

Computer Music Controller Based on Hand Gestures Recognition Through Web-cam

Junhao Jiang, Junji Ma, Yiye Jin

Department of Electrical Engineering, Stanford University

junhao.john.jiang@gmail.com majunji@gmail.com yiyejin1126@gmail.com

Abstract—the paper addresses a solution for a host of gesture-based controls for music player on laptop via webcam. The core functionality is a detection mode developed to capture hands, detect hands, recognize gesture, and trigger other event of interest, such as windows media player. The algorithm is developed to provide a real-time hand detection to control music player, which requires efficiency and rapid response.

Keywords—hand detection; gesture recognize; matlab image processing tool

I. INTRODUCTION

Hand detection and recognize is an important and hot research issue in field of human-computer interaction, because host of gestures have great potential to be used to interact and control computers efficiently in the near future [1, 2]. With the highly adoption of built-in camera in laptop, mobile phone and tablet, the huge number of applications of hand detection have been designed and demonstrated. However, most of them lack practical value to majority of users. In this paper is driven by a simple idea: a keyboard & mouse-free music controller. Take advantage of webcam's real time frame tracking function, a music player controller is implemented by Matlab code, combining skin detection, area labeling, erosion, dilation, and motion differentiation. In order to achieve the best performance and precision, three hand detection algorithms are designed and examined case by case.

In order to achieve the best performance and accurate detection, three different algorithms are designed and implemented. In the following three parts, these three algorithms are briefly introduced. More details are discussed in second section.

A. Skin and shape detection

In order to extract the hand from the background, a skin pixel detector has been implemented to filter the frame captured by the webcam. Since the part of face and the similar-color background items may be captured simultaneously, an erosion and a dilation are applied to suppress the noise, and an area labeling is used to find the largest detected area (assume that the hand should be the largest area of skin color item captured by webcam). And compute the contour extraction of obtained image, a hand-shape is detected [3]. The accurate of this algorithm is partially depending on the light condition and the skin color.

B. Skin and feature detection

Similar as the first algorithm, the skin pixel detector is first applied to the captured image. And then a noise reduction is performed by erosion and dilation. After the above two processing, a feature extraction filter is used to find fingertips for thumb, represented by peaks along the contour [4]. The five fingertip points detected accounts for a hand. The accurate of this algorithm requires stretching hand heavily and avoiding face detection noise.

C. Motion and contour feature detection

This hacking algorithm is developed since the above the two algorithms are conditionally performance-well. Follow the two standard processing, skin detector, erosion and dilation noise reduction, two successive processed images are applied to compute differentiation to obtain contour. The assumption is: the hand is placed closest to the webcam, so its movement is the easily detected than other items according to the 3D depth of field. Based on this assumption, differentiate the two successive captured images is to generate the most-movable item's contour. Use the second algorithm's method to detect the hand feature on the obtained contour [5, 6]. Compared to the above two algorithms, this one is much robust and accurate.

II. DETECTION ALGORITHM

This section describes the implementation details of the three algorithms mentioned above. Since the skin detection and the erosion and dilation processing are standard processing steps in all three algorithms [7], the first two steps are demonstrated first. Each algorithm's unique processing steps will be described individually.

A. Skin detection

The most straightforward “pre-process” should be applied involves separating potential hand pixels from non-skin pixels. Human skin color normally lies in a range in a designate color space [8]. In this process, the face and some light color background items may be detected as hand pixels, needing other process to filter out.

Here we developed two types of detectors with respect to two different color space representation YCbCr/HSV.

In YCbCr representation, first remove the Y value since it's luminance-dependent. Then by analyzing training images we map the Cb component and Cr component into a mask. With this mask, we then can segment the skin pixels when it falls in the range where we labeled as skin.

In HSV representation of color, we remove the V component and as what we did in YCbCr color space projection, we project "H" hue component and "S" saturation component into a plain represent by two axes. After training with 10 to 20 snapshots we set up a set of static thresholds.

Both skin segmentation algorithms will make the image into a binary mask where white pixels represent skin areas and black pixels represent non-skin areas.

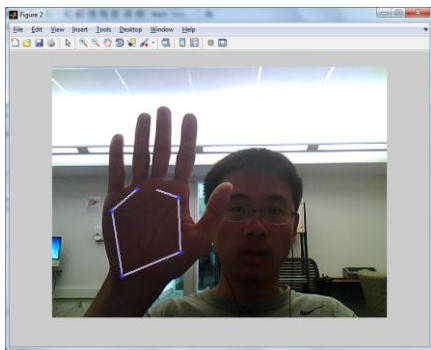


Figure 1, Label skin region in training pictures.

