Creating a instrument based on movements in webcam view

Hamid Shayestehmanesh

July 6, 2017

Contents

0.1 Main Project Idea

The main idea in this project is to build a HCI application which can shape the music and users can draw it. In other words, we want to design and build an application which is able to translate hands movements to meaningful sounds and build a new electronic music instrument.

0.1.1 More Specified

The goals of this project is to design an architecture of such application and build a prototype to detect hands and their movements. To do so, we use different colored cameras and regular speakers are used to generate sounds. Devices and implementation environment will be discussed in details later.

0.2 Literature

This section is the most important section in this report. All next parts are based on information explained in this chapter. This section tries to translate needed knowledge of music from musician language to computer scientists language thus they can use the information to build different applications related to music.

0.2.1 Music Language and Important Information

Any one of the systems of signs and words that are used by a particular group of people to communicate and transfer knowledge about a specific subject is called a language(Lan, 2017). Nowadays, translation has become very common but, it faces different difficulties. Finding the best word in destination language, demonstrating emotions, feelings and, keeping the structure of the article are some of the hardest problems of translating. Here, we are trying to translate language of a musician to engineering language. So, we will explain different important aspect of musical notes which are mandatory to know about while building a new instrument. To reach our goal we explain what is note language and state it in engineering form to find out what is important to write a music.

Note Language

Note language like other languages contains some sign which works as it's letters and they are written in lines. Human languages are usually written in a line with one lane however, note language has 5 lanes and they are very important. Each note has an actual sound, a length of time and power(bolandi seda).

Octave: Octave of a note is determined by its place on the lanes. It's determined by the start place and end place of the note vertically. For example in figure x first note starts in lane 3 and ends in lane 5.

Note: Notes are determined as the same as octave. Actually the vertical place shows note and octave together.

Octave and Note: Octave and Note are used together to find out the exact frequency that the music should be played in.

Length of Time: In the note language length of time is handle with the shape of a note. Any article in note language has a base time and each note means a ratio over the base time. For example, note **XXX figure folan!** is 1/2 of base time. If the base time of a music is 10ms then this note means 5ms therefore, the player will played a frequency specified by the place of the note for 5ms.

However with knowing time and frequency of each note this is possible to write a meaningful song but there's some more basic operators which can make a simple music to famous magical one.

dditional signs on top of the notes: Except the frequency and length of a note we should also determine how loud should it be played. Looking deep in music we can find this is meaningless to give each note a number or sign to demonstrate the loudness value. This would make the language very complex and also we should define loudness with measurable variable which is actually hard to understand for a musician. Musician mostly decides by intuition thus, they don't like counting! The language notify the player to play louder or more quite and also it shows where is the pick and how loud the pick is. In engineering aspect it may seem very obvious what to do when jumping to next note. You subtract your measure of current loudness from the pick and then divide it by the number of jumps but, it is not that simple in reality. As mentioned we are trying to design and implement a prototype so the obvious idea would be sufficient for now.

Talking about loudness this should be mentioned that loudness is different to harshness. When we make a music louder the frequency and length and the feeling doesn't change but when we make a music harsher we exactly change the way it is. For example, compare the song "Maste Cheshat" by Ebrahim Hamedi sang in 1379 with the same song by E. Hamedi sang in 1394. Same song, same lyrics, same singer but the harshness is very different. You may dance tango with first one but use the second one in disco! In a better word the feeling of the music can change it completely. Again, we skip the feeling and keep with the three basic principles mentioned.
Addition to frequency, length and loudness of a note many other things can be gain from note language like Bemol, Diez or Dot. We will omit more information of the language as they actually give advance data about these three principles or how to acrobat between them. They will complicate our report however it may have no gain for us.

0.2.2 Meta Data

Each music written in note language has some meta data. This meta data includes the speed of the song or in other words how long should a note with one unit of time takes. What should the feeling of the musician be? A few other information exists but they have no important affect on our architecture.

0.2.3 Conclusion

To conclude from an engineer approach, to play a song we need to know frequency and loudness of the song and also this have some different ways to move from a frequency and loudness to another frequency and loudness. It must be able to move fast or slowly.

0.3 Architecture

0.4 Implementation

During this section, first the prototype is discussed, then we will discuss one advanced input.

0.4.1 Prototype

Abstract

A minimal model is implemented using Python, OpenCV and Pygame based on our proposed architecture. In this model there is an element detected by camera which frequency and amplitude is computed base on its coordinates. Coordination of the element identifies by color segmentation.

Prototype

The prototype works with one specific element. The idea is to determine coordination of the object and use it as inputs. To find the element coordinates, we use color segmentation, in other word, we convert every pixel to black except those have a color with some special characteristics. Filtering the color or color segmentation will give us a black and white frame which everything should be black but the element due to the fact that it is built in the color. In practice some other small areas will be converted to white. **figure folan! noise! fogire az 2 3ta az hamina ke noise dashte bashe.** To solve this problem we add a noise reduction procedure after getting the black and white image. *Eroding* and *Dilating* the image will fairly solve the noise problem. Eroding will :definitions: . Dilating will :definition:. By performing Erode and Dilate functions small white pieces will be removed *figure folan! result of the noise removed after ouna.* It will also remove around the element part which may turn it to a few parts. Dilating is used to merge the main white unit if it's affected by the Erode function. In next step we should identify the element by one point to track it's movements. To detect the center of element, first draw a contour around each white color region. As a contour won't usually have a good shape, find the minimum circle around the contour. This makes a good represent of the element coordination in frame. Choose the biggest circle if there is more than one contour. Also the center of circle is a good *one-point* identifier.
Next, by comparing two last center of this imaginary circles can help us detect any movement of the element. Now that we have coordination and movement, we can update our frequency and amplitude. Generate and play a sound with specified frequency and amplitude using Pygame. Moving vertically changes frequency of sound and move it horizontally change sounds' amplitude.

