Beeldverwerking Assignment 3

Amber Elferink - a.s.elferink@students.uu.nl - 5491525

Thijs Ratsma - t.ratsma@students.uu.nl - 6315887

11-11-18

Introduction

For this assignment, we wanted to create a practical algorithm that could be applied and be useful in a real-life situation. Inspiration came from FAIR2Media, an acquainted design studio which is working on a table with a beamer projecting images on top of it. In the current iteration, the projection is always the same map. Through interaction with the table, the projection would update, allowing for interaction with the software. Since hands are a natural way to interact, our goal is to develop an algorithm that could detect the position of hands above a projected table and could distinguish between pointing hands and spread hands.

Because the beamer is fixed at a certain height above the table, there is not a lot of variation in size of hands, aside from the real-life size. However, this difference in scale is near negligible. For our collection of photos, there was no variation in light conditions. Many different photos of hands were taken, with variations in rotation in relation to the table, partial occlusions caused by the projection, arms with or without sleeves and hands either pointing or spread out. We also assume only one hand at a time can be detected. This resulted in the following context table:

Criterium	**Possible values**
Minimum/maximum size	Any size, but automatically cropped to 438x264
Lightning variations	Constant, caused by beamer
Rotation variations	Multiple different rotations
Occlusion	Often partially occluded by projection
Other	Arms with/without sleeves, hands pointing/spread

Because of the very specific context of the input images, it may be difficult to find similar images to test the function on. For this purpose, we have included a collection of photos we took ourselves that fulfill all criteria of the context. While different images found online may still work, be aware that the image will automatically be cropped to a specific size. This size can be modified in the LoadAndCropImage function in Form1.aPreprocessing.cs.

The desired output is a series of points showing the position of the hand, as well as a way to differentiate between a pointing hand and a spread hand. In case there are multiple hands in a picture, only one series of points depicting one hand will be shown.

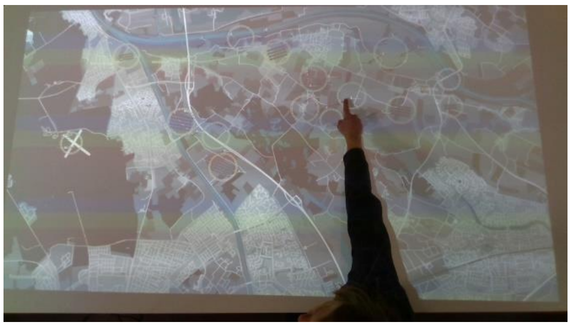


Figure 1: Example of an input photo

Pipeline

Pre-processing

All input images show parts of the table which don't have any part of the map projected onto them. Our first step was to resize the image to remove these areas and then converting the remaining image to a greyscale image. After experimenting with threshold values on multiple images and by comparing histograms, we concluded that, in most cases, the greyscale value of the hands lies between 15 and 70-90 (see Appendix A). However, because of the projection on the hands, this wasn't constant, so a fixed threshold wouldn't work. An automated threshold wasn't the solution either, as the values for the hands in the histogram were valleys compared to the peaks of the rest of the projected map. As such, the following pre-processing pipeline was decided on:

0. Resize the image and convert to greyscale.
1. Threshold the greyscale image, starting at a threshold value of 70.
2. Apply an opening filter to get rid of any noisy pixels (3x3 filter).
3. Apply a region labeling algorithm, count the number of regions and return the bounding box of the largest region.
4. If the number of regions increased compared to the previous iteration, don't update the bounding box.
5. Repeat steps 1-4 on the image derived from step 0 with an incrementing threshold value until the difference between the smallest number of regions and the current number of regions is larger than 3.
6. Return the thresholded opened cut-out of the bounding box of the largest labeled region.

This pre-processing pipeline ensures the best threshold of the hand with the least amount of noise interfering from the projected map in the background. Steps 1-4 generate a bounding box containing (presumably) the hand, while step 5 checks if the result is more suitable than the last iteration. The difference in regions of 3 was chosen experimentally, as we observed that 1 or 2 new regions are sometimes introduced without hampering the quality of the threshold too much, while 4 to 6 new regions allowed too much leeway and drastically lowered the quality of the overall threshold.

