Hand Gesture Recognition in Mobile Devices: Enhancing The Musical Experience

Oriol Nieto[1]* and Dennis Shasha[2]

[1] Music And Audio Research Lab
[2] Department of Computer Science
New York University
oriol@nyu.edu shasha@courant.nyu.edu

Abstract. Mobile devices typically include a built-in camera that allows high definition video recording. This, combined with last generation mobile processors, makes possible real-time computer vision that can potentially enhance the interaction with various mobile applications. We propose a system that makes use of the built-in camera to recognize and detect different hand gestures that could be ultimately mapped to any application. As an example, we show how the system can control real-time music synthesis parameters, and to better capture its effectiveness we provide links to video demos. We then show how our method is easily extendible to other applications. We conclude the paper with qualitative usage observations and a discussion of future plans.

Keywords: computer vision, hand recognition, mobile interface, mobile music

1 Introduction

The advent of multitouch mobile computing devices has led to a significant increase of the use of technology in our day to day lives. Many of these devices include multiple cores that make the computation of real-time high definition graphics and low latency, high quality audio feasible. Moreover, these devices typically include a built-in high resolution camera to provide photo and video recording to their users. One of the interesting questions that these portable devices raise is how we can improve the interaction with them.

We propose a framework to detect hand gestures with a built-in camera in real time. We then use these recognized gestures to control different audio synthesis parameters enabling a naive user to play the instrument in the air. In contrast to proposed systems [8, 13], our method identifies different hand gestures without the need for special color marked gloves or any other type of additional hardware. Thanks to recent technologies like, e.g. Kinect[3] or Leap Motion[4], high quality

* Thanks to Fundación Caja Madrid for the funding.
[3] http://www.xbox.com/KINECT
[4] https://www.leapmotion.com/

sensors have become widely available, so can support music and multimedia interfaces (e.g. see [16, 12]). However, in this project we rely solely on common portable devices.

As discussed in [2], traditional musical notions should remain in electronic music, otherwise the audience may not properly interpret the different cues in the audio. By having an instrument that is played similarly to a more classical Theremin [6] (i.e. by using different recognizable gestures that produce a qualitative transformation on the audio), we would not only make spectators appreciate the music in a desirable way, but would also provide the performer with a portable instrument that can be easily played virtually anywhere. (see Figure 1 for a Screenshot of our application).

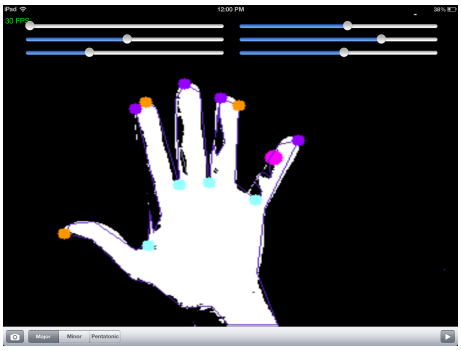

Fig. 1: Screenshot of AirSynth recognizing one open hand with the detected finger tips or convex points (in purple and orange) and finger web or defect points (in cyan).

Various hand detection algorithms have been presented in the past, mostly under the computer vision framework [10, 18], and more recently in portable devices[5]. In this project we propose to first recognize the hands using a color filter that is set up to map the skin color (similar to [17, 4]). Then, after applying different transformations such as blurring or high contrast, the image is treated as a polygon, where a classic convex-hull algorithm [1] is applied. Applying a specific heuristic algorithm, we can classify various gestures of the two hands at a given time.

These gestures and the distance to the camera (i.e. the size of the hands) can be used in real-time to capture different gestures as in [9]. In our music application, this results in a synthesizer that is played solely by moving the hands and fingers in the air (links to various video demos are available in section 3.3).

[5] http://www.nanocritical.com/nanogest/, http://eyesight-tech.com/technology/, or http://www.yonac.com/AirVox/index.html

2 Detecting Hands

In order to make the process achievable in real-time on portable devices, the proposed hand detection algorithm avoids expensive computational techniques such as neural networks [15] or support vector machines [18] (other machine learning techniques are described in [5, 7, 3]). Instead, the detection of the hands can be divided into two subproblems: the acquisition of the data points representing the defect areas (the triangular regions between the bases of pairs of fingers) for the convex-hull of each hand, and the determination of the number of finger webs for each hand given the triangular regions.

