Computer Assisted Image Analysis II
Spring 2019

Excercise 2: Minimum Cost Paths, Graph Cuts, Image Registration and Motion Tracking

The framework of minimum cost paths provides a flexible way of defining the similarity, or degree of connectedness, between pairs of pixels in an image. The notion of connectedness may depend both on the spatial proximity between the pixels, and on the pixel intensity values in the image. This concept has many applications in image processing, including segmentation. Image segmentation can also be performed by graph cuts. This is a method which can be used to binarize an image into foreground and background; in this case, the image is represented as a mathematical graph.

Image registration is the process of transforming one image (commonly known as the 'floating' or 'target' image) to align it to another image ('base' or 'reference' image). The transformations can include translation, rotation, scaling, warping, mirroring and local deformations (in case of elastic registration). Image registration is an ill-posed problem. Regularization is necessary, but the optimization problem to be solved is often computationally heavy. Motion analysis has similarities with image registration since estimation of velocity could be compared to estimation of disparity. However, the two tasks differ in a number of ways, mostly due to rather strong assumptions imposed in object tracking (motion analysis).

The purpose of this exercise is to give you the opportunity to try to perform image segmentation using minimum cost paths and graph cuts. Further, you are invited to solve one image registration problem and, finally, to address a motion tracking task on a short video.

Every part of the exercise has some instructions, and sometimes also some questions. Read these **before** *actually performing the task. Answer the questions explaining* **why** *you got a particular result, rather than* **how**. *If you get stuck, do not spend too much time on a particular problem, it is better to continue and ask us later on.*

Formality

- You need an account for the computer systems at the Dept. of Information Technology.
- You should work together in groups of two people. It may be advantageous to have someone to discuss ideas and issues with.
- You are requested to prepare a written lab report in pdf format, answering all the questions and including explanatory illustrations. Only one report per group needs to be turned in. Please, do not forget to write your name(s) on the report!
- The report is handed in via the Student Portal under 'Assignments' → 'Lab assignment 2'. Make sure that you belong to a group before starting the lab. You join a group in the Student Portal by going to 'Group divisions' → 'Lab assignment teams/Project teams'.
- Status of your reports will be visible at the Student portal under 'Progress' → 'Practical part (Labs+Project)'.
- **Deadline: March 4, 2019**

Getting started

In a lab at the Dept. of Information Technology running Windows: Log on to one of the workstations. Matlab is found under `ITC-Application Explorer(Zenworks)`

Download the lab material from the Student portal, 'Documents' → 'Lab assignments'

1 Minimum cost paths

The files `GeodesicDT, MaxDT, MinDT, FuzzyConnectedness` provide implementations of Dijsktra's algorithm for different path cost functions, interfaced with *MATLAB*. This exercise will help you to understand the basic working principles of minimum cost paths.

Set up

Change the directory to `your_path/MinimumCostPaths/`. Run the script `compile.m` to build the required **mex** files.

Assignment instructions

1. Run the script `example1.m`. What does the script do? Study the resulting distance transforms, and explain the difference between them.

2. As discussed in the lecture, the geodesic and the fuzzy connectedness path cost functions can be seen as special cases of a a general path cost function

$$f(\pi) = \left(\sum_{i=1}^{k-1} w(e_{v_i, v_{i+1}})^p \right)^{1/p} . \tag{1}$$

   Modify the implementation of Dijkstra's algorithm in the file `GeodesicDT` so that all edge weights are raised to some power p, that you may define yourself. For this, you can use the `pow()` function in C++. Make sure to run `compile.m` to recompile the code after making the changes.

   Compute distance transforms for different values of $p > 1$, and compute pixelwise the p:th root of the result. Do the results agree with the theory, i.e., does the geodesic distance transform become more similar to the fuzzy connectedness distance transform as you increase p?

3. Run the file `example2.m`. The script computes an approximation of the *boolean map saliency* (BMS) of an image, by iteratively thresholding the image at randomly selected threshold values. As described in the lecture, a more efficient method for calculating BMS can be achieved using Dijkstra's algorithm. Write a script that calculates the BMS of an image, based on the function `MaxDT`. You may use the function `GetBorderSeeds` to generate a seedpoint image where all pixels on the image boundary are marked as seedpoints.

   Verify that the two implementations converge to the same result when the `number_of_iterations` for the randomized algorithm is large enough. Compare the computation time of the two methods using the `tic` and `toc` functions in MATLAB.

Report instructions

Write a short summary describing what you did, and answer all questions given in the assignment intructions above. Include pictures of the results at the various steps.

2 Segmentation using Graph Cut

The file `gc_example.m` provides an implementation of Graph Cuts algorithm interfaced with *MATLAB*. This exercise will help you to understand the basic working principles of Graph Cut. When running the example, you will notice that **mex** compilation takes place. In case you are having trouble with compiler settings, copy the mex files provided for the suitable operating system into *GCmex* directory. The example image of the parrot, as seen in Fig.1, comes with a ground-truth (GT) segmentation, where the parrot is considered the object/foreground, and everything else background. In this task, the Dice-coefficient (as seen in Lecture 6, on evaluation) will be used together with the GT to quantitatively assess the quality of the resulting segmentations.

