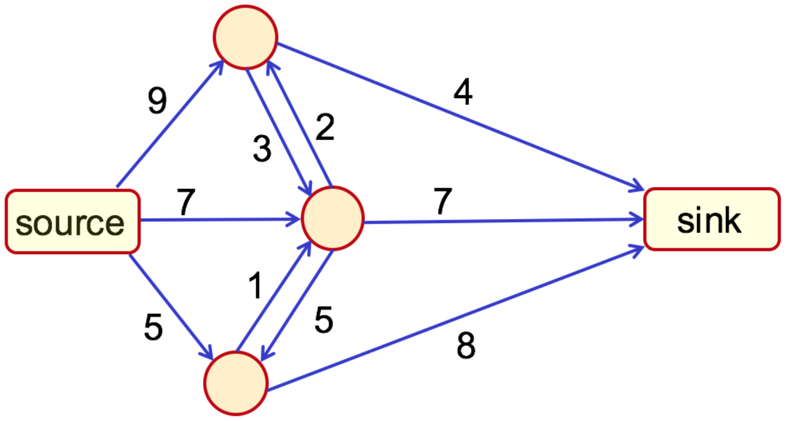


Figure 1. Graphical model.

1. Interactive Segmentation with Graph Cut

In this exercise you will apply what you learned about data term, smoothness term, graph cut and max flow algorithm.

Segmentation is one of the most fundamental application of computer vision. It consists in partitioning an image into segments (a segment being here a set of pixels). A basic task for segmentation is the separation of foreground objects from background. An example of application is medical imaging, where physicians need to be able to immediately detect organs in a scan.

Fully automatic segmentation is difficult to achieve without training a classifier on large datasets. A tradeoff is to consider interactive segmentation. The user labels parts of the image as foreground and parts as background, and lets the computer figure out which segmentation is the best.

In this exercise, you will be asked to implement an interactive segmentation tool inspired by [1].

Since segmenting foreground from background is a binary labeling problem, we can use graph cut methods to find the optimal labeling. To do so, one needs to build a proper graph that correspond to the problem and define the corresponding unary and pairwise cost. We then look for the minimal cut of the graph which gives the optimal labeling.

1.1. **Task 1 - Handling Max Flow.**

As seen in the lecture, looking for the minimal cut is equivalent to looking for the maximal flow through the graph. In [2], Boykov and Kolmogorov presented a powerful algorithm for computing max flow. You will use an existing implementation of this algorithm, provided with the homework.

In this task, you will use the max flow implementation that we provide you in order to solve the trivial graph represented in Figure 1. It will permit you to understand how to use the library in order to apply it for the interactive segmentation task.

- Step 1: run BK_BuildLib.m in order to build the library. Some compilation errors have been reported to us. See the section at the end of this handout for troubleshooting.

- Step 2: create the graph corresponding to Figure 1.
- Step 3: solve it and extract the optimal labeling.
- Step 4: solve it manually and verify your answer.

Hints:

- Here we consider that the source S is associated with the label 0 and the sink T with the label 1.
- You will need the following functions, BK_Create, BK_SetUnary, BK_SetPairwise, BK_Minimize, and BK_GetLabeling.
- BK_SetPairwise asks you to provide a penalty e_{00} and a penalty e_{11}, i.e. a cost for having two neighbouring nodes with label 0, and a cost for having them both with label 1. This is useful when we want to favor some labeling over another when we don't have much data (e.g. favor 0 over 1). In this case, we are not interested in such behaviour, so you should set e_{00} and e_{11} such that $e_{00} = e_{11}$, and such that the submodularity of the pairwise term is respected.
- Matlab returns labels 1 and 2, instead of 0 and 1.

Deliverables: You need to provide the following:

- Matlab code that computes the labelling.
- Draw the labelling result on top of Figure 1, i.e. write the label of each node and draw the cut. You can draw the graph and the result by hand, and then insert a photo in your report.
- Draw the first and the last iterations of the Max Flow algorithm performed manually. You can insert a photo in your report.

1.2. **Task 2: Interactive Segmentation.** You will use the max flow library to implement an interactive segmentation algorithm. We provide you the code which creates the interactive interface and extracts the position of points that are labeled by the user. Your task will be to focus on creating the graph using this information, solving the graph, and extracting the optimal labeling.

