Body Photo Browser

Bastiaan Weijers 3256669
Colin Smits 4075390

March 20, 2015

1 Introduction

Currently, virtual reality is starting to grow in the digital world, which makes it possible for a user to move in a virtual world using motion, which is then detected by sensors. This paper will discuss an implementation of processing hand gestures in order to operate a photo browser system (Windows Photo Gallery[1]). First, we will discuss the techniques used in order to create a working system, and after that the requirements needed for the system to work will be explained. In addition to this paper, please refer to the film [2].

2 Processing Steps

For building our system, we used Python as programming language, due to it's ease of use with matrix calculations. The testing was done using a 1.5MP integrated web cam. Our system is based on detecting the hands using color filters, and getting the center of the hands which can then be tracked. The following gestures can be recognised:

Swipe Left: Using one hand (in view), going from right to left in the view. This will give the previous photo.

Swipe Right: Using one hand (in view), going from left to right in the view. This will give the next photo.

Rotate left: Using two hands (in view), going up with the right hand, and simultaneously going down with the left hand. This mimics the idea of rotating a picture counter clockwise.

Rotate Right: Using two hands (in view), going up with the left hand, and simultaneously going down with the right hand. This quite mimics the idea of rotating a picture clockwise.

[1] http://windows.microsoft.com/nl-nl/windows-live/photo-gallery
[2] `http://youtu.be/ye8yR4tcteo`

Firstly, we process the frame by converting it from BGR (channels switched)into a YCrCb color coded file. Then, we apply the cv.inRange() function in order to filter the pixels that are in the color range. Then, we use 2 iterations of erosion and dilation, and a Gaussian blur, after which we compare the original image to the *mask*. This generates a better qualitative comparison, as well as bigger area, so that contours can be detected more easily (explained further on). The pixels which are contained in both files will be included in the further processed file.

Secondly, we try to detect hands in the picture. For that we use contours by using the OpenCV operation findContours(), after which we filter contours with an area too small, as these may bbe noise inducted contours. We then find the center of each large enough contour by applying cv.moments(). Using these centers, we can track the hands and recognise gestures by comparing the center's location per frame.

3 Assumptions & Requirements

In order for our system to work we have to set the following requirements:

- Due to the fact we detect skin color, the only parts included in the view must be hands. If a face is in view, it might be detected as a contour, and thus be classified as hand.
- As we want to detect hands, the user must wear long sleeves. If there is a "naked" arm in view, the moments function will find a COG somewhere in the middle of the arm. This COG will have less movement, so nothing is detected.
- The background must be of a color which is not similar to the user's skin color, i.e. a wooden chest will not suffice as a valid background. A white wall, however, will.
- Due to color boundary issues, the user must be a person having a light skin color.

4 Result

In the beginning, we struggled with implementing our idea, because it was difficult to detect skin color the way we wanted to (in HSV color space). However, when we started using YCbCr as color space, it worked wuite well, and we could continue. At last, we think that our product is overall robust (low amount of false positives) in a controlled environment (see Section 3).