Final Project Report

교과목명	창직종합설계프로젝트1	교과목번호	101811-6
학점(설계)/시수	3학점/3시수(2/1)		
개설학과(학년)	컴퓨터공학전공(4학년)	담당교수	박준철
과제명	딥러닝을 활용한 가상 악기 연주 Playing virtual instrument through deep learning		
팀명	조찬아(Isn't it Good?)		

팀원	학번	이름	연락처/e-mail
	B611128	유창조	010-2169-9975 / changcho77@naver.com
	B711121	윤석찬	010-9922-0729 / alsdal@naver.com
	B835176	박현아	010-5765-1038 / millyoum@gmail.com

1.

2. Project Overview

We are planning a program service to play a virtual piano with hand movement by using a hand gesture recognition algorithm. We will train CNN models with data and use it to recognize real-time images to make sound sources. The users of the program can tap their fingers on a sheet of paper with a keyboard drawing printed on it and play it like a real piano.

3. Project Results Summary

- describe in detail the final status of your project design and implementation results as below

3.1 Requirement Analysis

- define/describe system requirements and analysis

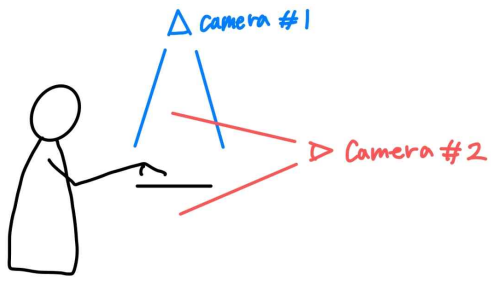
- keyboard recognition: separated white keys from the frame and sectioned them.
- finger recognition: recognize the fingers at the speed of our needs
- finger movement recognition: allocate a certain function(import keyboard) when the finger points are on the key

3.2 System Design (Synthesis and Analysis)

- overall structure, data and control flow, data structures and algorithms, user interface, interface to DBMS, …

The user uses two cameras. One camera recognizes the keyboard and the fingertips to determine which keyboard area the finger entered. The other camera is used to determine whether the keyboard is pressed or not by determining the height of the finger.

The user sets the paper piano in front of the camera and makes the keyboard recognition model detect the keys.



1. When you show your hand the hand detecting model creates key points of the 5 fingertips of each hand to detect whether the keyboard is touched or not with a pre-trained model (mediapipe hand model).
2. The first camera tracks the coordinates of the fingertips and recognizes the fingertip entering which area of the keyboard.
3. The second camera determines the height of the fingers.
4. When the fingertip enters the recognized key area and the height of the finger is lower than the height criteria, the matching sound of the key plays until the finger moves out of the key area.

3.3 Development Environment

- **OS, programming languages, software, tools, IDE, ...**

Programming languages

- Python

Library

- Opencv: real-time image processing
- Mediapipe: detecting the hand and return coordinates of the fingertips

3.4 Test and Demonstration

Piano recognition (piano_recognition_module.py)

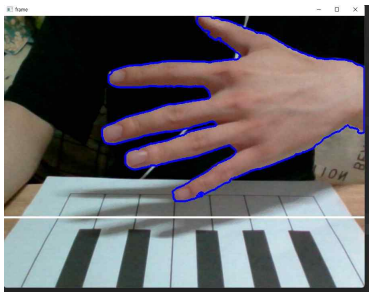
- We don't need to recognize the piano at every frame. We can recognize it and have a fixed pixel area for the keys. So we developed the module based on a single frame. Used simple contours finding algorithms and masking.
- The algorithm to find black and white keys are almost the same. After detecting them separately, we combined them to one keyboard contour and connect other modules.

Hand recognition using mediapipe hand-model.

- The hand recognition model provided by mediapipe provides 20 landmarks. You can interact with the keyboard by using the coordinates of the end point of the finger (4, 8, 12, 16, 20 of the landmark) used when pressing the keyboard.

In the right photo, a blue circle is temporarily drawn at the tip of the finger for visibility.

Hand recognition using RGB data(skin detection)



Using the RGB data of the image, the area included in the rgb range of skin color is recognized as skin.

