Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

MATHEMATICAL FOUNDATIONS OF COMPUTER GRAPHICS AND VISION

EXERCISE 4 - SAMPLING PATTERNS AND GRAPH CUTS

Handout date: 02.04.2019
Submission deadline: 15.04.2019, 23:59
Demo date: 16.04.2019, at the exercise session

General Rules

Plagiarism note. Copying code (either from other students or from external sources) 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 assistants and provide a relevant medical certificate.

Software. All exercises for this exercise are to be implemented in the MATLAB or Python programming language. The MATLAB distribution is available from IDES for ETH students. See the exercise session slides for hints or specific functions that could be useful for your implementation.

What to hand in. Upload your solution in a .zip file on Moodle. The file must be called "MATHFOUND19-*-firstname-familyname.zip" (replace * with the assignment number). The .zip file MUST contain a single folder called "MATHFOUND18-*-firstname-familyname" with the following data inside:

- A folder named "code" containing your MATLAB or Python code
- A README file (in pdf format) containing a description of what you've 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. This homework is 8.3% of your final grade. Your submission will be graded according to the quality of the images produced by your program, and the conformance of your program to the expected behaviour of the assignment. The submitted code must produce exactly the same images included in your submission. Code that does not run will receive a null score.

To ensure fairness of your grade, you will be asked to briefly present your work to the teaching assistants. Each student will have 3-4 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 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.

Note on programming language

For this exercise, you have the choice of your programming language: you may use either MATLAB or Python. Select one language and use it for all the exercises of this homework. Extra codes necessary for the exercises are provided in both languages. For Python, you should install the following libraries:

- Numpy
- Matplotlib
- Pillow (PIL)
- Scipy
- Tkinter

1. PART 1: Analyzing Sampling Patterns

In this part of the exercise, we will implement two analysis techniques to analyze sampling distributions used for representation and integration of functions.

1.1. **Task 1: Computing periodograms of sampling patterns.** A classical approach for analyzing sampling patterns is using the so-called periodogram of a point distribution. The periodogram is computed by taking the Fourier transform of the impulse process corresponding to the sampling pattern. It is estimated by $P(\omega) = |\mathcal{F}\left[\frac{1}{n}\sum_{i=1}^{n}\delta(\mathbf{x}-\mathbf{x}_i)\right]|^2$, where δ is the Dirac delta function, $\mathbf{x}_i$'s are the locations of the points in a given point distribution, $\mathcal{F}$ denotes the Fourier transform, and $|\cdot|^2$ the magnitude of a complex number.

In practice, this is implemented by rasterizing the function $\frac{1}{n}\sum_{i=1}^{n}\delta(\mathbf{x}-\mathbf{x}_i)$ into an image, and taking the discrete Fourier transform of this resulting image. The output for each point distribution is a grayscale image. These grayscale images computed for different point distributions generated by the same algorithm are averaged to reduce the variance of the periodogram estimator. The final output for each algorithm is thus this averaged grayscale image.

We provide 10 different point distributions generated by different algorithms in different directories: *Balzer, Dart, FPO, Matern*. You will perform the following steps for each of these algorithms to generate an averaged periodogram.

Steps of this task:

- Step 1: Initialize a matrix with resolution 400×400, which represents the initial image. Fill it with zeros. Then set the pixels that the points $\mathbf{x}_i$ fall into to $\frac{1}{n}$. This image is thus an approximation of the sum of Dirac deltas at point locations.
- Step 2: Take the fast Fourier transform of the image you computed in the previous. Save the complex squared magnitude of the numbers in this image into a new image.
- Step 3: Average periodograms for the 10 different point sets for each algorithm.
- Step 4: Plot the periodogram for each algorithm by and save the resulting image (the periodogram is scaled by 200 to have a clear visualization).

Implementation Hints. Depending on the chosen language, you can use the following commands:

- MATLAB
  - *fft2* for the fast Fourier transform
  - *fftshift(fft2(imageMatrix))* to have centered periodograms
  - *imshow(periodogramMatrix * 200)* to plot periodograms
- Python
  - *np.fft.fft2* for the fast Fourier transform from the numpy package
  - *np.fft.fftshift(np.fft.fft2(imageMatrix)))* to have centered periodograms
  - *plt.imshow(periodogram * 200, cmap='gray', clim=(0, 1))* from matplotlib to plot periodograms

1.2. **Task 2: Computing the pair correlation function of sampling patterns.** Another way of analyzing point distributions recently introduced to the visual computing community is via

