A PERCEPTIVE MOBILE PROGRAM TO ANALYZE FINGER POSTURE ON THE PIANO USING MACHINE LEARNING AND OBJECT DETECTION

Yuhang Zeng[1], Jonathan Sahagun[2]

[1]Troy High School, 2200 Dorothy Ln, Fullerton, CA 92831
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

I wanted to solve this problem due to noticing many new pianists having issues with their hand posture. Therefore, my project solves the problem by making an app that uses a phone's camera in order to track finger movements [1]. Since smart phones are extremely commonplace, this solution is available for most people. A firebase server is used in order process images from the app and return numbers back to the app in order to display it [2]. The app also tracks practicing sessions and displays data about the hands in every session. I had to optimize a lot of parts of the program, including determining the frequency in which a frame is sent to be analyzed to the server. I also experimented with testing out the different phone angles as well as the latency of the server. Overall, this is an innovative app for improving piano posture for new pianists by encouraging good practicing habits.

KEYWORDS

Piano, Tracking, Computer Vision, Object Detection

1. INTRODUCTION

Every year, many people try to learn the piano, and a common issue with new pianists is bad finger posture. Usually, their hands are either arched too high, or too flat [3]. This creates a bad habit for them and could restrict them in reach and could cause hand pain. When I first started playing the piano, for example, I didn't have the best hand posture. That was pointed out by my teacher and I had a hard time with fixing it. My younger sister had this problem when first starting out with the piano. While this doesn't seem like a big issue early on, this can evolve into a bigger issue farther down into one's piano-playing journey. Therefore, I am trying to solve this issue. Proper piano playing posture allows players to execute complex finger movements accurately and precisely, while poor piano playing can lead to more mistakes being made, especially when playing advanced pieces of music. Good piano playing posture is also fundamental in having a large range of motion: in more advanced pieces, often pianists have to reach for an octave or more, and proper hand posture is essential for being able to reach those notes.

The first methodology also tries to correct piano posture by also using machine learning, albeit with a higher level of depth [4]. It uses depth maps in order to analyze the hand with better accuracy. The added technologies add to the accuracy of their solution, but my project simplifies this approach and puts it on a mobile platform for better accessibility.

The second methodology monitors body posture instead of hand posture. It uses 3D motion capture in order to compare the captured images with the correct posture [5]. However, this project doesn’t account for hand posture while mine does.

The third methodology attempts to solve the same problem through utilizing both visual as well as audio cues in order to determine the best posture for pianists. However, this method is only tested with the right-hand view in mind, while my project works on both sides of the piano.

My solution is an app that uses a phone’s camera in order to track one’s hand movements.This solves the problem by allowing the pianist to look back and see when their hand postures were good and when it needs improvement. This is an effective solution because this allows the pianist to go back and reflect after practicing. The feedback can be received in just a few seconds after playing, which means this solution has semi-real time feedback. The pianist can also look back on their old data in the sessions tab. Therefore, a person can review their past practices if so desired. This has the added ability for pianists to keep track of their practicing schedule, encouraging them to practice more with better efficiency. This solution is very accessible, as anyone with a phone can download this free app and start improving their skills.

The first experiment I did was to test the accuracy of the software when the phone is placed in different angles with respect to the piano. I set up the experiment by playing the piano with the app recording my hands at two different angles. I found that the angle does play a significant role in the accuracy. I found that the best accuracy comes when the phone is placed directly above the phone due to being able to see more fingers. The second experiment was a test on the latency of the software. I set up the experiment by adding a latency stat next to the other numbers and accessed it through the firebase server [6]. I discovered that the average speed for the server to process the image is about 2.5 seconds, though this number was skewed severely by some outliers, possibly due to slow internet or lost packets.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The Frequency

When implementing the discover page, or the camera feature part of the program, I had to think about the frequency in which the program takes a picture from the camera and sends it to the server to be analyzed. I could make the program send multiple frames a second, which would make things easier to analyze and better for accuracy, but that would make the processing a lot slower as the server has to deal with more images every second. I can also make the program send one frame every few seconds, which makes the server processing faster, but that comes at a cost of having less accuracy on the stats.

