Final Project: Hand Tracking
Computer Vision

Mick van Gelderen
m.m.vangelderen@student.tudelft.nl
4091566

July 2015

This document describes my final project for the Computer Vision course. I wanted to create a hand tracker that accurately tracks the tip of your finger so that you could use it as an input device. The pose of the entire hand could signal events such as clicking or dragging. Additionally, writing all the code myself was more important to me than getting the thing to run.

1 Starting off

Many papers I read started off with something similar to this: "Hand tracking is very useful. Hand tracking is very difficult because hands can assume many different poses". Stubborn as I am I thought: "So? I can recognize hands easily, why can't we teach a computer to do the same?". As I started playing around with some images I captured with my webcam and failed to produce anything of use (what a surprise) I quickly realized I would probably have to limit the scope.

Because of a course I recently took and my interest in showing my project to friends and family I started playing around with the HTML5 user media and webgl interfaces. After a lot of searching around and experimentation I managed to get images from my webcam on my GPU and process them using a fragment shader. I wrote a skin detection-, edge detection- and general 5x5 filter shader. These can be found here: `https://mickvangelderen.github.io/cv-final/`.

During further development it was easier to use MATLAB, especially when I wanted to compute local histograms. .

2 Learning from others

After some google scholaring and reference following, I realized that a lot of work has been done in this area. Old systems used gloves and markers. The past ten years there have been a lot of cheap single camera based systems. Nowadays most research is being done using stereo cameras and/or depth cameras. The kinect in particular is a popular device because of its price, quality and availability. Overviews of research and methods in this field are given in [2, 8]. Most systems consists of 2 phases: 1) detecting and tracking the hand and 2) determining the position.

3 Hand detection

Hand detection is a challenge, M. Van den Bergh puts it nicely: "There are several cues that can be used: appearance, shape, color, depth, and context. In problems like face detection, the appearance is a very good indicator. The eyes, nose and mouth always appear in the same configuration, with similar proportions, and similar contrasts" [11]. Hands however are able to assume many formations and are usually viewed from different directions, unlike faces.

3.1 Skin detection

In order to detect a hand in an image, many people use a combination of detectors. The most common one is the skin detector [12, 5, 1, 6, 7, 11, 4, 2]. It is difficult however to capture the many different skin tones people have while being insensitive to illumination without classifying background pixels as hand pixels. This is why in more recent works, an effort is made to personalize the skin color model on the fly [11].

The model used for the skin color is usually a, possibly mixed, multivariate gaussian model. A histogram based method is used in [12] but from my experiments its not substantially better, its just slower to compute. I've also tried using the mixed gaussian model from [4]. See the jones2002statistical files in `https://github.com/mickvangelderen/cv-final/tree/master/matlab/experiments`. The same story held for this model: not significantly better but slower.

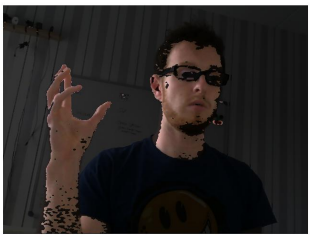

Figure 1: Skin detection on the GPU. Non skin regions are darkened.

Figure 2: Final skin detection

Skin detection amounts to classifying patches of an image as being skin or not. There are infinitely many approaches to this but I will discuss two: fitting a multivariate gaussian and histogram comparison. I used the following dataset for some experiments: `http://www.robots.ox.ac.uk/~vgg/data/hands/`. The code for these experiment is in `https://github.com/mickvangelderen/cv-final/tree/master/matlab/hands_dataset`.

Figure 3 shows the results of using this dataset. I used the annotations in the dataset to extract image

patches containing the hands. I then calculated the histograms for all these patches and the resulting mean histogram. I hoped that background noise would be removed. I compared histograms of patches in the input images to the mean histogram. Unfortunately there was a lot of black and white in the backgrounds, resulting in a useless mean histogram. I needed some way to extract a better histogram or a dataset of a higher quality. I decided to go for the second approach.

Figure 3: Skin detection based on the dataset using a histogram. The white dots on the left are the detected skin regions. Yes, it is garbage.

I took some images from the hand dataset and some images I captured with my webcam to create a dataset of 21 pictures. I manually created a mask of the hand pixels using Photoshop.