2.1 Defect Points Acquisition

The goal of this part of the algorithm is to obtain a set of points representing the defect areas from an RGB image represented by a 2D matrix X. In general, a defect area is the area between a polygon and its convex hull. A convex hull is the smallest outer envelope that contains a specific set of points (in our case, these points are a polygon that approximates the shape of the hand). The block diagram of this process is depicted in Figure 2, and a visual example of it can be seen in Figure 3.

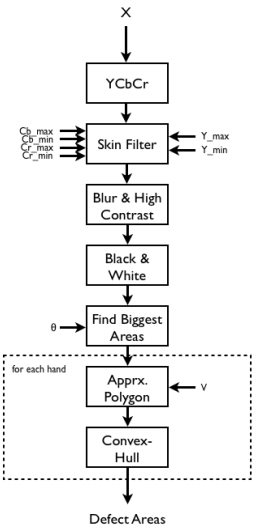

Fig. 2: Block-diagram of the acquisition of the defect points from an RGB image.

The process starts from an RGB image X that is transformed into the YC_bC_r color space. The YC_bC_r model, instead of being an additive color space like RGB, stores a luminance value Y combined with two different color parameters: the blue difference C_b and the red one C_r. This model makes the skin filtering

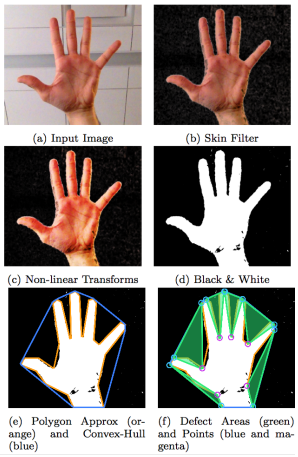

Fig. 3: Example of the acquisition of the defect points.

method more robust and easy to implement than additive/subtractive methods such as RGB or CMYK, since the color is determined by only two values (C_b and C_r). Generally, the only parameter that must be tuned is the luminance Y. More information on the YC_bC_r model for skin detection can be found in [11].

The next block filters out those pixels from the YC_bC_r image that are not within a set of threshold parameters that represent skin color, quantized by the maximum and minimum values of Y, C_b and C_r. These parameters should be tuned every time the light of the environment changes. However, since we are using a YC_bC_r model, the thresholds for Y are the only ones that will dramatically change depending on the luminance of the room.

After that, the filtered image is blurred by an average filter. A non-linearity of high contrast is applied in order to remove small artifacts that might appear after the filtering. Then, the colors of the image are transformed into black and white to be able to apply the convex-hull algorithm in a more robust way.

Fig. 4: Example of the acquisition of the defect points.

In the following block we look for the two biggest areas in the B&W image. These areas represent the actual hands being tracked by the camera. They are required to be greater than a certain area threshold θ to avoid having little artifacts mistakingly identified as hands. In case there are two hands, we distinguish them by checking the center of the detected areas: the left center will belong to the left hand and the right center to the right one. If there is only one hand facing the camera, we assume that it is the right hand, which will be useful in order to map it to different controls as we will see in Section 3 (this could easily be changed for left-handed users).

Once we have recognized the area of the hands, we approximate a polygon for each hand so that we can ultimately apply a convex-hull algorithm for each polygon. The polygon is approximated by a specified number of vertices V, using the classic algorithm described in [14]. Then the convex-hull algorithm, initially introduced in [1], is applied. This algorithm returns the smallest convex polygon that envelopes the polygon representing the hand (see Figure **??**). After that we can finally obtain the defect areas, which are the triangular areas between the convex hull and the actual polygon. This is the output of this part of the algorithm.

2.2 Recognizing Gestures

Our proposed algorithm identifies different gestures by counting the number of finger *webs* (i.e. the space in between two stretched fingers). There are five different finger web combinations for each hand: from 0 to 4 finger webs detected. This yields a total of 25 different possible combinations to map to various application parameters when using both hands. On top of this, we also detect the distance of the hands to the camera (by checking the area size), thus providing another dimension to these 25 recognizable gestures. In this subsection we detail how to heuristically characterize these gestures. A list of possible detected gestures for a single hand can be seen in Figure 4.