2.2. The Hand Tracking

Another component of the program would be the hand tracking [7]. The hand tracking isn't 100% accurate and can often mistake correct fingers as incorrect and vice versa. This is caused by machine learning just not being correct all the time. There is no way to completely resolve the problem, because there is always going to be machine error with a new technology, but there are a few ways to optimize the program so it produces the best results with minimal error. First, the app takes snapshots every few seconds, so that even with one bad snapshot analysis, the overall accuracy of the session isn't terribly screwed. Next, the location of the phone could be adjusted to a position where the overall accuracy is good.

2.3. Tracking the Hands in Different Positions

Another component of the program is tracking the hands in different positions. The app doesn't function great when placed on the side of the piano, due to not being able to see all the fingers. However, the phone is best placed here on the side, due to the side of the piano providing support for the phone. The alternative to this is to place the phone more in the middle, where it can see and recognize all the fingers to the hand. This comes with the cost that it's harder for the phone to be placed, but this can be resolved with a phone stand, tripod, or a DIY phone support device.

3. Solution

The 3 major components that the program link together are the MediaPipe server, the application on the phone itself, and the piano itself [8]. First, the user of the app plays the piano, with the phone in the background recording. The app uses the camera on the phone to send snapshots occasionally to the server for processing. The app controls how long the interval is for each snapshot of the camera to be sent to the server for analysis. The server then takes the photo and processes it, returning back to the app several useful data, such as the the number of hands detected, the number of correct fingers, and the number of correct hands. This is done by calculating the hand posture. The curvature of the fingers are calculated, and should it exceed a certain threshold, it is marked as a failed finger. This is to ensure the hand posture of the pianist is in a correct position. If more than 3 fingers are correct, then the entire hand returns correct. If less than 3 fingers are correct, then the hand returns incorrect. All of this data is then sent back to the app via internet. Then, the data is displayed by the app. After the session is finished, the app itself does some processing that calculates the percentage of correct fingers. A mediapipe server is used to make this program [9]. Mediapipe is a pose estimation server that recognizes the different poses of objects. This is applied to the fingers to calculate the angles of the fingers in the aforementioned process of snapshot analysis. This is a process hosted on firebase.

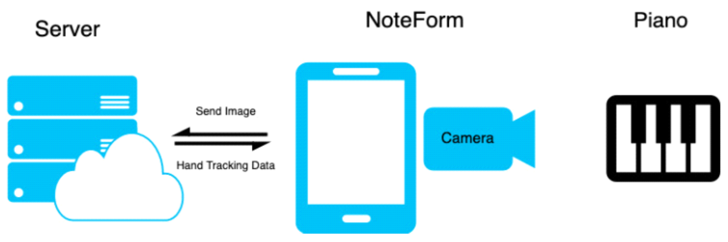


Figure 1. Overview of the solution

One of the components I established in 3.1 is the mediapipe server. The component's purpose is for pose estimation, and this is applied to the fingers on the snapshots of the fingers to calculate the correctness of each finger through calculating angles. The component relies on machine learning, and has been trained with sample images that are correct.