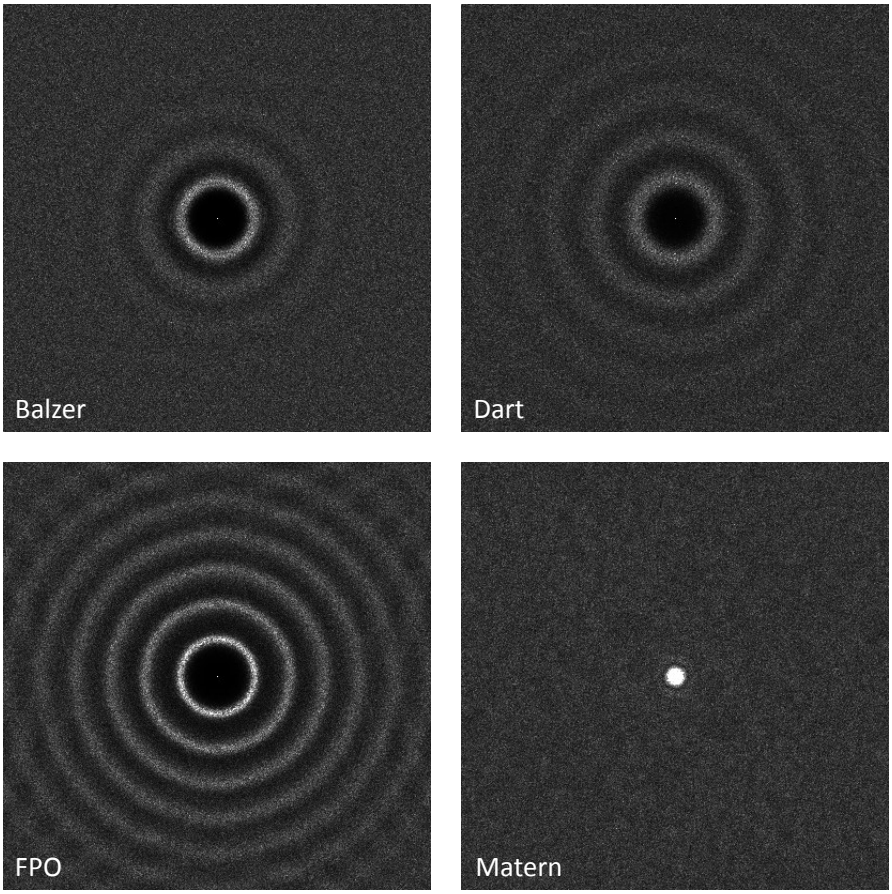


FIGURE 1. Averaged periodograms for the distributions generated by the algorithms *Balzer, Dart, FPO, Matern.*

point process statistics. In this task, we will compute the point process measure *pair correlation function* (PCF) of point distributions.

- Step 1: Read Section 3.2 of the paper *Analysis and Synthesis of Point Distributions based on Pair Correlation* [3] (pdf is included with the exercise pack), and in particular understand the estimator of the pair correlation function in Equation 2 of that paper.
- Step 2: Initialize a 1D array of size 100. This will hold the computed PCF. The first element of this array corresponds to the minimum distance r_a, and the last element to the maximum distance r_b.
- Step 3: For each element in the array, compute the PCF value via Equation 2 of the mentioned paper. Hint: $d = 2$, $|V| = 1$, $|\delta V_d| = 2\pi$, $\sigma = 0.25$, $r_a = 0.01\sigma$, $r_b = 5$.
- Step 4: Plot the resulting graphs for each algorithm.