To do so, you will need to read [1], especially section 3 and the introduction of section 4 in order to understand what the unary and pairwise costs should be. Additional information is available in the comments at the top of the provided file interactiveGraphCut.m.

- Step 1: build a function that creates a color histogram using data from a given set of pixels.
  - Choose a resolution of 32 for your color histograms in order to have consistency (see hints).
  - Before normalizing your histogram, you should smooth it in order to make it more general. In Matlab, 3D histograms can be smoothed using the function *smooth*3. Use gaussian smoothing with filter size 7.
- Step 2: build a function that create the unaries used in [1], using the color histograms you have previously built.

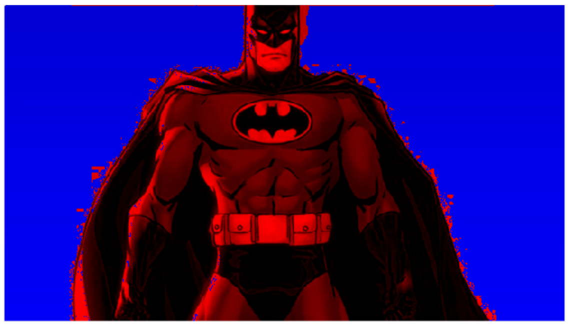
- Step 3: build a function that creates the pairwise terms used in section 4 of [1]. Use the formula from section 4 of [1], with $\sigma = 5$.
- Step 4: build and solve the graph. Produce segmentation images like Figures 2 and 3.
- Step 5: Use the obtained segmentation to change the background of your image.

Hints:

- Each of the RGB channels in the color images can take a value between 0 and 255, which means that there are 256 possible bins that can be filled in order to create the histogram. This is too large for proper generalization considering that we only use few pixels from a single image to build histograms. This is why we choose a resolution of 32.
- However, some entries might still be never filled. Before applying the logarithm to your histograms when you build the unaries, you should also add a small value, such as 10^{-10}.
- Since the graph is undirected, you can use BK_SetNeighbours instead of BK_SetPairwise (not mandatory though, both work). In this case you need to build what is called an adjacency matrix. The dimension is $N \times N$ where N is the number of pixels, and the entry (i, j) has the value of the weight of the edge between pixel i and j if they are neighbours, 0 if not. This matrix will be too large for dense storage. You will need to use the *sparse* function of matlab with the following configuration : $S = sparse(i, j, s, m, n)$, where:
  - i and j are the location of values s in the matrix such that $S(i(k), j(k)) = s(k)$.
  - m and n are the dimensions of S
- In [1], the authors define a value $K = 1 + \max_{p\in\mathcal{P}} \sum_{\{p,q\}\in\mathcal{N}} \mathcal{B}_{p,q}$ for the unaries of pixels marked by the user as foreground (resp. background) if they are linked to the source $\mathcal{S}$ (resp. the sink $\mathcal{T}$). It is supposed to make sure that these pixels will not be marked as background (resp. foreground) after optimizing. You don't need to compute K, you just need to set it very high. The easiest way is to set it to infinity, by using matlab *inf* command.

1.3. **Written output for Part 2.**

- Optimal labeling for the graph of task 1.
- Code for getting color histograms.
- Code for building the unaries and the pairwise terms of the graphs.
- Segmentation of the provided images (Batman and Van Damme), using the provided scribbles for foreground and background, and the following parameters:
  - The Batman picture with $\lambda = 1.0$, and with $\lambda = 0.002$.
  - The Van Damme picture with $\lambda = 1.0$, and with $\lambda = 10^{-4}$.
- The Batman and Van Damme pictures with a new background. Be creative ;-)
- Segmentation of an image chosen by you.


(a) Batman $\lambda = 1.0$

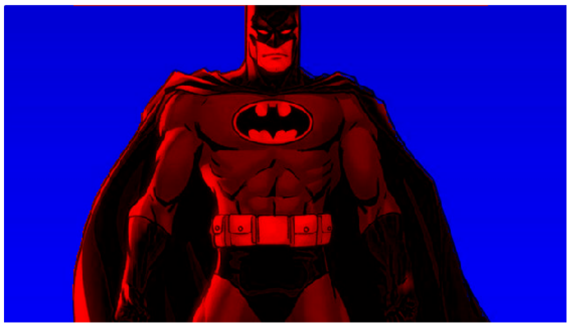
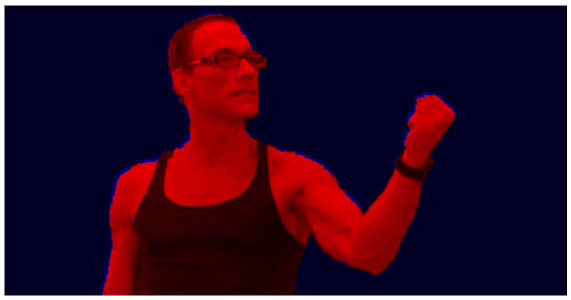

(b) Batman $\lambda = 2 \cdot 10^{-3}$

Figure 2. Segmentation results for Batman.