3.1.1 Gaussian

I trained a gaussian model in the RGB color space and one in the IQ components of the YIQ color space. I determined thresholds for the output of the probability density functions of both models by minimizing the error through a brute force approach. This was fast and detailed enough. The resulting error curves are shown in Figure 4.

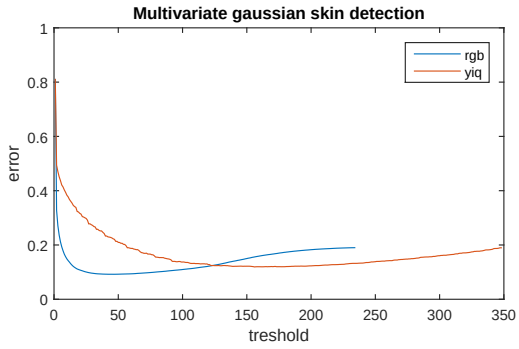

Figure 4: Threshold selection for the gaussian skin models.

The RGB model has a lower minimum error for my test set (which also happens to be the training set ahum). Perhaps this is not the case for a test set with more diverse lighting conditions. I tried to pick images with very different lighting and skin tones from the hands database however.

The results of applying the RGB and YIQ models with their minimum error thresholds are shown in Figure 5.

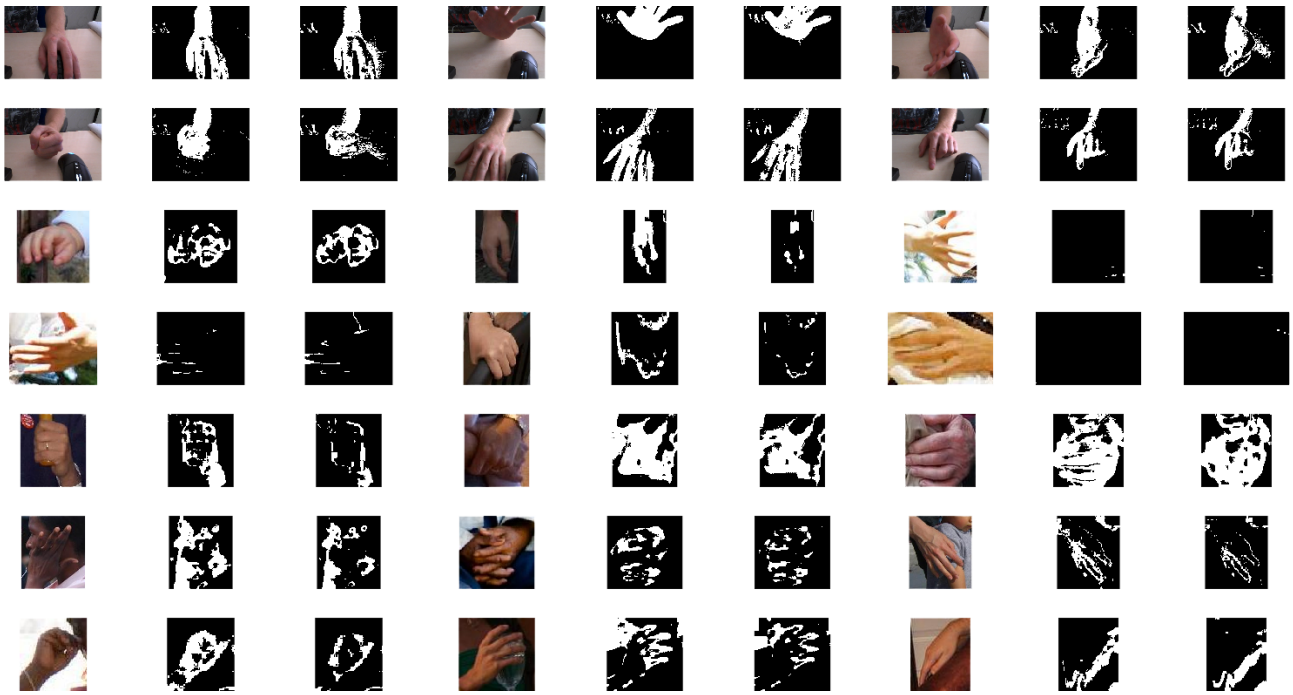

Figure 5: Evaluation of gaussian RGB and YIQ models. Every three subsequent images (left to right, top to bottom) show in order, the input image, the RGB model classification and the YIQ model classification. White is skin, black is non-skin.