0.4.2 Object Tracking

As mentioned we need to detect and follow an object in camera streams. In this section we will demonstrate how and why the object is evolved. We will discuss

each element one by one and explain their flaws.

Try 1

The first two objects are recognized by their size and color in frames. The problem with these objects is that if any object with the same color appears in the frame and has an appropriate size, the system may follow the wrong object. So the system is not robust.

FigureObject1 Used from top (right image)

Try 2

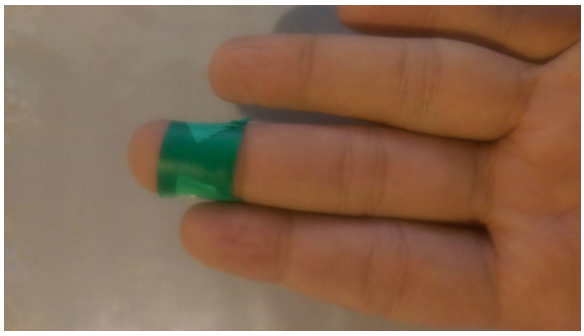
One easy idea to solve the problem is to increase the size of the object. This way it become the largest object and must be recognized easier.

FigureObject2 Used from side

Due to the fact that this step still recognize the object by size and color increasing the size won't help, mostly because of noises, sometimes the contour around the detected color becomes small specially in fast pace movements. Also increasing the size of the object has two consequences, first due to big size of the contour around the detected object it cannot move upon one direction much. It's shorter than what it should be. A little noise can change the center's coordination too much and make the system very vulnerable to noise. To sum up, an element with this size is far from practical.

Try 3

In next attempt we need to resize the element and one idea to solve the size problem is to create a local background, therefore the colored point won't be lost in the background color. First attempt with this idea is one stick on hand.



FigureThe green color was seperated to detect the point

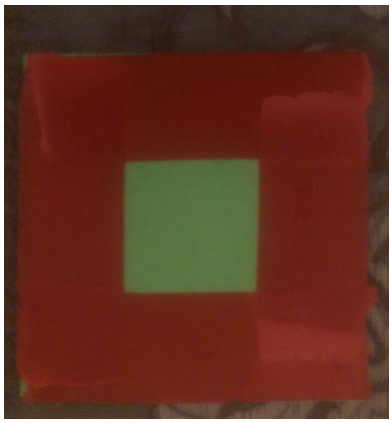
Try 4

Even though a green stick on the finger seems detectable but in practice the color of skin is not very stable so it isn't very helpful as a background, also light has a enormous effect on the color.

Try 5

To solve the problem some red stick are added up and down and then on left and right. Even the system works fine with only up and down red sticks but adding left and right ones will help the system when very bright lights exists in the frame and makes it more robust to light.

Due to the fact that sticks are very annoying on hands I created something with papers.



FigureCreated with papers

Try 6

The papered object worked fine but because of the small background sometimes following the point became hard. To solve the problem we use the fact that one's hand won't move very fast, so if name current position of point $p_1 = (x_1, y_1)$ and next position $p_2 = (x_2, y_2)$ then the distance between p_1, p_2 is less than a

constant C. So if we know the place of object's center, so we don't have to search the hole next frame to find the point. We have to search only a part of it. A square with upper left point of $p_1 - (C, C)$ and the lower right point of $p_1 + (C, C)$ seems adequate to search. With this object the system works very find and follows the target very well but it is common to lose the target in sharp lights.

Try 7

If we could ever make the background bigger to omit sharp lights more, the system could work in more complex situations. We created a glove with a circle on it. The big background it creates around the circle and the local square search mentioned in try 6 helps the system to track the point very well even if sharp light exists.

Different colors like, blue, black, white, light red, dark red were test as glove and the best color to use as a local background for a green circle is dark red. Not a very standard evaluation has been done to choose it. Colors have been test by sticking a green color on them and them using the system and then the best color has been chosen by author(tester).

0.4.3 Playing Notes

To play notes we break the frame into 9 parts, one for silence and 8 others for 8 different notes. Shown below. Each .2 seconds the place of object is detected and a new note based on object coordination will be played.

0.4.4 Counting Number of fingers

(**?**)

0.5 OpenCV

0.6 Future Work

It can be implemented on Android or any other mobile frameworks.

0.7 Other Technologies

A programming language called Chuck has been created in Princeton university. ChucK is a programming language for real-time sound synthesis and music creation. The problem with this language is that it's doesn't support any library which can use webcam or any type of camera, neither easy way to connect it to another programming language.

Bibliography

Definition of language, May 2017. URL https://www.merriam-webster.com/dictionary/language.