(a) JCVD $\lambda = 1.0$

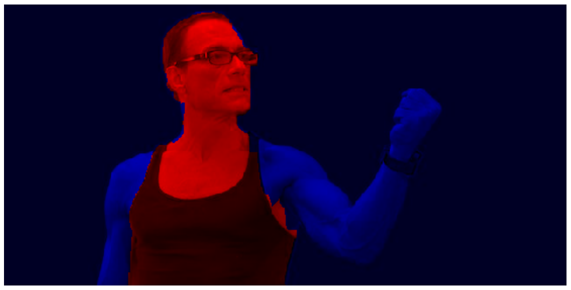

(b) JCVD $\lambda = 10^{-4}$

Figure 3. Segmentation results for Van Damme.

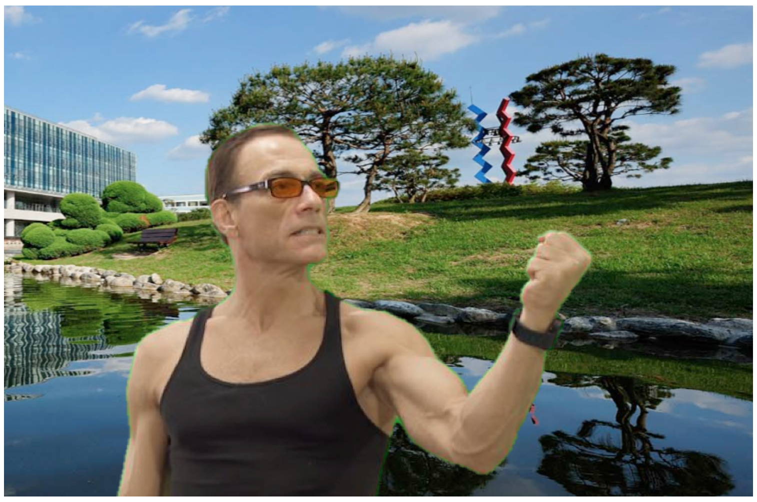


Figure 4. New background: Van Damme at KAIST.

Troubleshooting with the max flow algorithm. We have been reported that the following errors might occur when running BK_buildLib.m :

- Error using mex
  No supported compiler or SDK was found. For options, visit
  http://www.mathworks.com/support/compilers/R2015b/maci64.html.
  Error in BK_BuildLib (line 65)
  eval(mexcmd);
  - If you are using windows, the following link might help you
    http://ch.mathworks.com/matlabcentral/answers/141184-error-using-mex-no-supported-compiler-or-sdk-was-found
  - If you are using MAC, the following link might help you
    http://ch.mathworks.com/matlabcentral/answers/246507-why-can-t-mex-find-a-supported-compiler-in-matlab-r2015b-after-i-upgraded-to-xcode-7-0
- Error using mex
  bk_matlab.cpp
  /your/working/directory/GraphCut/bk_matlab.cpp(10):
  error C2371: 'mwSize': redefinition; different basic types
  /usr/local/matlab/r2016a/extern/include/tmwtypes.h(795): note: see declaration of 'mwSize'
  /your/working/directory/GraphCut/bk_matlab.cpp(11):
  error C2371: 'mwIndex': redefinition; different basic types
  /usr/local/matlab/r2016a/extern/include/tmwtypes.h(796): note: see declaration of 'mwIndex'
  Error in BK_BuildLib (line 65)
  eval(mexcmd);
  - In the file bk_matlab.cpp comment lines 9 (#if defined(MX_API_VER)...) to 12 (#endif). Then add the following header:
    #include <tmwtypes.h>

If you still face compilation errors, contact TA.

References

[1] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *International Conference on Computer Vision (ICCV)*, 2001.

[2] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(9), 2004.