The relatively underrepresented yellow-ish and overlit hand images are completely misclassified. To counter this we could lower the thresholds (but increasing the error) or try using a mixed gaussian model. The first option is probably better because you can use the computation time you save for adapting the model to the user.

3.1.2 Histogram

Another approach is to calculate a histogram of the training skin pixels and comparing those with the local histograms in the input image. We hope that hands have a certain color palette that can be represented with a frequency histogram. of colours There are multiple parameters to tune:

1. The histogram training method
2. The color space/preprocessing
3. The histogram detail (number of bins, edge locations)
4. The method of histogram comparison
5. The histogram window size
6. The classification method

I trained the histograms by computing them per image for all hand labeled pixels (by the masks I drew in Photoshop) in that image. I could also move over each image using a certain window size and obtain the histograms that way but the first method is more stable. Perhaps it is worse however at capturing the local color palettes however. One thing is sure, it is hella slow to compute local histograms in MATLAB.

The color space I used was YIQ to limit variance due to illumination. I used 32 bins and took care that the I and Q components are between ± 0.5957 and ± 0.5226 respectively. The histograms are compared using the histogram intersection described in [10]. It basically sums the overlap between bins. The window sizes I tested were 4x4 and 8x8. The results are shown in Figures 6 and 7. The classification methods I used were the maximum of the similarities between the input histogram and all trained histograms, and the similarity between the input histogram and the mean histogram.

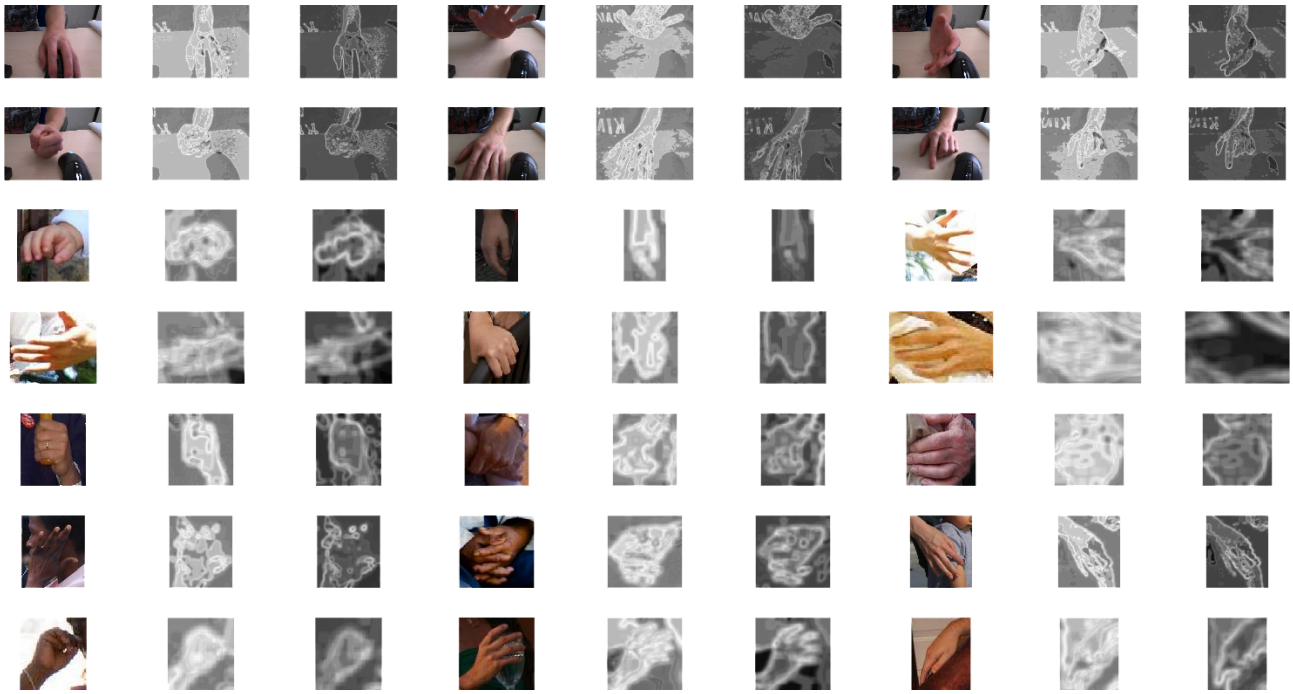

Figure 6: Evaluation of histogram models with a 4x4 input image histogram window. Every three subsequent images (left to right, top to bottom) show in order: the input image, the all histogram similarity and the mean histogram similarity. Bright areas denote skin.

