Assignment #02 (Part 03)

IT-Universitetet i København
Introduction to Image Analysis and Machine Learning
(Spring 2017)

April 26, 2017

Purpose In this assignment you will explore various applications of *homographies*. The use of homographies is related to both *computer vision* and *computer graphics* applications and it is (in general) very useful for many applications. You are going to do this assignment during our exercises class, but expect also to use some time outside class to finish it. Please follow the plan closely otherwise you will be busy in the last week.

It is therefore important that you don't fall behind schedule. It may be that you will not finish everything in this part of the assignment this week, but get as far as you have time for. If you are stuck in a part of the assignment please don't hesitate to ask for assistance (also outside class hours) so that you do not waste too much time on things we could easily help you with. The reports will be a part of the exam but we are also aware that it may be slightly challenging. We expect that you do your best in solving the majority of the assignment, but if you are unable to do this, please contact us for a *Plan B* and do NOT wait until a few days before deadline.

The **report for this assignment** should be handed in by the same groups as in Assignment #01 on the **May 07, 2017**. Please try to finish one part every week as the subsequent parts are also meant to last (at least) one week. Each section specifies what kinds of images or plots you should generate. The report should be self-contained and not just answer the questions. Again for this assignment, you should possibly delegate the tasks among group members. The report should additionally be handed-in together with the other assignments by the end of the semester to learnIT. Make sure your report contains:

- Your **names** and **emails** on the front page (*Standard Front Page.pdf* file);
- The filename should be as follows: *Assignment2_GROUPINDEX.zip*; (your group index is available at learnIT > Mandatory Assignment #02 > groups)
- A link to where the code can be download; and
- The printed program (as an *Appendix*).

Your report should NOT be a list of answers to the questions given in the exercise sheet. The questions should be considered as a guide that can help you to write a good report. The report should then either directly or indirectly answer the questions. Each report should be self contained and you have to describe what is the overall method (perhaps supported with figures showing individual steps) and what your assumptions are. Remember to show that your implementation works (testing). This involves showing image sequences (multiple images), when the system works and when it does not work. A set of image sequences can be found in *Inputs* folder. This means that you have to describe in words and pictures when the method works and when it is challenged. So read through this document carefully and decide how you can work in the best possible way.

The **final report** (collection of all assignments) should be handed in to learnIT at the end of the semester and it should contain:

- Your written report, perhaps arranged nicely; :)
- A DVD with your report(s);
- Code; and
- Result videos and possibly test data.

Each mandatory assignment MUST be passed to be able to go to exam. Please see the guidelines for writing a report on learnIT. **Remember:** There is NOT only one solution to the assignment and we encourage you to be creative when solving it :)

Learning Goals The learning goals for this part of the assignment is using homographies for point transfers and texture mapping.

Expected outcome for this week Before next week, it is expected that your person map location system will be able to show the location of a person walking on the ground floor in an overview map. We encourage you to do as many as you have time for. Questions indicated with "(*Extra*)" can be skipped at first.

Get Code and Data Download *AssignmentMaterial_2.zip* from learnIT. Extract the zip file and copy the contents into the source folder of your project. The *Assignment2.py* file contains the code skeleton for the assignment. You are naturally free to make changes to the file as you prefer. The *SIGBTools.py* file contains various auxiliary functions that may become helpful in solving the assignment. Use the OpenCV reference manual or the online documentation for additional information and help about the OpenCV functions.

Tasks, Exercises and Numbers The text contains several questions that will guide you through the assignment. *Tasks* are activities to test or improve your system. *Exercises* are mandatory activities related to development of the person map location system and linear texture mapping system. *Extra Exercises* are optional activities can help you to achieve a better solution if time permits. Individual questions marked with "(*Extra*)" are not required (e.g. if time is short).

Output of this Assignment You are going to send us a report and image sequences showing how your system works. Some exercises have information about the videos (e.g. *filenames*) you must record to show us the partial results of your code. Have in mind that we do not expect you to develop a method that works in every image sequence automatically, nor do we expect your method to be flawless. However, we do expect that the methods work in most of the frames. In your report, you should describe what you did, why it works and why it fails. It is allowed to be ambitious and try to develop generic and flawless methods (this is hard).

The report should contain images with the results of your method. **Show several images of both when the method fails and when it gives good results**. Use figures and the images to describe the methods and to analyze the methods.

Remember that the final report that is handed to learnIT MUST contains image sequences of your results on a digital medium.

Important Check if you have filled out all combo boxes illustrated in the first page (i.e. requirements for assignment report) and in the document (i.e. output files generated by the developed exercise).

Exercise 2.01. *Person Map Location* – In this exercise you will make a component to show where a person is walking on an overview map. You do not need do the tracking your self, as the tracking data has been provided. But you have to show the location of a person walking on the ground floor in an overview map. However, please feel free to experiment with background subtraction and connected components or any other of the techniques that you have heard about in the course to improve the results.