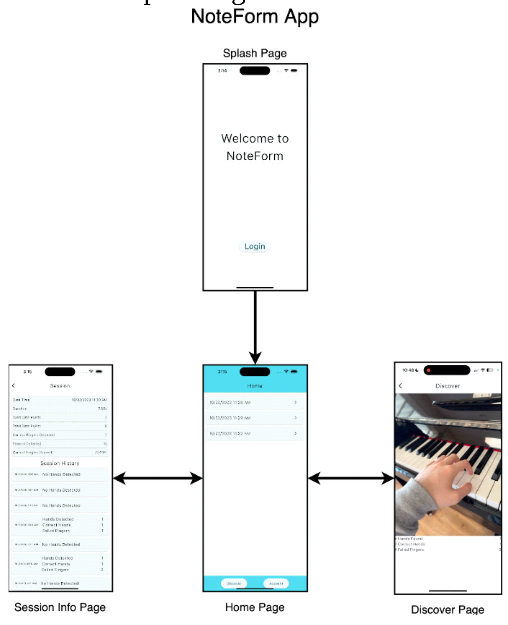
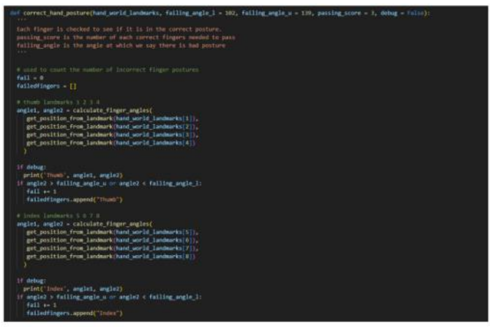


Figure 2. Screenshot of the APP

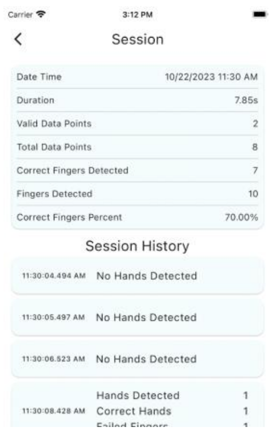


```
def correct_hand_posture(hand_world_landmarks, failing_angle_l = 102, failing_angle_u = 139, passing_score = 3, debug = False):
    '''
    Each finger is checked to see if it is in the correct posture.
    passing_score is the number of each correct fingers needed to pass
    failing_angle is the angle at which we say there is bad posture
    '''

    # used to count the number of incorrect finger postures
    fail = 0
    failedfingers = []

    # thumb landmarks 1 2 3 4
    angle1, angle2 = calculate_finger_angles(
        get_position_from_landmark(hand_world_landmarks[1]),
        get_position_from_landmark(hand_world_landmarks[2]),
        get_position_from_landmark(hand_world_landmarks[3]),
        get_position_from_landmark(hand_world_landmarks[4])
    )

    if debug:
        print('Thumb', angle1, angle2)
    if angle2 > failing_angle_u or angle2 < failing_angle_l:
        fail += 1
        failedfingers.append("Thumb")

    # index landmarks 5 6 7 8
    angle1, angle2 = calculate_finger_angles(
        get_position_from_landmark(hand_world_landmarks[5]),
        get_position_from_landmark(hand_world_landmarks[6]),
        get_position_from_landmark(hand_world_landmarks[7]),
        get_position_from_landmark(hand_world_landmarks[8])
    )

    if debug:
        print('Index', angle1, angle2)
    if angle2 > failing_angle_u or angle2 < failing_angle_l:
        fail += 1
        failedfingers.append("Index")
```

Figure 3. Screenshot of code 1

This code defines a function called correct_hand_posture that checks the posture of a hand based on the angles formed by the landmarks of its fingers. The function takes several parameters: hand_world_landmarks: The landmarks of the hand in the world coordinates. failing_angle_l: The lower threshold angle for considering a finger posture as bad. This is set to 102 degrees. failing_angle_u: The upper threshold angle for considering a finger posture as bad. This is set to 139 degrees. passing_score: The number of correct finger postures needed to pass. The default value is 3. debug: A boolean parameter that, when set to True, prints debug information about the angles of each finger. The function calculates the angles for each finger using the calculate_finger_angles function, which presumably takes four positions corresponding to the finger landmarks and returns two angles. The angles are then compared to the specified upper and lower threshold angles (failing_angle_u and failing_angle_l), and if the angles fall outside this range, the finger posture is considered incorrect. The function keeps track of the number of failed fingers (fail) and maintains a list of the names of the failed fingers (failedfingers). After checking all fingers, the function returns a tuple: The first element is a boolean indicating whether the overall hand posture is correct (i.e., the number of failed fingers is less than the passing score). The second element is the number of failed fingers. The third element is a list containing the names of the failed fingers. The function is designed to be used as a part of a larger system or application that involves hand pose analysis or gesture recognition, and it provides a convenient way to evaluate the correctness of hand postures based on finger angles.