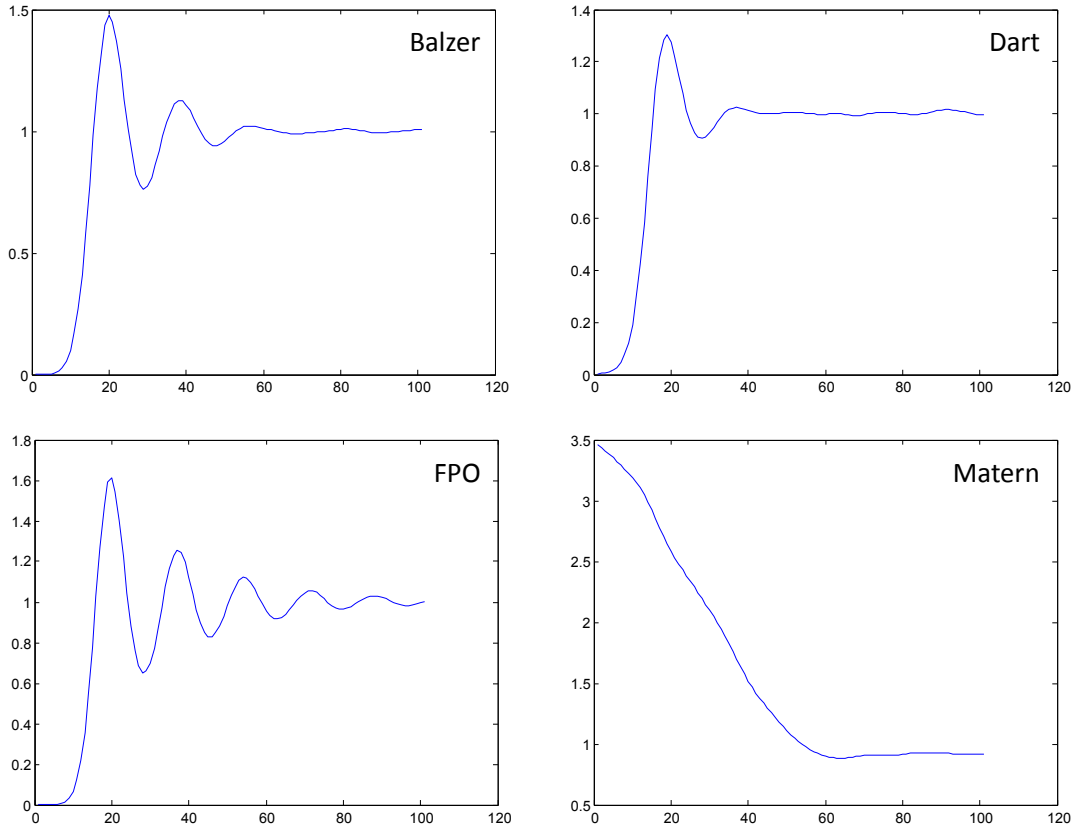


FIGURE 2. Pair correlation functions for the distributions generated by the algorithms *Balzer, Dart, FPO, Matern*.

Hints:

- When computing the PCF, don't forget to normalize your points using one of the formulas for $r_{\max}$ that are proposed in the references of the paper [3]. Also be careful that when you normalize the points samples, the domain is not the unit square anymore: you therefore need to change $|V|$ accordingly.

1.3. **Written Output from PART 1.**

- Please provide the periodograms and the PCF's for the provided algorithms.
- Discuss why we had to avereage over multiple point sets for each algorithm when computing the periodograms, but not when computing the PCF's.
- Discuss if the PCF's are sufficient to describe the provided point patterns, as they are only one dimensional, while the periodograms are two dimensional. Do the periodograms contain more information for the provided patterns?

2. PART 2: Interactive Segmentation with Graph Cut

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 the 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.

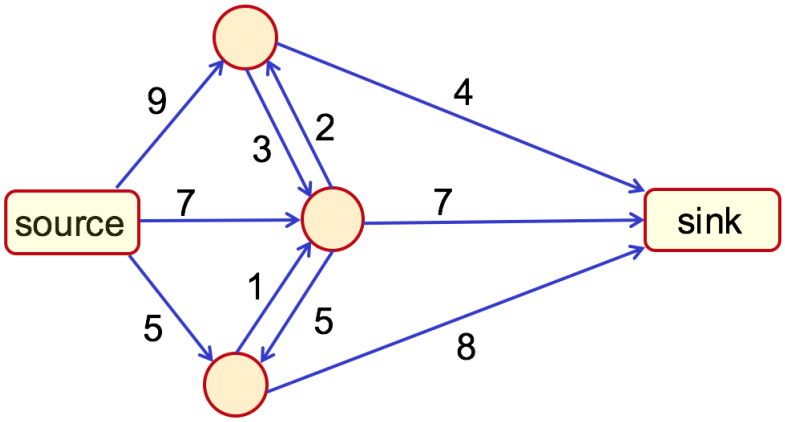


Figure 3. Graphical model

2.1. **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 gave 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 on figure 3. It will enable you to understand how to use the library in order to apply it for the interactive segmentation task.