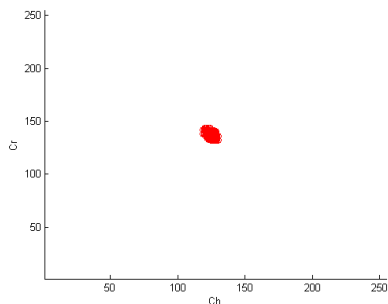


Figure 2, CbCr color space projection.

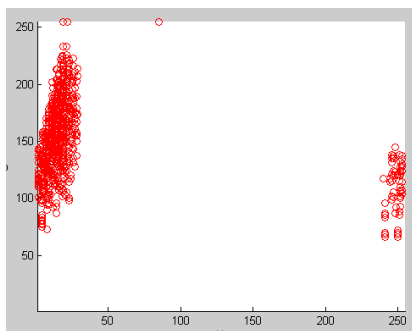


Figure 3, HSV color space projection

From the distribution of the colors projected in the two spaces, we can see that in Cb-Cr axis, the skin colors cluster is better than those in H-S axis. So we choose Cb-Cr color projection in our following algorithms to get skin area.

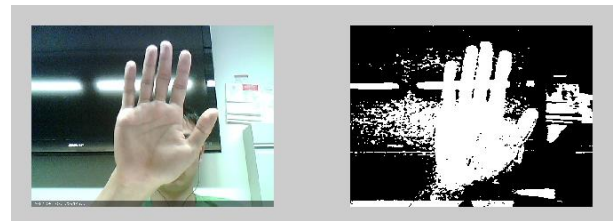


Figure 4, skin segmentation using Cb-Cr color space (without face disturbance).



Figure 5, skin segmentation using Cb-Cr color space (with face disturbance)

B. Noise reduction by erosion and dilation

After the skin detection, all potential hand skin areas are detected and depicted as white pixels. Of course there are small noisy regions will be present, but we assume that the region corresponding to the hands is the largest. Thus we first suppress the noise by "disk" erosion. Those small noisy regions, mistakenly detected as hand skin, are normally skin-color like items under certain light condition [1]. The erosion process can effectively erase those small noisy regions.

But the real regions corresponding to the hand may also shrink according to the erosion process designed for noise. Thus we use dilation to strengthen the largest detected region (i.e. hand), help to enhance the desired hand detection. After this process, the hand shape contour can be outlined on the image as white pixels. The rest of the image processing steps are to determine what we obtain so far is a "hand", which requires feature recognizing model.



Figure 6, Filtered hand detection (without face disturbance)



Figure 6, Filtered hand detection (with face disturbance)

We can see that sometimes due to a large area of skin recognized at the face, the result could be very different. So here we assume that user will always put their hands in front of their face since it's the most common posture to look at the webcam. So with this assumption, our method is reliable.

C. Convex contour and area property recognize (skin and shape detection algorithm)

In our first algorithm, we use convex contour to envelope the detected hand-shape region. Use this convex contouring region minus the detected region to obtain approximately six separate individual regions. Four of the regions are the space between to neighbor fingers and two are inter-space between hand outline and convex contour [4]. We apply the centrifugal calculation to remove those two eccentric inter-spaces, leaving four fingers' inter-spaces. The threshold count is set to three or four to determine a hand, which means if there are three or four separate regions left, we assume the detected shape is a hand.

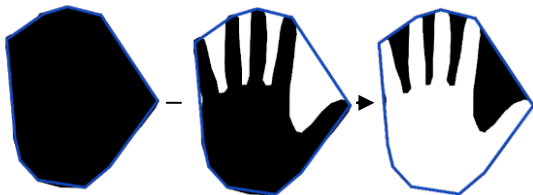


Figure 7, Convex hull subtract the hand region to get regions between fingers

D. Convex contour and feature recognize (skin and feature detection algorithm)

In our second algorithm, we still use convex contour to envelope the detected hand-shape region. Because the convex contour has five intersections with detected hand-shape region, which are the five fingertips, we can use this as a standard. The threshold count is set to four or five to determine a hand, which means if there are four or five intersection points, we assume the detected shape is a hand [6]. To get the accurate number of finger tips detected, we calculate the overall centroid of all the points detected and only counts ones above the average centroid.

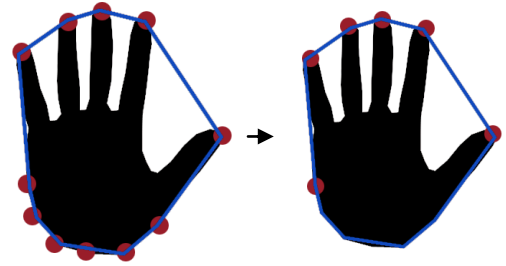


Figure 8, Intersection points with convex hull (left)
Filtered by position of points(right)

E. Differentiate motion and feature recognize (motion and contour feature detection algorithm)

In our third algorithm, we try to find the difference between the successive frames captured by the webcam. This assumption is based on that: since the hand is placed closest to the webcam, the tiny movement of the hand is most easily captured. By the difference between two successive, the hand-shape contour is detected. And we apply the edge detection and vertical direction detection to filter out the vertical certain-length edge. For a perfect hand-shape, 10 vertical straight edges will be shown [3]. The threshold count is set to six to ten to determine a hand, which means if there are six or up to ten vertical straight edges detected, we assume that a hand is detected.