At this point of the algorithm we have the defect areas represented by $A_i = \{P_s^i, P_e^i, P_d^i\}, i \in [1, N]$, where P_s^i, P_e^i, and P_d^i are the starting, ending, and depth points of the i-th defect area respectively (note that all defect areas are triangular, so we need only three points to represent them), and N is the number of defect areas. We are now ready to capture different gestures in real time.

For simplicity, we assume that the finger tips are going to be facing the upper part of the image, even though this could be easily inferred from the different points programatically. First, for each defect area A_i, we compute the average Euclidean distance D_i between the starting and ending points to the depth point:

$$D_i = \frac{||P_s^i - P_d^i||_2 + ||P_e^i - P_d^i||_2}{2} \tag{1}$$

If D_i is greater than a threshold value η —heuristically determined by the area of the entire hand j—, then we might have identified the separation of two stretched fingers (i.e. a finger web). This heuristic value is computed as follows:

$$\eta_j = \frac{\sqrt{H_j}}{5} \tag{2}$$

where H_j is the hand area of the j-th hand, and η_j is the threshold for this same hand. Note that there will be a maximum of two hands at a given time, therefore $j \in [0, 2]$. This value proved to work best in different qualitative tests and it is highly robust to the distance of the hand to the screen.

Finally, we check that the starting and ending points (P_s^i and P_e^i respectively) are under their associated depth point P_d^i in the vertical dimension of the image. Since we assume (for the sake of discussion) that the fingers will always be facing the upper part of the image, we can easily determine whether the fingers are being showed or not by checking the vertical dimension of these points. If this is the case, and D_i is greater than η_j, then we will have found a finger web for the hand j. Formally:

$$D_i > \eta_j \wedge y_{P_{d_j}^i} < y_{P_{s_j}^i} \wedge y_{P_{d_j}^i} < y_{P_{e_j}^i} \tag{3}$$

where $y_{P_j^i}$ is the vertical dimension (y) of the i-th point P of the j-th hand. We do this process for each defect area of each hand.

This algorithm has been implemented in a 3rd generation Apple iPad using the OpenCV framework[6], where methods for polygon approximation and convex-hull are available. The entire algorithm runs in real time, processing around 20 frames per second, including the audio synthesis described in the following section.

3 Application To Music

In order to show the potential of this proposed technology, in this section we describe a specific application to manipulate and generate audio in a digital music instrument. The process of creating music is one of the most expressive activities that humans can perform, and by making use of different gestures the interaction with the device becomes more engaging, both for the player and the listener. Moreover, having this new instrument implemented in a portable device makes it easier to use and to perform virtually anywhere. We call this instrument AirSynth since it can be played solely by moving the hands and fingers in the air.

In AirSynth, each hand has a very different role: the right hand is the one responsible to generate audio, and the left hand the one to control the other synthesis parameters and background drones (this of course could be switched for left-handed performers).

[6] http://opencv.org/

3.1 Right Hand

AirSynth maps the distance of the right hand to the pitch of a specific type of wave. Thus, the interaction resembles that of a Theremin[6], where the closer you place your hand to the device, the higher the pitch will be.

When no finger webs have been detected in the right hand (see Figure 4a), the sinusoid is silenced. For other possible gestures (see Figures 4b–e), different types of waves are produced: sinusoidal, saw tooth, square, and impulse train.

3.2 Left Hand

The left hand is responsible for adding different features to the sound being produced by the right hand. It can also trigger different drone samples. The effects we have implemented are reverb and delay, which are activated by facing 2 or 3 finger webs respectively. To do so, we have only to show these gestures once, and these effects are activated until zero finger webs are detected. With two finger webs detected, the pitch of the right hand will be locked to a pentatonic scale instead of a constant *glissando*. Finally, to launch different samples or drones, 4 finger webs will have to be detected during at least 1 second.

3.3 Musical Performance

The interaction with the instrument is a very expressive one, both sonically and aesthetically. Various videos are available online[7]. The fact that both hands have a relevant and distinct role when performing, that neither is *touching* anything, and that the interaction is happening without noticeable delays, may make it attractive to different performers to explore various sounds and other possible mappings using this technology.

In a real environment of a music performance (e.g. rock concert), light is likely to be too dim or to frequently change. In order to overcome the problem of light adjustment, we could add a small light underneath the device to effectively detect the hands.