The all histogram similarity yields better results. This may be due to using the training images as test images. The method was too slow in MATLAB however. I did look for ways to implement the histogram computation on the GPU but this would take a significant amount of time to learn and implement [9].

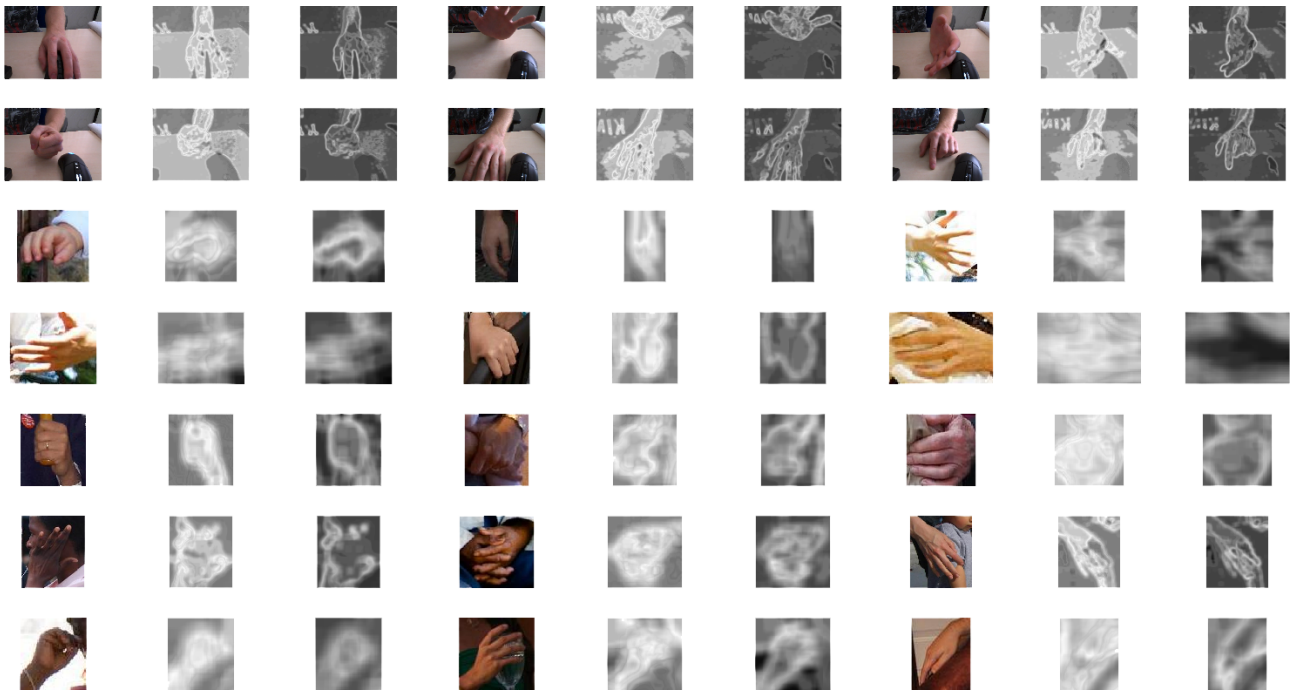

Figure 7: Evaluation of histogram models with a 8x8 input image histogram window. Every three subsequent images (left to right, top to bottom) show in order: the input image, the all histogram similarity and the mean histogram similarity. Bright areas denote skin.

4 Final application

The final application still requires some work. I combined skin detection with edge detection to find potential hand regions. Now I need to extract the local maxima and perform some form of gesture classification to provide interation and get rid of the false positives such as my face. Also I would have liked to and adapt the skin model on hit and use a kalman filter to track the hand. Figure 8 shows the current state of the project.