For the report In the report, you should specify your assumptions on how you uses homographies to map distinct two-dimensional planes. Show the path where a person is walking in the overview map, e.g. in a series of not necessarily consecutive images. In the report you have to discuss under which circumstances your method fails/works and why?

Outcome The outcome of this exercise is a figure or movie where it is possible to see where the person is walking (i.e. the path) in the overview map (based on ITU building).

Data The image sequence (*ITUStudent.avi*), the corresponding tracking data (*trackingdata.dat*) and the map of ITU building (*ITUMap.bmp*) can be found in the *AssignmentMaterial_2.zip* from learnIT. Some functions have been provided in *Assignment2.py*. You can start with making changes to this file, but you may want to make additional file to get a better structure of your code.

The file *trackingdata.dat* can be loaded in Python by `loadtext("trackingdata.dat")`. Each row in the resulting matrix contains the coordinates of three rectangles in the corresponding image in the sequence (i.e. *row 1* contains the coordinates of the rectangles in image 1, and *row 2* the rectangles in image 2, etc.). Each rectangle corresponds to a subregion of the person being tracked where a set of coordinates pairs (x_1, y_1) and (x_2, y_2) are the corners of one rectangle. Hence, the rows in *trackingdata.dat* contain three rectangles (e.g. 12 numbers), according shown in Figure 1. The private method `__showFloorTrackingData()` in *Assignment2.py* shows how to load and display the tracking data.

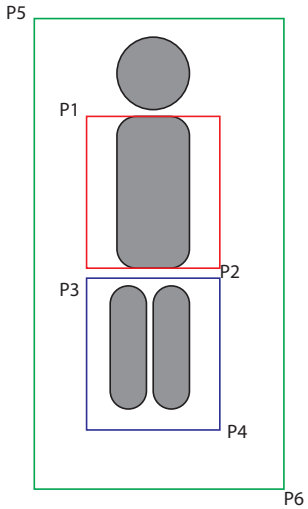

Figure 1: Example of three sub-regions of the person being tracked.

In the following, the image sequence *ITUStudent.avi* will be denoted S, the ground floor in S will be denoted G and the map of ITU building will be denoted M, as shown in Figure 2.

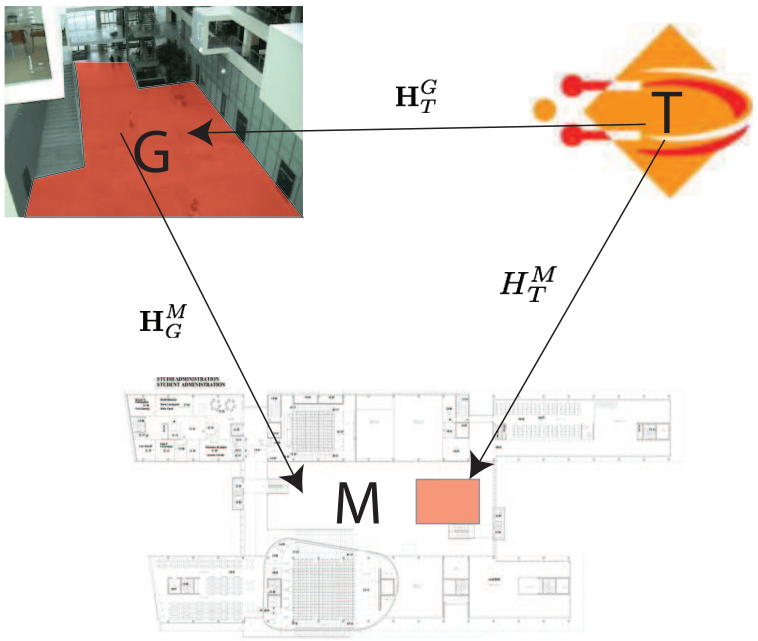

Figure 2: Naming of planes and transformations.

(a) Run the function `__showFloorTrackingData()` in *Assignment2.py*. You should now see 3 boxes surrounding the person walking on the ground floor.

(b) The transformation from the ground floor, G in S to the overview map M is defined by a homography – why?

(c) How many corresponding points are needed to estimate a homography?

Initially, you have to calibrate the system so that points on the ground floor (G) can be mapped to the overview map (M) via the homography H_G^M, as show in Figure 2. That is, the points in the ground floor and the overview map that correspond to the same points in space should somehow be defined. The corresponding points can be used to estimate the homography between planes in the two views. In the following, you will be selecting the corresponding points manually via mouse clicks.

(d) Read the help for the function `SIGBTools.GetHomographyFromMouse()` in a Python prompt.

(e) Have a look at the code for `GetHomographyFromMouse()` method in `ImageManager` class to get an overview of what it does. You will perhaps later need code fragments of the function in your own implementation.

(f) Use the first image (or any image you like favorite one) in the image sequence (S) and overview map (M) to find the homography (H_G^M) from ground floor (G) to the overview map (M). You can use `SIGBTools.GetHomographyFromMouse(I1, I2, N)` function to select the corresponding points in the two images.