One of the components I established in 3.2 is the session history section. The component's purpose is for keeping track of the history of practice sessions in order to help users learn from their mistakes. The component replies on storing data on in the system and accessing that data.



Date Time	10/22/2023 11:30 AM
Duration	7.85s
Valid Data Points	2
Total Data Points	8
Correct Fingers Detected	7
Fingers Detected	10
Correct Fingers Percent	70.00%

Session History

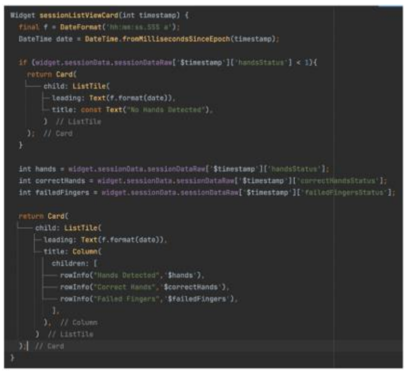
11:30:04.494 AM No Hands Detected

11:30:05.497 AM No Hands Detected

11:30:06.523 AM No Hands Detected

11:30:08.428 AM Hands Detected 1
Correct Hands 1
Failed Fingers 1

Figure 4. Screenshot of components



```
Widget sessionListViewCard(int timestamp) {
  final f = DateFormat('hh:mm:ss.SSS a');
  DateTime date = DateTime.fromMillisecondsSinceEpoch(timestamp);

  if (widget.sessionData.sessionDataRaw['$timestamp']['handsStatus'] < 1){
    return Card(
      child: ListTile(
        leading: Text(f.format(date)),
        title: const Text("No Hands Detected"),
      ) // ListTile
    ); // Card
  }

  int hands = widget.sessionData.sessionDataRaw['$timestamp']['handsStatus'];
  int correctHands = widget.sessionData.sessionDataRaw['$timestamp']['correctHandsStatus'];
  int failedFingers = widget.sessionData.sessionDataRaw['$timestamp']['failedFingersStatus'];

  return Card(
    child: ListTile(
      leading: Text(f.format(date)),
      title: Column(
        children: [
          rowInfo("Hands Detected", '$hands'),
          rowInfo("Correct Hands", '$correctHands'),
          rowInfo("Failed Fingers", '$failedFingers'),
        ],
      ), // Column
    ) // ListTile
  ); // Card
}
```

Figure 5. Screenshot of code 2

The given Flutter code defines a function named sessionListViewCard that takes a timestamp as an argument and returns a Card containing information about a session. It uses the DateFormat class from the intl package to format the timestamp into a human-readable time format. The timestamp is converted to a DateTime object. The function then checks if the number of hands detected at the specified timestamp is less than 1. If true, it returns a simple card with a ListTile indicating "No Hands Detected." If the hands detected are greater than or equal to 1, the function retrieves additional information about the session, including the total number of hands detected, the number of correctly detected hands, and the count of failed fingers. It then constructs a more detailed card with a ListTile displaying the formatted timestamp and a Column of information about hands, correct hands, and failed fingers using a helper function called rowInfo. Several of these “cards” are made per session, effectively displaying the data in the session history.

Another component in the program is the discover page tiself. this compoennt is used in order to gather and send data to the pediapipe server. The phone's camera is used to implement this system. The component doesn't rely on a special concept. It just takes frames of a camera and sends it to the mediapipe server for analysis.