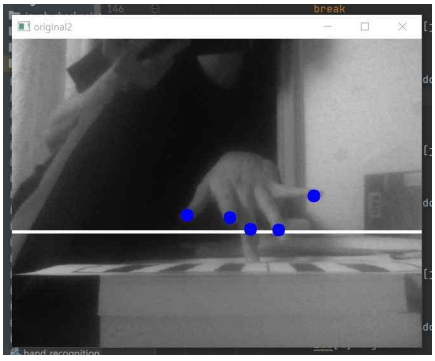
This recognition model is used to determine whether the keyboard is pressed or not.

4. Problems Encountered and Changes from Original Plan

After adding the black keys we found the problem that it was impossible to press the black keys without the fingers crossing the white key areas. Since we didn't have a complete method to determine if the fingers are pressing the keys or not when the fingers are in the keyboard area, the white key sounds were automatically played when your hand is moving to press the black keys. So it was inevitable to add a new camera in front of the hands to determine whether the finger is pressing the key or not by detecting the height of the fingers.

Finger Height detection

- In detecting the fingers' height from the front camera, first we used the same mediapipe hand detecting module. Using the same finger index numbers from the top camera, it was easy to detect which finger was pressing the key but had a fundamental problem having poor accuracy in detecting the hands from the angle in front of the hands.

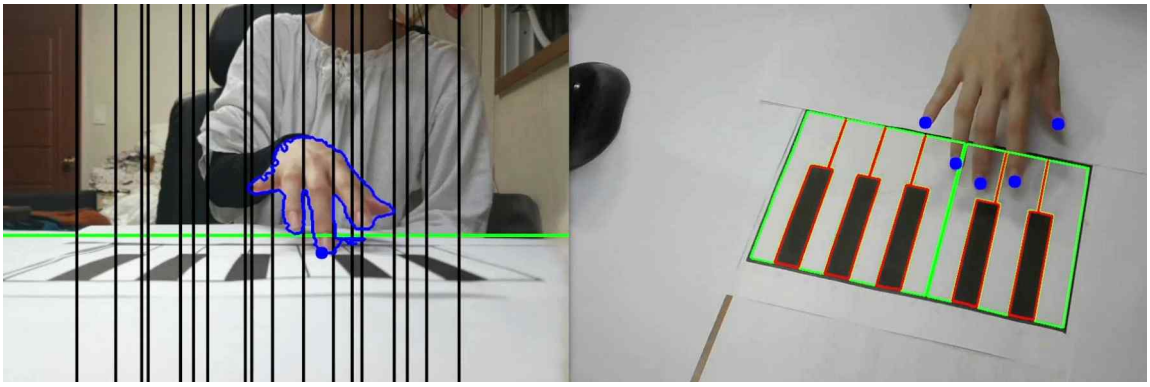


- We decided to use skin detection for better speed and accurate detection of the fingers. The skin detection module detects the height by making an area of the detected RGB colors similar to the skin in the video captured. The only problem with using this method was that it was impossible to detect each identical finger from the front angle view. So we solved this problem by additionally dividing the keyboard areas from the front camera.

Dividing the keyboard areas

- We manually set the height and divide the keys. The key will be played when ‘the finger is inside the key’, ‘the finger is under a certain height(green line)’, and ‘the finger is on the area of the key’.

  The second camera detects rather the finger is pressing or not, and which area the finger is at. The first camera determines which key is being pressed in more detail and narrows it down to one certain key.



Online service

- The planning includes providing the program to online service. However, the quality of the program itself seems more important, so decided to defer the online service part of the project and focus on solving problems in the program.

5. Team

유창조: Hand recognition

윤석찬: Hand recognition

박현아: Piano Recognition

References

[1] https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html

[2] Augmented Piano Reality, October 2015, International Journal of Hybrid Information Technology 8(10):141-152 DOI:10.14257/ijhit.2015.8.10.13

[3] https://pyimagesearch.com/2015/04/20/sorting-contours-using-python-and-opencv/

[4] https://github.com/nvs-abhilash/tutorials/blob/master/tutorials/opencv_contour_scale_rotate/Scaling%20and%20Rotati_ng%20contours.ipynb

https://www.onemotion.com/chord-player/