(g) Why is only one calibration with four points needed for the entire sequence?

(h) What happens if the camera or ground floor move relative to each other?

(i) Modify the function `__showFloorTrackingData()` to use the estimated homography and display the trace of the person in the overview map. *Notice:* you should only map points that are on the ground floor (G) to the overview map (M) (e.g. the feet and not the head). The example function `__ShowImageAndPlot()` should give you sufficient information on how you can display and save the image data.

(j) Extend `__showFloorTrackingData()` function so that it saves the result into an image. The images should be used in the report.

- **Output:** A image with the results of entire trace of the person in the overview map. You should name this image as: "*mapImage.png*".

(k) Extend `__showFloorTrackingData()` function so that it also saves the homography (H_G^M) in a file. *Hint:* use `save()` and `load()` functions from Numpy to save and load numpy arrays.

- **Output:** A numpy file contains the homography matrix. You should name this file as: "*homography1.npy*".

(l) Extend `__showFloorTrackingData()` function that for each image in the sequence shows the current position of the person in the overview map (M), e.g. like a real-time system.

(m) Extend `__showFloorTrackingData()` function to save a video showing the current frame in the sequence and the current location of the person in the overview map (M). Add this video to the final report for the examination.

- **Output:** A video contains each step walked by the person in the overview map. You should name this video as: "*MapLocation.wmv*".

Exercise 2.02. *Linear Texture Mapping* – In this part you will develop some components to add the ITU logo as texture mapping over some specific surfaces in the analyzed images.

For the report In the report, you should specify your assumptions on how you uses homographies to create texture mapping. Show some image sequences in which you added the ITU logo in the analyzed images. In the report you have to discuss under which circumstances your method fails/works and why?

Ground Floor The `__SimpleTextureMap()` function in *Assignment2.py* file is an example of how linear texture mapping can be done using OpenCV. Run `__SimpleTextureMap()` function and use it to texture map the large area in front of the auditorium in the overview map with the image of the ITU logo.

(a) Improve the `__TextureMapGroundFloor()` function that places your favorite texture on the ground floor (G) for each image in the image sequence (S). Notice when setting the N parameter to negative values in `SIGBTools.GetHomographyFromMouse(I1, I2, N)` that the method assumes the corners of the `I1` image are used as input. In this way, you only have to select points in the destination image to perform the texture mapping.

- □ **Output:** A numpy file contains the homography matrix. You should name this file as: "*homography2.npy*".

(b) Change the weights when adding the textures so that the texture appear as nicely as possible.

- □ **Output:** A video contains a texture mapping over the ITU ground floor. You should name this video as: "*TextureMapGroundFloor.wmv*".

Exercise 2.03. *Linear Texture Mapping* – In this part you will develop some components to add the ITU logo as texture mapping over some specific surfaces in the analyzed images.

For the report In the report, you should specify your assumptions on how you uses homographies to create texture mapping. Show some image sequences in which you added the ITU logo in the analyzed images. In the report you have to discuss under which circumstances your method fails/works and why?

Moving Objects In the following, you will experiment with texture mapping on image sequences where the objects move. The *GridVideos* available on *Videos* folder (i.e. `Grid\_0X.mp4`) contains a few video sequences in which there is a grid pattern which should be texture mapped with the *ITULogo.png* (available on *Images* folder). The function `__TextureMapGridSequence()` shows how to detect the grid in an image sequence.

(a) Perform automatic texture mapping on the image sequences in the directory.

- **Output:** Five videos that contain the texture mapping over the chessboard pattern. You should name these video as: "*TextureMapGridSequence_Grid0X.wmv*", which $1 \leq X \leq 5$.

(b) In some of the frames, it appears that the texture map fails yet it does so in a consistent way. What happens?

Assignment #02 (Part 03) [*Extra*]

IT-Universitetet i København
Introduction to Image Analysis and Machine Learning
(Spring 2017)

April 26, 2017

Exercise 2.03. *Linear Texture Mapping* – In this part you will develop some components to add the ITU logo as texture mapping over some specific surfaces in the analyzed images.

Moving Objects In the following, you will experiment with texture mapping on image sequences where the objects move. The *GridVideos* available on *Videos* folder (i.e. `Grid\_0X.mp4`) contains a few video sequences in which there is a grid pattern which should be texture mapped with the *ITULogo.png* (available on *Images* folder). The function `__TextureMapGridSequence()` shows how to detect the grid in an image sequence.

(c) $^{(Extra)}$ Develop a way of solving the problem so that the texture is mapped even during rotations.

- **Output:** Five videos that contain the improved texture mapping. You should name these video as: "*TextureMapGridSequence_Grid0X_Extra.wmv*", which $1 \leq X \leq 5$.

(d) $^{(Extra)}$ The homography maps points to the grid. But how can this homography be used to map to the corners of the paper?