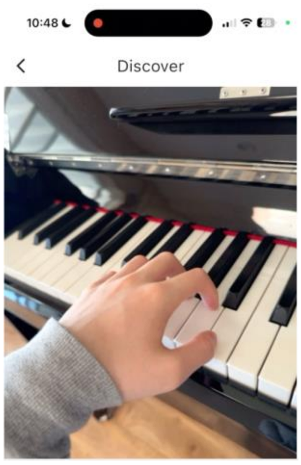
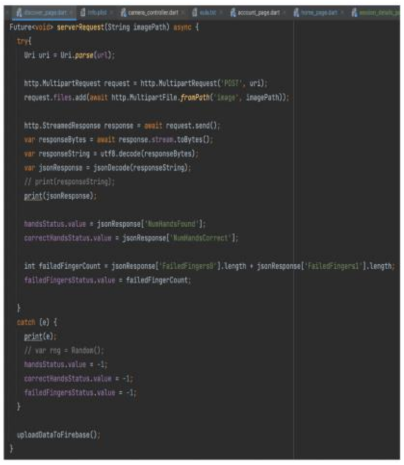


Figure 6. Screenshot of Discover picture



```
Future<void> serverRequest(String imagePath) async {
  try{
    Uri uri = Uri.parse(url);

    http.MultipartRequest request = http.MultipartRequest('POST', uri);
    request.files.add(await http.MultipartFile.fromPath('image', imagePath));

    http.StreamedResponse response = await request.send();
    var responseBytes = await response.stream.toBytes();
    var responseString = utf8.decode(responseBytes);
    var jsonResponse = jsonDecode(responseString);
    // print(responseString);
    print(jsonResponse);

    handsStatus.value = jsonResponse['NumHandsFound'];
    correctHandsStatus.value = jsonResponse['NumHandsCorrect'];

    int failedFingerCount = jsonResponse['FailedFingers0'].length + jsonResponse['FailedFingers1'].length;
    failedFingersStatus.value = failedFingerCount;

  }
  catch (e) {
    print(e);
    // var rng = Random();
    handsStatus.value = -1;
    correctHandsStatus.value = -1;
    failedFingersStatus.value = -1;
  }

  uploadDataToFirebase();
}
```

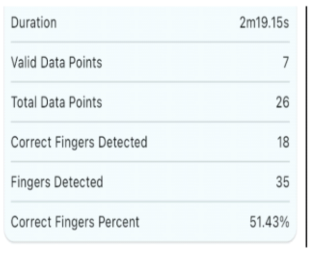
Figure 7. Screenshot of code 3

This given code first takes in an image. In this case, the image is the freeze frame of the camera with the hand playing the piano. It converts the image into the uri format and sends the request to the mediapipe server for analysis [10]. If a response is received, the number of hands found is stored in the variable “NumHandsFound”, the number of correct hands is stored in the “NumHandsCorrect” variable, and the number of failed fingers is stored in “FailedFingers”. If there is no response for any reason, the values of all the variables are all set to -1, and nothing is returned.

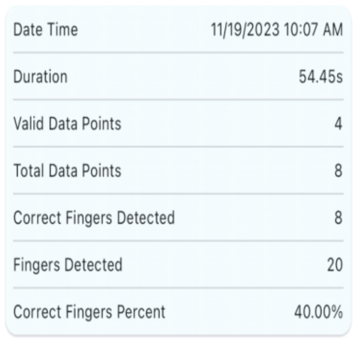
4. EXPERIMENT

4.1. Experiment 1

This given code first takes in an image. In this case, the image is the freeze frame of the camera with the hand playing the piano. It converts the image into the uri format and sends the request to the mediapipe server for analysis. If a response is received, the number of hands found is stored in the variable "NumHandsFound", the number of correct hands is stored in the "NumHandsCorrect" variable, and the number of failed fingers is stored in "FailedFingers". If there is no response for any reason, the values of all the variables are all set to -1, and nothing is returned.
The experiment will be with two different angles. First, the phone will be placed in the optimal angle and the hand will play with 100 percent correct posture. Next, the phone will be placed on the side and the hand will also play with 100 percent correct posture. Next, I will see how many fingers the program calculated as "wrong" with each angle. Since the hand is played with correct posture, the less the calculated wrong values there are, the better. The experiment is set up this way to measure how much error each angle induces without adding in other variables that mess with the experiment.