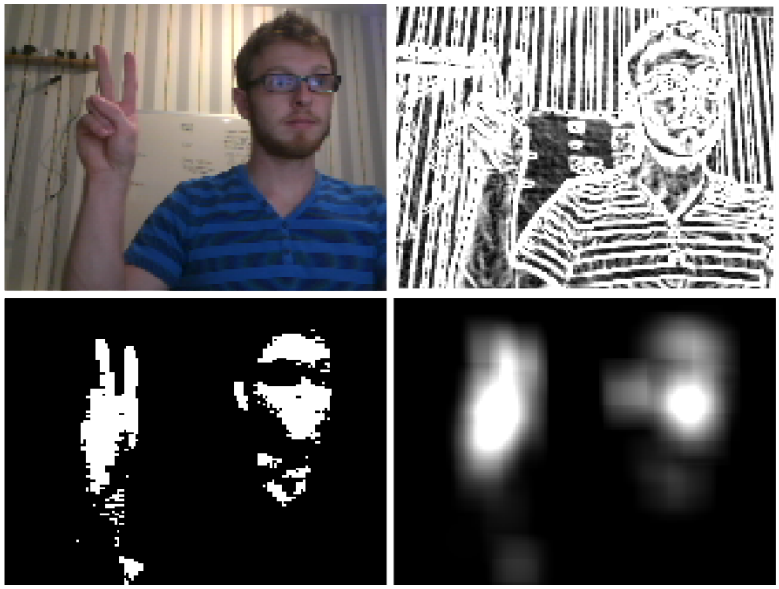

Figure 8: Final application. Showing left to right, top to bottom the input, edge detector, skin detector and combined and blurred hand detection likelyhood image.

5 Other considerations

I also experimented with K-means segmentation a little bit in order to eventually detect hands outlines. Figure 9 shows a frame from my webcam feed clustered at an interactive framerate. The resolution however was too low. Image segmentation brings a whole load of problems with it that need to be dealt with before we can consider different segments as potential complete hands.

Figure 9: K-means on webcam feed.

I played with motion detection and background substraction. Motion detection is good for determining interest regions during detection. Background detection is very nice for segmentation but it is akward to set up in a working application. Asking to fix the camera and provide a background image should not be necessary.

Also I wanted to look into removing shadows [3] to decrease illumination sensitivity.

6 Conclusion

In the end I did not manage to create an interactive drawing application. I could have decided to limit the scope of my project by using markers or detecting a single hand position but I was more interested in experimenting with different skin detection models, reading about applications of hidden markov models, radial basis functions, recurrent neural networks and model based posture estimation, histogram comparison methods such as Kullback Leibler and Swain intersection, HTML5 user media, webgl, clustering and segmentation methods, edge detection methods such as sobel, DOG, scharr and canny. I easily get lost in learning about all those interesting topics (which I do not mind because I like learning).

Thanks for the support and teaching. Computer Vision is a course that enables us to talk proudly of what we study.

References

[1] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. "Hand gesture recognition using a real-time tracking method and hidden Markov models". In: *Image and vision computing* 21.8 (2003), pp. 745–758.

[2] Ali Erol et al. "Vision-based hand pose estimation: A review". In: *Computer Vision and Image Understanding* 108.1 (2007), pp. 52–73.

[3] Clement Fredembach and Graham Finlayson. "Simple Shadow Removal". In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. Vol. 1. IEEE. 2006, pp. 832–835.

[4] Michael J Jones and James M Rehg. "Statistical color models with application to skin detection". In: *International Journal of Computer Vision* 46.1 (2002), pp. 81–96.

[5] Chan Wah Ng and Surendra Ranganath. "Real-time gesture recognition system and application". In: *Image and Vision computing* 20.13 (2002), pp. 993–1007.

[6] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. "Markerless and efficient 26-dof hand pose recovery". In: *Computer Vision–ACCV 2010*. Springer, 2011, pp. 744–757.

[7] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. "Tracking the articulated motion of two strongly interacting hands". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 1862–1869.

[8] Siddharth S Rautaray and Anupam Agrawal. "Vision based hand gesture recognition for human computer interaction: a survey". In: *Artificial Intelligence Review* 43.1 (2015), pp. 1–54.

[9] Thorsten Scheuermann and Justin Hensley. "Efficient histogram generation using scattering on GPUs". In: *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. ACM. 2007, pp. 33–37.

[10] Michael J Swain and Dana H Ballard. "Color indexing". In: *International journal of computer vision* 7.1 (1991), pp. 11–32.

[11] Michael Van den Bergh and Luc Van Gool. "Combining RGB and ToF cameras for real-time 3D hand gesture interaction". In: *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*. IEEE. 2011, pp. 66–72.

[12] Ho-Sub Yoon et al. "Hand gesture recognition using combined features of location, angle and velocity". In: *Pattern recognition* 34.7 (2001), pp. 1491–1501.