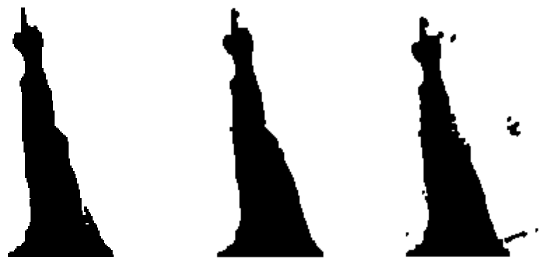


Figure 2: Opened thresholded bounding box cut-outs for thresholds of 80, 88 and 92 respectively. 88 was found to be the optimal threshold, while 92 was the point where the algorithm stopped running because more than 3 new regions were introduced.

Object recognition

After the cut-out of the largest object in the image has been extracted, the object recognition part of the function can be applied. For this, we first apply a Harris corner detection algorithm on the cut-out. Then, we use a gift-wrapping algorithm based on the set of points derived from the corner detection to determine the convex hull of the object. Then, using a boundary trace, we find the convexity defects of the object. We can then calculate the angle between a convex hull point and its two neighbouring convexity defect points. If there is only one angle below a specified value, we assume this to be a finger and identify the object as a pointing hand. If there are multiple angles below a specified value, we assume this to be multiple fingers and identify the object as a spread hand. If there are no angles below a specified value, we label the object as unidentified. If possible, we try to refine the search by finding a bounding box of 60x60 with the largest number of corners and apply all steps in the object recognition part again.

Refinement

The last step is to take the original input image and show the object in it. For this, we use the upper-left corner of the bounding box as a reference and put coloured pixels at the position of the convex hull and convexity defect points. We use green pixels for a pointing hand, yellow pixels for a spread hand and red pixels for an unidentified object. With this, the object recognition function is complete.

Reflection

The most lacking part of our function is that we assume that the biggest region described in the image is a hand. If something simple like a package is on the table while someone is pointing, the function will fail, simply because the package will describe a larger region than the hand, meaning that the object recognition part will only happen on the package. A way to fix this would be to apply the object recognition for all other regions beside the largest one as well, but we feel this couldn't have been realized for this assignment.

The function currently only works for the photos we took ourselves, because we hardcoded the removal of the unprojected table parts from the image. Next time, we would have to implement this step in the pre-processing pipeline so that images in a different context would be suitable for this function as well.

While we did use auto-contrast corrected histograms for determining the threshold required for the hand, we didn't use any form of auto-contrast correction in the function itself. This means that, while the threshold of 70-90 is optimal for auto-contrast corrected images, this may not be the case for all our non-corrected images. While this is not too big of an issue because of the way our pre-processing pipeline is structured, it is still something that could be paid attention to next time.

A big improvement to the function could be made by separating the hand from the arm it is attached to. This way, the object recognition can be applied solely to the hand itself, allowing for more detailed analysis. However, we were unable to find a way to do so with the methods currently available to us. Currently, we approximate the centroid of the hand by applying a method to the list of corners from corner detection that returns the average centre point. There are usually more corners detected in the hand area of the region, meaning that most of the time, the centre point will lie somewhere in the hand. While we did try to refine the set of convex points and convexity defects to just encompassing the hand, we were unable to do so, as the hand isolation part described at the end of the object recognition part doesn't seem to work.

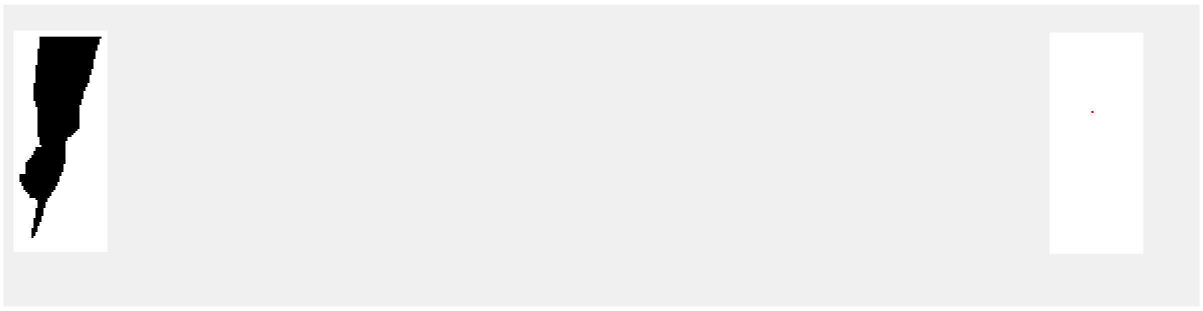


Figure 3: Centre point of all pixels in the region. Note the centre point lies on the arm instead of the hand.