Duration	2m19.15s
Valid Data Points	7
Total Data Points	26
Correct Fingers Detected	18
Fingers Detected	35
Correct Fingers Percent	51.43%



Date Time	11/19/2023 10:07 AM
Duration	54.45s
Valid Data Points	4
Total Data Points	8
Correct Fingers Detected	8
Fingers Detected	20
Correct Fingers Percent	40.00%

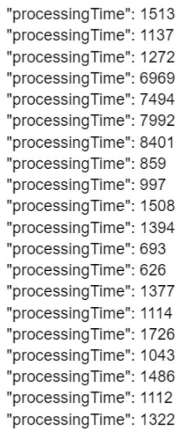
Figure 8. Figure of experiment 1

The data shows that the model is affected by the angle in which the phone is placed. The results show that the phone had a hard time recognizing hands when the phone is placed on the side of the piano at a 90% angle. Contrastingly, the app works marginally better when it is placed directly above the hand. This makes sense as it is easier to calculate the angles of all the fingers when all the fingers are visible. When the phone is placed on the side of the piano, only a few fingers are seen. With the design of the app, 3 or more fingers constitutes a correct hand, and since the phone can barely detect 3 fingers at times, inaccuracies will occur more often. There is no specific way to fix this entirely, as the app is just designed to use the 3 fingers as a benchmark to determine whether a hand is right or failed. One possible solution is to change the way fingers are detected to be relative to the number of fingers detected.

4.2. Experiment 2

Another blind spot in the program is the speed of the response of the feedback from the server. This is important because the feedback should be as soon as possible in order for the user to learn from the feedback.

The experiment will be conducted via adding a new feature to the server that measures the response time. On the firebase server, new code is added that allows the server to output the response time on the data server. This doesn’t change what is displayed on the app itself, but in the server now the latency is displayed. Next, I played the piano with the phone placed on the side of the piano for a few minutes. Next, I look at the data from the server and extract the latency.



```
"processingTime": 1513
"processingTime": 1137
"processingTime": 1272
"processingTime": 6969
"processingTime": 7494
"processingTime": 7992
"processingTime": 8401
"processingTime": 859
"processingTime": 997
"processingTime": 1508
"processingTime": 1394
"processingTime": 693
"processingTime": 626
"processingTime": 1377
"processingTime": 1114
"processingTime": 1726
"processingTime": 1043
"processingTime": 1486
"processingTime": 1112
"processingTime": 1322
```

Figure 9. Figure of experiment 2

The mean value of the data is 2501 milliseconds. The lowest value is 626 milliseconds while the highest value is 8401 milliseconds, which equates to a range of 7775. This shows that the latency of the data transferring from the phone to the server is inconsistent and can vary greatly. However, most of the values are actually in the 1000 milliseconds, which is reasonably fast. This delay is acceptable because an one-second calculation time allows the user to receive feedback quickly in order to change their posture. The outliers could be due to differing internet speeds, which transports the data to and from the server. Other factors that affected the data is the processing speed of the firebase server, which can be slow at times due to having to process many requests in a short time period. Overall, the latency of the app isn’t bad, but there aren't a lot of things that can be done to lower the latency. One possible solution to minimize the latency is to rework the code in the processing server so it’s more efficient in order to reduce runtime.

5. RELATED WORK

This problem has been attempted by MIT students in 2020 [11]. Their solution also uses machine learning, but uses more complex algorithms and mechanics, such as utilizing depth maps to determine depth context features. The solution segments hands from the depth maps for analysis. This solution is extremely effective as the added technologies contribute to a more accurate result. My project isn’t as accurate as their project due to the low amount of technology used, but I am able to use this technology in the form of a mobile application with instant feedback, which makes my own project valuable for many users.

This problem has also been attempted somewhere else [12]. However, the same problem is being solved in a different way. The way this group attempted to solve the problem is by monitoring the body posture rather than the hand posture. This model uses motion capture technology and 3D visualization in order to compare the posture to the correct posture of someone playing the piano. The solution is effective because it can be used to calculate posture given any type of view. Therefore, it is unaffected by the angle of where the camera is located. However, the limitation of this method is that the pianist may still have poor hand posture given a good body posture. My project therefore attempts to focus on hand posture only.