Figure 9, Frames difference due to hands' jittering

III. EXPERIMENTS AND RESULTS

In this section, we present our experiments about the above three algorithms. The whole application is implemented in Matlab, including three main parts: trigger the webcam and capture 30 frames a second, hand detection algorithm processing, and controlling mechanism on windows media player. There are three indicates for system performance, detection accurate rate, robust of the system, and execution lagging time.

A. Algorithm one: skin and shape detection

The Fig 10 and Fig 11 show the first hand algorithm to detect the shape of hand.



Figure 10, No hand detected

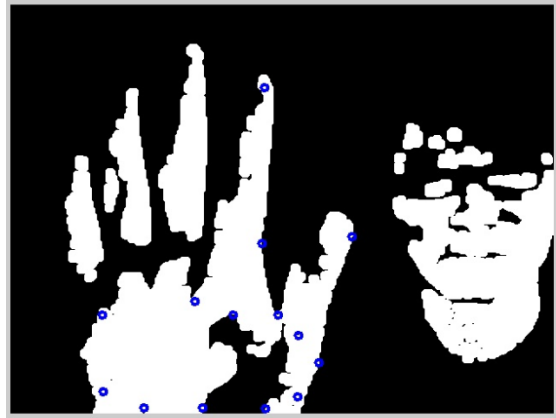


Figure 12, Hand not detected



Figure 11, Hand detected

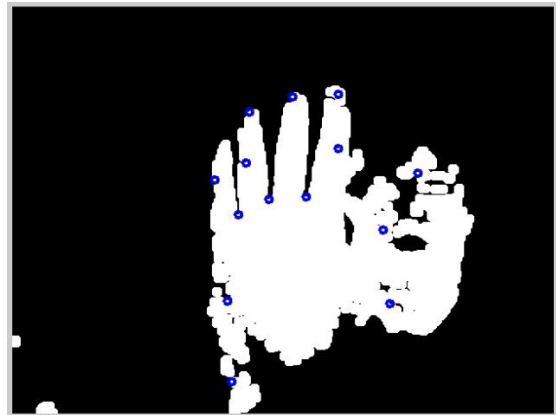


Figure 13, Hand detected

B. Algorithm two: skin and feature detection

The Fig 12 and Fig 13 show the second hand algorithm to detect the feature points, such as fingertips and corners.

C. Algorithm three: motion and contour feature detection

The Fig 14 and Fig 15 show our application' controlling mechanism. Start the application with setting up the webcam and the media player. When the hand is placed in front of webcam for 2 seconds and removed, the media player is triggered to play the music, shown in Fig 14. The music will be paused when the hand is placed up for 2 seconds, shown in Fig 15. Fig 16, Fig 17 shows when hand is not detected or not and depending on which, whether the music play/pause function is triggered.