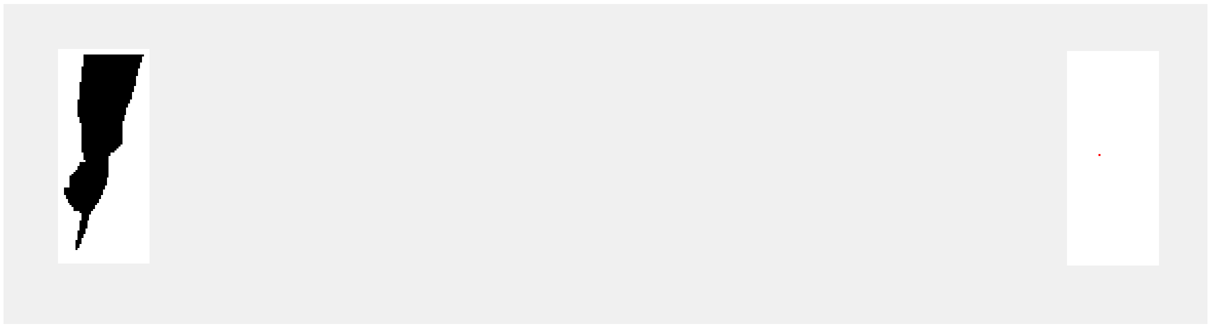


Figure 4: Centre point of the corners after applying corner detection. Note that the centre point now lies closer to the hand, but still not entirely in the centre of it.

While the full pipeline seems to be working, it's not foolproof. Oftentimes errors still occur because of a problem regarding the boundary trace looping around the image multiple times, or an inaccurate set of convex and convexity defect points is generated, causing in an inaccurate identification of the hand. This is possibly the result of the thresholding finding an incomplete region describing the hand and selecting that as the optimal threshold. While this could possibly be lessened by applying a filter like a Gaussian filter before applying the threshold, the whole pipeline still rests on too many assumptions to get it to work 100% correctly.

Results

This is what the output of the pipeline will look like. The crosses depict the collection of convex points and convexity defects, while the colour of the crosses indicate the type of object that is identified. In this case the crosses are yellow, meaning the pipeline identifies this as a spread hand. Unfortunately, this is not the case, meaning the pipeline fails to identify this hand properly. This is one of our clearer images, so we were hoping that the pipeline would at least properly identify this one. In the reflection, we already discussed what parts of the pipeline could be improved in order to obtain more accurate identifications.

The other results and discussion about those results can be found in Appendix B.

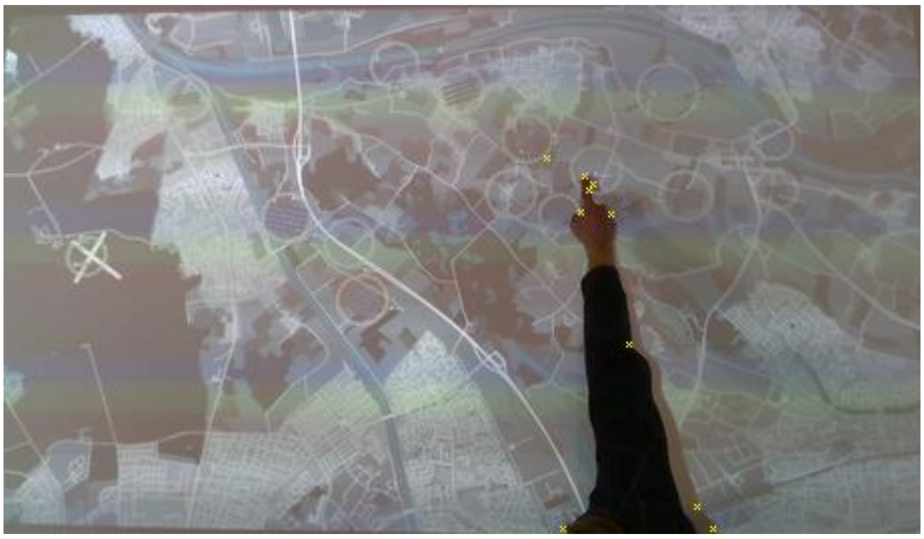
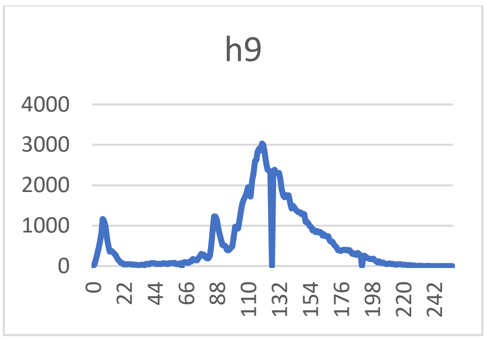
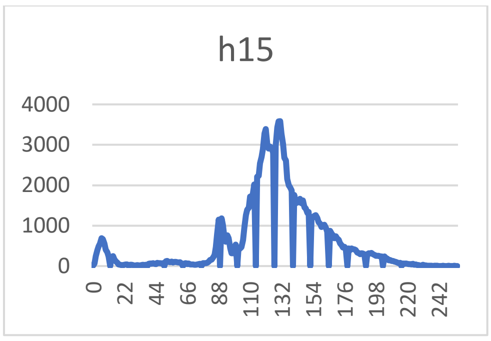