This problem has also been looked at through a completely different lens [13]. The third group attempted to solve this problem through using both visual and audio cues in order to determine the correct posture for pianists. The solution utilizes an audio-visual tensor fusion network. This solution utilizes the traditional visual tracking, but adds a new layer in the form of using a database that represents the visual-audio information in order to account for times when the visual quality isn't the best, which makes this solution very effective. One limitation of this solution is that only videos of pianists playing on the right side of the piano were used. My solution uses angles in order to calculate posture, so it should work on both the left and the right side of the piano. If my solution included audio processing, I would be excluding a selective number of users, as every user's pianos are different. Some pianos don't have the same dynamic range and thus the solution would not work as well for some users, limiting the accesibility of the app. Therefore, audio processing was not omitted for my app.

6. CONCLUSIONS

As mentioned above, one limitation to the project is the limited functionality of the app when placed in an undesirable angle where not all the fingers are in view. This is because the app itself uses geometry between the angles of the fingers in order to determine whether the finger is correct or not [14]. As shown by the other group's attempts to solve this problem, there isnt' exactly a solution that completely fixes the problem, however, if I had more time with my project, I would edit the code responsible for determining whether a hand is correct or not into a more complex system. It could have a lot of additional factors like the angle from the piano, the number of fingers that are visible, and the direction (from left or right) where the phone is placed relative to the piano.

Overall, I learned a lot in terms of technical information (machine learning, object detection, firebase servers) while making this app. In addition, I learned the basics of app creation, UI design, and the steps in order to publish an app on to the app store [15]. Even though the app at its current state isn't perfect, I would still consider the project to be a success.

REFERENCES

[1] Carey, James R., et al. "Finger-movement tracking scores in healthy subjects." Perceptual and motor skills 79.1 (1994): 563-576.

[2] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.

[3] Johnson, David, Daniela Damian, and George Tzanetakis. "Detecting hand posture in piano playing using depth data." Computer Music Journal 43.1 (2020): 59-78.

[4] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." Science 349.6245 (2015): 255-260.

[5] Horprasert, Thanarat, et al. "Real-time 3d motion capture." Second workshop on perceptual interfaces. Vol. 2. 1998.

[6] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.

[7] Zhang, Fan, et al. "Mediapipe hands: On-device real-time hand tracking." arXiv preprint arXiv:2006.10214 (2020).

[8] Veluri, Ravi Kishore, et al. "Hand Gesture Mapping Using MediaPipe Algorithm." Proceedings of Third International Conference on Communication, Computing and Electronics Systems: ICCCES 2021. Singapore: Springer Singapore, 2022.

[9] Lin, Yiqiao, Xueyan Jiao, and Lei Zhao. "Detection of 3D Human Posture Based on Improved Mediapipe." Journal of Computer and Communications 11.2 (2023): 102-121.

[10] Price, Catalina, et al. Universal Research Interchange (URI) Format: PDB to XML format converter. Diss. University of Rhode Island, 2005.

[11] Johnson, David, Daniela Damian, and George Tzanetakis. "Detecting hand posture in piano playing using depth data." Computer Music Journal 43.1 (2020): 59-78.

[12] Mora, Javier, Won-Sook Lee, and Gilles Comeau. "3D visual feedback in learning of piano posture." Technologies for E-Learning and Digital Entertainment: Second International Conference, Edutainment 2007, Hong Kong, China, June 11-13, 2007. Proceedings 2. Springer Berlin Heidelberg, 2007.

[13] Park, So-Hyun, and Young-Ho Park. "Audio-visual tensor fusion network for piano player posture classification." Applied Sciences 10.19 (2020): 6857.

[14] Chow, Vivian W., et al. "An overview of APP processing enzymes and products." Neuromolecular medicine 12 (2010): 1-12.

[15] Ferreira, Jennifer, James Noble, and Robert Biddle. "Agile development iterations and UI design." Agile 2007 (AGILE 2007). IEEE, 2007.