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

Implementation Hints:

- MATLAB
  - 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.
  - 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.
- Python
  - Install the maxflow library using pip: pip install PyMaxflow.
  - Use the class GraphCut defined *graph_cut.py*. Use the methods *set_unary*, *set_pairwise*, *minimize*, *get_labeling*.

Hints:

- Here we consider that the source S gives a label of 0 and the sink T a label of 1.

2.2. **Task 2: Interactive Segmentation.** You will use the max flow library in order to implement an interactive segmentation algorithm. We provide you the code wich 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.

- 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 *smooth3*, in Python, you can use *ndimage.gaussian_filter* (from scipy.ndimage). Use gaussian smoothing with filter size 7.
- Step 2: build a function that create the unaries used in [1], using the color histograms you would 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 4 and 5.
- Step 5: Use the obtained segmentation to change the background of your image.

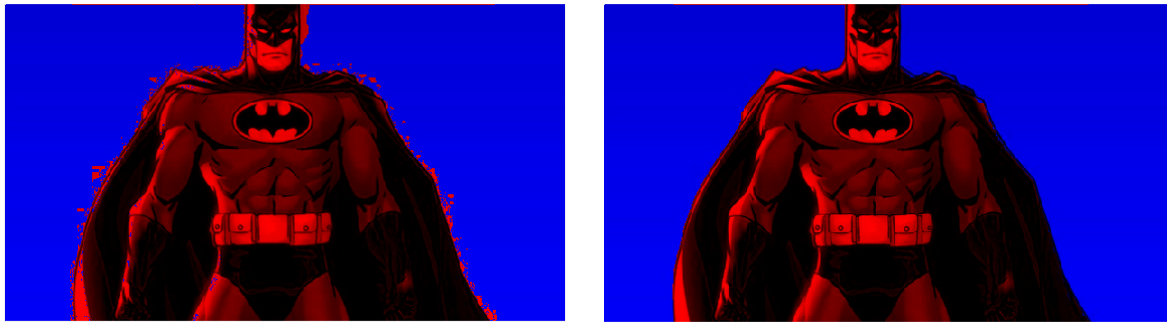
Implementation Hints.

- MATLAB
  - 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
  - Python
    - You will only have to modify *graph_cut_controller.py*. Look for the TODO comments.
    - Use coo_matrix from the scipy.sparse library to initialize large matrices. The coo_matrix has the following syntax for initialization: *coo_matrix((data, (row, col)), shape=(width, height))*

  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}.
- 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. For Python you can use numpy's inf parameter.

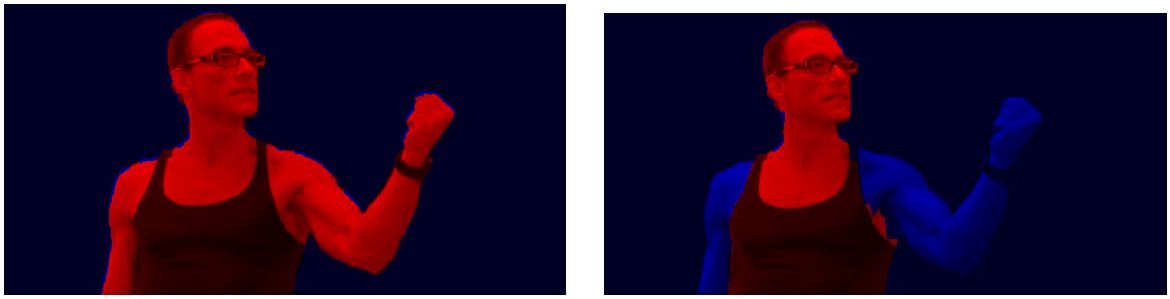
2.3. **Written output for Part 2.**

- 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 Van Damme picture with a new back ground



(a) Batman $\lambda = 1.0$ (b) Batman $\lambda = 2 \cdot 10^{-3}$

FIGURE 4. The Dark Knight Segmented



(a) JCVD $\lambda = 1.0$ (b) JCVD $\lambda = 10^{-4}$

FIGURE 5. Van Damme Segmented

- Segmentation of an image chosen by you.

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

FIGURE 6. Van Damme at ETH

- 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 Ian Cherabier (ian.cherabier@inf.ethz.ch).

REFERENCES

[1] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112 vol.1, 2001.

[2] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, September 2004.

[3] A. Cengiz Öztireli and Markus Gross. Analysis and synthesis of point distributions based on pair correlation. *ACM Trans. Graph. (Proc. of ACM SIGGRAPH ASIA)*, 31(6):to appear, 2012.