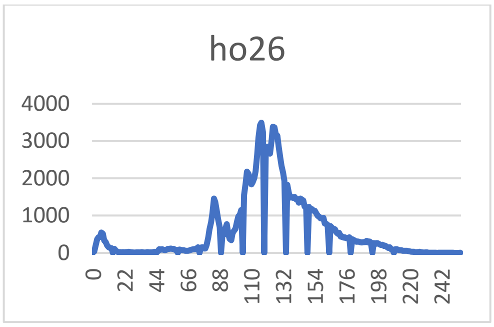
Figure 5: The output of the pipeline: the original image, in which coloured crosses represent the location of the image. The colour of the crosses determines how the object has been identified. Green crosses: pointing hand, yellow crosses: spread hand, red crosses: unidentified item.

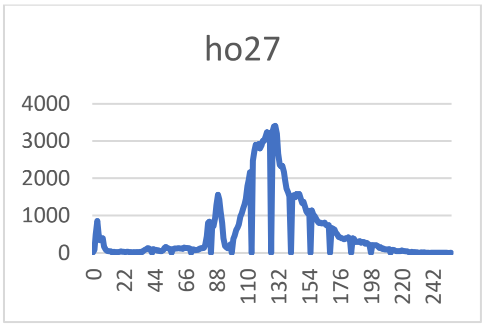
Appendix

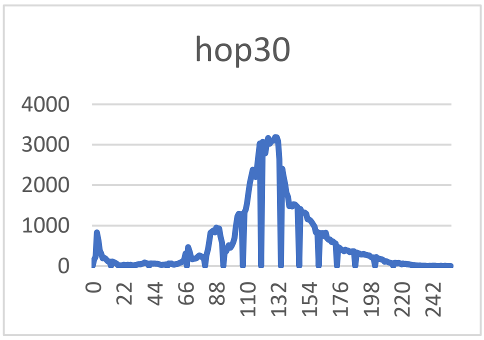
Appendix A: Histograms of 6 different images after auto-contrast correction. While we originally thought that the hand would be described by the small peak between values 70 and 90, this actually described the parts of the table that had no map projected onto them. The small peak between 0 and 15 described the floor at the bottom of the image, and the big peak described most of the map. After further experimenting, the hand was found to be best described by the valley between 0-15 and 70-90.

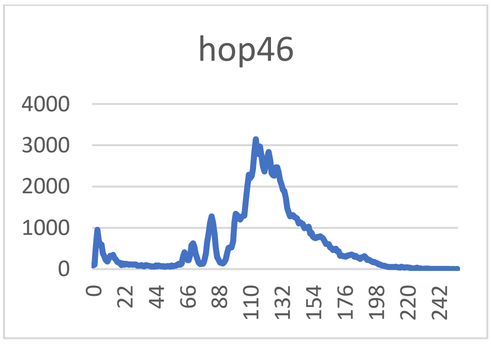












Appendix B: 22 output images, some succeeding in identifying the hand correctly and others failing to do so. This could be caused by several different reasons: the initial threshold returning only part of the hand because of the beamer poorly lighting up the hand, improper placement of convex points and convexity defects, or the finger counting algorithm being either too generous or too strict in its definition of what a finger is. How these issues could be resolved has already been discussed in the reflection.