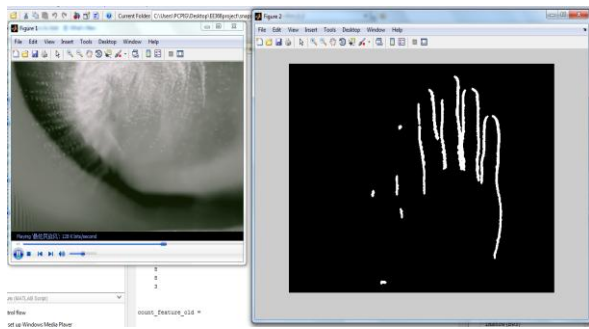


Figure 14, Music playing triggered by putting hand in front of webcam

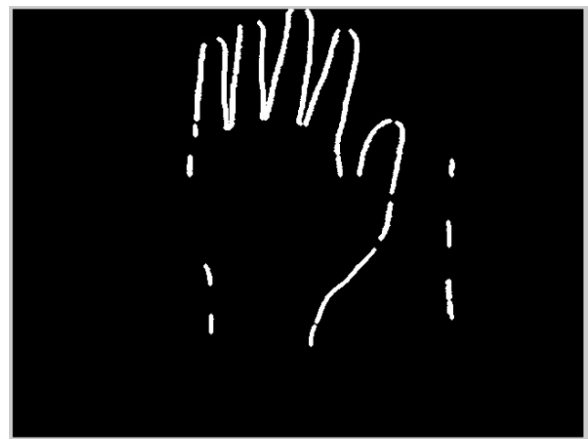


Figure 17, Hand detected

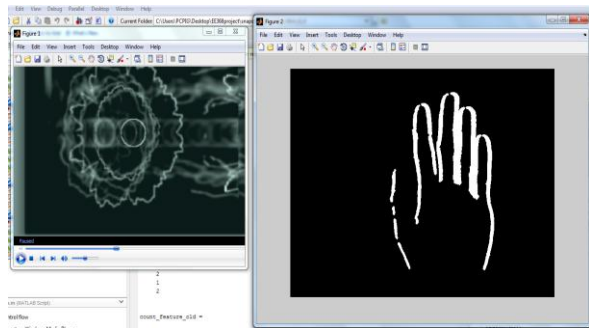


Figure 15, Music pausing triggered by putting hand in front of webcam



Figure 16, Hand not detected

We also tested the three algorithms in different situations for robustness. The situations contain two different lighting conditions which are dimming light and natural strong lighting condition and similar ship non-hand objects. The results are listed as in Table 1.




Algorithm	 #1	 #2	 #3
Dimming Light (Real Hands)	6/10	5/10	10/10
Natural Strong Light (Real Hands)	2/10	2/10	10/10
Dimming Light (Objects)	0/10	0/10	2/10
Natural Strong Light (Objects)	0/10	0/10	1/10
False Positives	0	0	3
False Negatives	12	13	0
Correctness	70%	62.5%	92.5%

Table 1, Results table for different test situations.

IV. CONCLUSION

This paper presented a real-time hand gesture detection controlling music player. We designed three different hand detection algorithms to make sure the best performance. The third one, motion and contour feature detection algorithm, is the best performed compared to the rest two. The hand detection accurate rate is up to 80% according to our 108 sample tests, including three different hand skin color volunteers with four different light condition and three orientations of hand placement. The hand skin color and the light condition are combined to degrade the detection results, which could be improved by machine learning or machine training. The hand placement parameter requires hand orientation detection after shape detection. Three determination standards are implemented to distinguish these three hand orientations. We also designed a fast-responsive “state machine” to avoid long time lagging when user puts his hand in front of the webcam. Currently, the average lagging time is 2.4s. There is a tradeoff between the lagging time and the robust of the system, because the robust feature demands comprehensive, sometimes redundant checking mechanism, resulting a lagging time inevitably. Please refer to the source code for a whole fully functional music player programmed in Matlab with three method implementations available to test.

ACKNOWLEDGMENT

This paper would not have been possible without a lot of help and support. First, we would like to thank the Professor Bernd Girod for this resourceful class, which help to build our fully knowledge about image processing from simple point operation to complicated feature detection. We cannot imagine we could build this “magic” application at the very first beginning of this class. EE368 is a great class. Second, we would like to TA David Chen and Derek Pang, and our project

advisor Mina Makar to help us through this class and have patience to give us any guidance we need.

WORK CONTRIBUTION

Junji Ma: hand detection algorithm design, optimization and implementation.

Junhao Jiang: hand detection algorithm design and state machine implementation.

Yiye Jin: hand detection algorithm design, and webcam frame capturing and music player controller implementation.

REFERENCES

- [1] Erdem Yoruk, Ender Konukoglu, Bulent Sankur, and Jerome Darbon, “Shape-Based Hand Recognition”, IEEE Transactions of Image Processing, VOL. 15, NO. 7, July 2006
- [2] Mathias Kolsch and Matthew Turk, “Robust Hand Detection”, Department of Computer Science, University of California Santa Barbara, CA.
- [3] R. Lockton and A. Fitzgibbon, “Real-time gesture recognition using deterministic boosting”, In Proc. of British Machine Vision Conference, 2002
- [4] Mathias Kolsch and Matthew Turk, “Analysis of Rotational Robustness of Hand Detection with a Viola-Jones Detector”, Department of Computer Science, University of California Santa Barbara, CA.
- [5] X. Zhu, J. Yang, and A. Waibel, “Segmenting Hands of Arbitrary Color”, In Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition, 2000
- [6] P. Viola and M. Jones, “Robust Real-time Object Detection”, Int. Journal of Computer Vision, 2002.
- [7] M. J. Jones and J. M. Rehg, “Statistical Color Models with Application to Skin Detection”, Int. Journal of Computer Vision, 46(1):81-96, Jan 2002.
- [8] Eng-Jon Ong and Richard Bowden, “A Boosted Classifier Tree for Hand Shape Detection”, Center for Vision, Speech and Signal Processing, University of Surrey, Guildford, GU2 7XH.