1. ***In*** `gc_example.m` ***explain how a graph is constructed from the image? More specifically, try to correlate this example with*** `min cut-max flow` ***algorithm discussed in the lecture. What is the effect of varying parameter*** `k`***? Explain the cost term associated with labeling the nodes*** `Dc`***.***

2. ***Run*** `sourcesink.m`***; output will look like Fig.1. Your task is to add label nodes (select image patch coordinates) smartly in function*** `addSrcSnk` ***so that you can segment out only the parrot. The result is shown in Fig 2. Try to get as close to it as possible. Report the coordinates for Source-Sinks selected. Try selecting only one patch for each of the source and sink. Inspect the reported Dice-coefficient for the resulting segmentation, which is expected to surpass 0.9 at this stage.***

Figure 1: Parrot segmentation failed

Figure 2: Parrot segmented by graph cut through source and sink selection.

2.1 Optimizing the weights

So far we did not optimize the weights for source-sink, nor the edge estimates. We will do it here.

3. ***Increase*** `scsnk_wgt` ***in steps of*** 20 ***and report the value where the improvement almost stopped, in terms of Dice-coefficient.***

4. ***Pick the best*** `scsnk_wgt` ***from above and now increase*** `edges_wgt` ***from*** 0 ***to*** 10 ***in steps of*** 1 ***and report the value where best segmentation was achieved, in terms of Dice-coefficient.***

3 Image Registration

Exhaustive search

As a first assignment on image registration you will study a prepared example of an exhaustive search approach for image registration. Change the directory to `your_path/Registration` and open the file `ex_registration_exhaustive.m`. Run the code and, while it runs, go through the code line by line and make sure you understand what is happening. Do not forget to look closer at the functions that are called (e.g. `exhaustive_match.m`). The result is illustrated in Fig. 3.

Figure 3: **Top:** Three map sections from three different map services. All sections have overlapping regions but they have different rotations. Leftmost image is set as fixed image. **Bottom:** Result of a registration process.

Now answer the following questions:

1. Which similarity measure is used in the example?
2. What information in the two images is compared?
3. Translation and rotation in the example are always performed in a certain order. What is that order and why do you think it is done this way?
4. How do you locate the optimal transformation in the score space for this example?

Chamfer matching

Now we will take a look at a gradient descent approach to image registration. The biggest issue with the exhaustive approach is that it (as you now know) takes a lot of time to compute. The time consumption increases exponentially with each added degree of freedom.

An approach to registration of shapes (segmented objects in binary images) is to use a method known as **Chamfer matching**. For this method the similarity measure is the distance between two binary images (generated using e.g. thresholding or an edge detector such as Canny). The perhaps easiest way to implement this is illustrated in Fig. 4. The binary floating image (in this example the edge map) is placed on top of the distance map generated from the binary fixed (base) image. The sum of the distance values from the overlap is used as the score for that specific transformation.

Instead of evaluating all possible transformations in the allowed transformation space, Chamfer matching can be used with a gradient descent approach to find an optimal values of transformation parameters. Given a certain initial transformation (e.g. a certain position/translation and rotation), a number of predetermined transformations (e.g. move a little to the left, rotate clockwise 5° etc.) are evaluated. The transformation that yields the lowest distance is then chosen as the starting point for the next iteration. This behaviour is repeated until a predetermined stopping criterion is met.

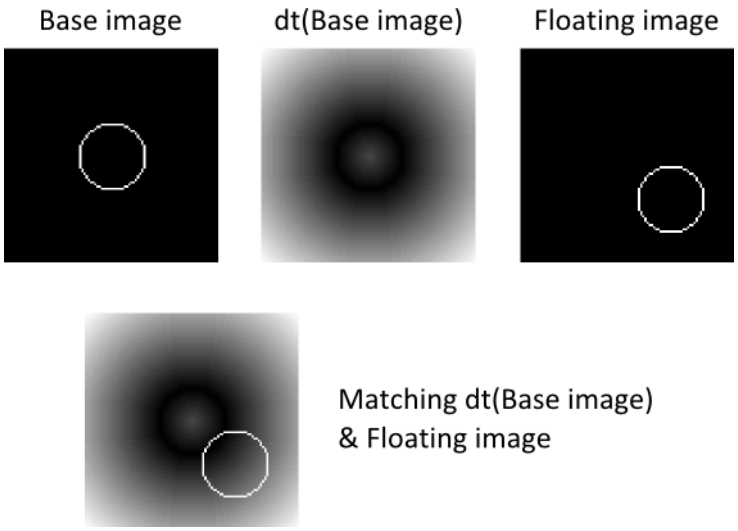

Figure 4: Key steps of the Chamfer matching process.

