Project Report

Simulation of Theremin

Weirong Shao(ws1352)

Abstract

Theremin is an electronic musical instrument controlled without physical contact by the performer. The instrument's controlling section is using hands to control the frequency and amplitude of the sound. The project is designed to simulate theremin, using computer camera to capture the gesture of hand and calculate the area of hand, judge the location of hand to control the frequency and amplitude of the sound.

1. Principle

The main principle of my project can be divided into two parts. One is Gaussian Mixture-based Background/Foreground Segmentation Algorithm to subtract background. The other is using moment to calculate the barycentric coordinates and the area of the image.

1.1 Gaussian Mixture Model for Background Subtraction

In the Gaussian Mixture background model, it is considered that the color information between pixels is irrelevant, and the processing of each pixel is independent of each other. For each pixel in the video image, the change of its value in the sequence image can be regarded as a random process of continuously generating pixel values, that is, the Gaussian distribution is used to describe the color rendering rule of each pixel [single mode, multimodal].

For a multimodal Gaussian distribution model, each pixel of the image is modeled by a superposition of multiple Gaussian distributions of different weights, each Gaussian distribution corresponding to a state in which the color represented by the pixel is likely to be generated, and the weight of each Gaussian distribution. And distribution parameters are updated over time. When processing a color image, it is assumed that the image pixel points R, G, B are separated from each other and have the same variance. For the observation data set {x1, x2, ..., xN} of the random variable X, xt = (rt, gt, bt) is the sample of the pixel at time t, then the mixed Gaussian

distribution probability density function of the single sample point xt is:

$$p(x_t) = \sum_{i=1}^{k} w_{i,t} \times \eta(x_t, \mu_{i,t}, \tau_{i,t}) \tag{1}$$

$$\eta(x_t, \mu_{i,t}, \tau_{i,t}) = \frac{1}{\left|\tau_{i,t}\right|^{1/2}} e^{-\frac{1}{2}(x_t - \mu_{i,t})^T \tau_{i,t}^{-1}(x_t - \mu_{i,t})} \tag{2}$$

$$\tau_{i,t} = \sigma_{i,t}^2 I \tag{3}$$

Where k is the total number of distrib u tion modes, $\eta(x_t, \mu_{i,t}, \tau_{i,t})$ is the i-th Gaussian distribution at tim e t, $\mu_{i,t}$ is its mean, $\tau_{i,t}$ is its covariance matrix, $\sigma_{i,t}$ is the variance, I is a three-dimensional unit matrix, and $w_{i,t}$ is the weight of the i-th Gaussian distribution at time t.

1.2 Calculate the barycentric coordinates and the area of the image

Moment can be used to calculate the area and barycentric coordinates and the area of the image.

Defined as the definite integral of f(x)*P(x) with respect to x. In the binarized graph, the zero moment is defined as follows:

$$M_{00} = \sum_{I} \sum_{J} V(i, j) \tag{4}$$

V(i,j) is the gray value of the (i,j) p int. The original meaning of this definition is the sum of the o gray values of all pixels, but because in the binarized graph, white is 1 and black is 0, so the result of M_{00} is the sum of the pixel values of all white areas, and can also be used as the area of the white area.

The first moment is defined as follows:

$$M_{10} = \sum_{I} \sum_{J} i \cdot V(i, j) \tag{5}$$

$$M_{00} = \sum_{I} \sum_{J} j \cdot V(i, j) \tag{6}$$

i, j are the x, y coordinates of each p xel respectively. This definition is the product of the x and i y coordinates of all the pixels, respectively, multiplied by the pixel value, and then summed. Equally, the result of M10 is the sum of the x coordinates of all white area pixels. M01 is the sum

of the y coordinates of all white regions.

Using the first moment, we can find the coordinates of the center of gravity of the graph. The formula is:

$$x_c = \frac{M_{10}}{M_{00}}, y_c = \frac{M_{01}}{M_{00}} \tag{7}$$

As shown above, moment can be used to c lculate the barycentric coordinates and the area of a the image.

2. Processing

2.1 Camera data acquisition

In order to capture video, I set up an object named VideoCapture. Its parameter is the index number of the device. Since this project uses laptop and laptop has build-in cameras. Therefore, the index number is 0.