4 Conclusions

We have presented a robust technology to detect different hand gestures using an efficient algorithm that is capable of running in real-time on portable devices with a built-in camera (e.g. smartphones, tablets). The algorithm applies various color filters and transformations to the input video image to detect the hands and number of separated fingers in each hand. We have presented a music instrument that makes use of this technology (implemented in a 3rd generation Apple iPad) to obtain an expressive and enjoyable experience for both the performer and listener (see the provided URL in section 3.3 for video examples).

[7] http://tinyurl.com/qd4f58p

References

1. A. M. Andrew. Another Efficient Algorithm For Convex Hulls in Two Dimensions. *Information Processing Letters*, 9(5), 1979.
2. C. Bahn and D. Trueman. Physicality and Feedback : A Focus on the Body in the Performance of Electronic Music. In *Proc. of tICMC*, pages 44–51, La Habana, Cuba, 2001.
3. B. Caramiaux and A. Tanaka. Machine Learning of Musical Gestures. In *Proc. of NIME*, Daejeon & Seoul, Korea Republic, 2013.
4. A. Elgammal, C. Muang, and D. Hu. Skin Detection - a Short Tutorial. *Encyclopedia of Biometrics by Springer-Verlag Berlin Heidelberg*, 2009.
5. N. Gillian, R. B. Knapp, and S. O. Modhrain. A Machine Learning Toolbox For Musician Computer Interaction. In *Proc. of NIME*, pages 343–348, Oslo, Norway, 2011.
6. A. Glinsky. *Theremin: Ether Music and Espionage*. University of Illinois Press, 2000.
7. C. Kiefer, N. Collins, and G. Fitzpatrick. Phalanger : Controlling Music Software With Hand Movement Using A Computer Vision and Machine Learning Approach. In *Proc. of NIME*, Pittsburgh, PA, USA, 2009.
8. T. Mitchell and I. Heap. SoundGrasp : A Gestural Interface for the Performance of Live Music. In *Proc. of NIME*, number June, pages 465–468, Oslo, Norway, 2011.
9. T. Mitchell, S. Madgwick, and I. Heap. Musical Interaction with Hand Posture and Orientation : A Toolbox of Gestural Control Mechanisms. In *Proc. of NIME*, Ann Arbor, MI, USA, 2012.
10. A. Mittal, A. Zisserman, and P. Torr. Hand detection using multiple proposals. In *Procedings of the British Machine Vision Conference 2011*, pages 75.1–75.11, Dundee, Scotland,UK, 2011. British Machine Vision Association.
11. P. Patil and M. Patil. Robust Skin Colour Detection And Tracking Algorithm. *International Journal of Engineering Research and Technology*, 1(8):1–6, 2012.
12. F. Pedersoli, N. Adami, S. Benini, and R. Leonardi. XKin - eXtendable Hand Pose and Gesture Recognition Library for Kinect. In *Proc. of the ACM International Conference on Multimedia*, pages 1465–1468, Nara, Japan, 2012.
13. B. Pritchard and S. Fels. GRASSP : Gesturally-Realized Audio , Speech and Song Performance. In *Proc. of NIME*, pages 272–276, Paris, France, 2006.
14. U. Ramer. An Iterative Procedure For The Polygonal Approximation of Plane Curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.
15. E. Stergiopoulou and N. Papamarkos. Hand Gesture Recognition Using a Neural Network Shape Fitting Technique. *Engineering Applications of Artificial Intelligence*, 22(8):1141–1158, Dec. 2009.
16. S. Trail, T. F. Tavares, G. Odowichuk, P. Driessen, W. A. Schloss, and G. Tzanetakis. Non-Invasive Sensing And Gesture Control For Pitched Percussion Hyper-Instruments Using The Kinect. In *Proc. of NIME*, Ann Arbor, MI, USA, 2012.
17. Y. Wu, Q. Liu, and T. S. Huang. An Adaptive Self-Organizing Color Segmentation Algorithm with Application to Robust Real-time Human Hand Localization. In *Proc. of the Asian Conference on Computer Vision*, pages 1106–1111, Taipei, Taiwan, 2000.
18. B. Yao and L. Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9–16, San Francisco, CA, June 2010.