Your task is to implement a Chamfer matching algorithm to match the three map segments. As help you are given the file `chamferDemo.m` which basically solves the problem described in Fig. 4.

For your task you should use `01.png` as the base (fixed) image. The function `meanRGB` can be used to merge up to three images so you can see how well you have succeeded. When you are done you should have something similar to the result in Fig. 4(d) (image in the 2nd row).

For the report you should answer the following 5 questions:

1. Which image transformations are you evaluating for the floating image at each iteration and why?
2. What stopping criterion have you chosen and why?
3. Supply a plot showing the value of the distance measure for each iteration (see Fig. 5).
4. How do you locate the optimal transformation in your score space?
5. How would you avoid getting stuck in local minima (see Fig 5)?

Extra task!

If you feel that you cannot get enough of Chamfer matching, here's a challenge for you. Try matching the image 04.png to any of the other map images (01.png, 02.png or 03.png). Slightly more difficult but definitely possible.

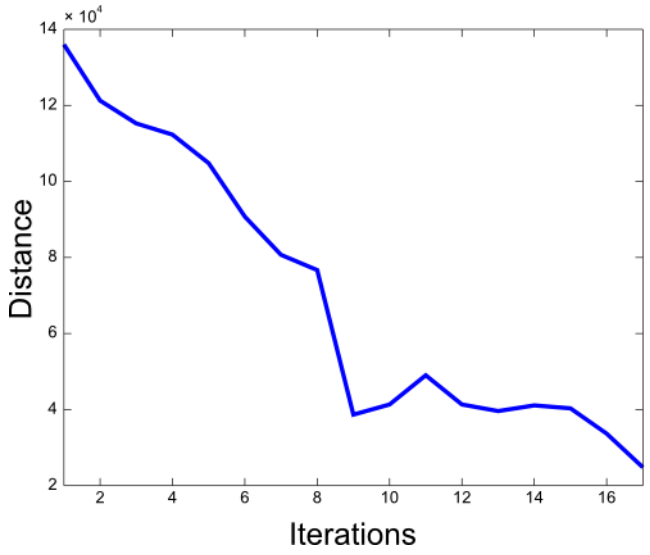

Figure 5: Plot showing the Chamfer distance against iterations in the registration process.

4 Motion Tracking

Motion analysis has some similarities to image registration, since estimation of velocity could be compared to estimation of disparity. However, in motion analysis we usually make some additional assumption, compared to image registration.

1. Which are the main assumptions usually made in object tracking?
2. What is the difference between *motion field* and *optical flow*?
3. How is optical flow used in motion analysis?
4. What is the idea behind *differential motion analysis*?
5. What are the main steps in *motion analysis by feature point tracking*?

Please, provide short answers to the questions above in your lab report.

To try some motion tracking yourself, you will be working with an image sequence depicting prawns moving around in a doughnut shaped container. The image sequence is taken from a dataset of an ongoing research project where the movement of the prawns is studied.

Change directory to `Motion`.

The image sequence you are going to analyse, named `source_sequence.avi`, consists of 581 frames. The first 250 frames could be considered as "easier" whereas the remaining frames contain a number of difficulties. To help you a little a mask (named `bwmask.png`) has been supplied that can be used to remove the central part of the frames if needed. An example result has also been supplied, named `example.avi`.

1. ***Create an algorithm that enables tracking the prawns for the first 250 frames. In the report, specify which of the previously introduced approaches you chose for this task and explain how you implemented it. Include the relevant parts of the code in the report and a video capturing the final tracking.*** Hint: ***Take a look at some of the suggested functions below.***
2. ***Now run your algorithm on the final frames of the sequence. Does it fail in any way? If it fails what is the problem? Write a short summary of the problems that arise in the second part of the sequence and suggest some possible solutions (you do not need to implement them).***

4.1 Some help

In `VideoUtilities.m` you will find a skeleton code that can help you read, play and write videos in MATLAB, as well as how to annotate frames to show the locations of the objects. The code uses objects from the vision toolbox and further information is found at

- `https://se.mathworks.com/help/vision/ref/vision.videofilewriter-system-object.html`
- `https://se.mathworks.com/help/matlab/ref/videoreader.html`
- `https://se.mathworks.com/help/vision/ref/vision.videoplayer-system-object.html`

Mind that there are other possibilities in MATLAB to play and write videos, independent of the vision toolbox with slightly different properties and documentation:

- `https://se.mathworks.com/help/matlab/ref/videowriter.html`
- `https://se.mathworks.com/help/images/ref/implay.html`

Feel free to use whatever you prefer or write your own code for the video handling!

References

[1] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, Inc, Upper Saddle River, New Jersey, 2nd Ed, 2002.

[2] Milan Sonka and Vaclav Havlac and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson Learning, 3rd ed, 2008.