Camera.read() will return a boolean value, if the frame is read correctly, it will return True. I check the return value to judge whether the frame can be read successfully.

2.2 Processing of video data

Three of the easier-to-use methods are already included in OpenCV: BackgroundSubstractorMOG, BackgroundSubstractorMOG2, BackgroundSubstractorGMG. Here I am using BackgroundSubtractorMOG2.

BackgroundSubtractorMOG2 is a foreground/background segmentation algorithm based on a mixed Gaussian model. It models the background pixels using a K(K=3) Gaussian distribution. The length of time that these colors used in the video is regarded as weight. The color of background usually lasts the longest and is more static.

Here I use the OpenCV's built-ifunction:cv2.createBackgroundSubtractorMOG(). This function has some optional parameters, such as the length of time to model the scene, the number of Gaussian blend components, the threshold, and so on. Set them all to their default values. Then in the entire video we need to use backgroundsubtractor.apply() to get the foreground mask(Figure 1).

Figure 1

By Gaussian blurring, we create smooth transition from one color to another and reduce the edge content(Figure 2).

Figure 2

I use thresholding to create binary images from grayscale images(Figure 3).

Figure 3

2.3 Gesture detection and recognition

Use the convexDefects pit detection function provided by opencv to detect the concave points of the image and then use the cosine theorem to calculate the angle between the two fingers. According to the coordinates of the (starting point, ending point, and far point) in the concave point. It must be an acute angle and the gesture is judged according to the number of acute angles(Figure 4).

Any depression on the object is a convex defect. There is a function cv.convexityDefect() in OpenCV that helps us find convex defects. The function call is as follows. If you want to find a convex defect, the parameter returnPoints must be False when using the function cv2.convexHull to find the convex hull.

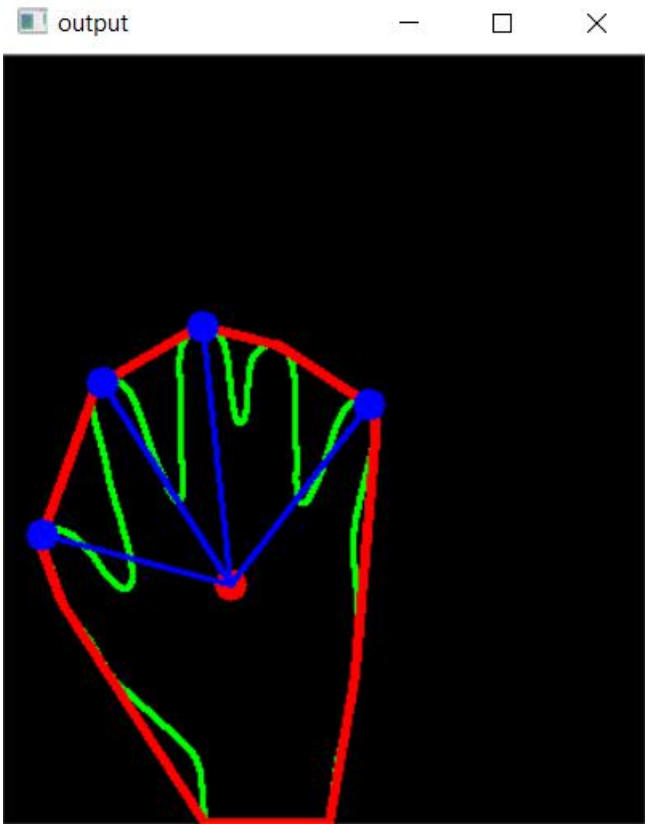

Figure 4

2.4 Generate sound

In my project, we use sine wave to generate sound. Because sound generation and gesture capture use the same loop, cv2's waitkey will block sound, which will cause unstable results, so I use a multi-threaded way to let sound play through the thread to share frequency variables, the

main thread to modify the frequency through cv2 gestures So that the sound will play continuously.

2.5 Gesture Control

Keep sound going. Set up functions which update volume and frequency. Call these functions in main function so that I can use hand to control the volume and frequency of the sound. When I move my hand to right, the sound will go up. When my hand move left, the sound will go down. When the camera detect the area of my hand become larger, the frequency will become larger as well. On the contrary, the frequency will become smaller. When I press ‘b’, computer will capture gesture, when I press ‘r’, computer will reset background and press ‘ESC’ to exit.