CS 415 – Final Project Report

Personal Views about the Fields and Applications of Computer Vision:

I have always been interested in different fields and areas of Computer Science ever since I was in high school, and Computer Vision was one of the fields I was most interested in, leading me to take CS 415 this semester as one of my tech electives. As with most people, I think of Computer Vision as a powerful tool to help process digital images and videos and recognize objects to assist with and automate various tasks. However, I like to look at this field alongside or in collaboration with other various fields of Computer Science from a very human-centered perspective as human-centered computing is the concentration for my CS degree. To put it another way, just like how I might be interested in seeing if AI can replicate how a human thinks and their brain functions, I wanted to see if a computer can see the world exactly as humans do. While it might not be possible right now, I believe it will one day become a real possibility with more powerful hardware, and in collaboration with other CS fields in the future.

At this point, there are already many different active applications of Computer Vision such as Forensics, Computational Photography, Optical Character Recognition, Face Detection/Recognition, Object Identification/Recognition, Motion Tracking, and so many more. Out of all these applications, I am most interested in those that can apply to and are useful for Virtual and Augmented Reality as CS 428 is another class I am taking this semester, more specifically body/motion tracking. As of now, VR mostly relies on tracking devices with sensors to track the user's body, but I am starting to see more tracking that utilizes hand and even eye tracking using the built-in cameras in VR headsets over time. With further developments in the field of Computer Vision, I believe it will become possible to track and even fully scan the user's entire body to convert into avatars and such for VR soon.

What I Learned in CS 415:

Throughout the semester, I learned a lot more about Computer Vision than I ever did, such as Image Filtering, Edge Detection, Hough Transform, Segmentation, Harris Corner Detection, Feature Invariance, Feature Descriptors, Geometric Transformations, Image Alignment, Random Sample Consensus (RANSAC), Neural Networks, and Convolutional Neural Networks. I was simply listing out all the topics we learned so far, but I did not learn all the topics equally due to various factors. I believe I absorbed the content and materials for all the topics from the first half of the semester (up until Harris Corner Detection) much better than those in the second half of the semester as I was not too busy with other classes at the time. I learned all the topics on which we did mini projects for the best as I got to test them out.

What I Want to Study More:

For this class, I wanted to focus on more practical aspects that use video input instead of just still images. As I mentioned earlier, I am interested in looking more into body/motion tracking, and the various approaches that could be taken for that. I understand this class is mainly an intro class for Computer Vision that focuses on the core topics, but I thought I would at least get to touch on body/motion tracking a bit in the class due to a video clip on it being shown during our very first class, but it never happened in the end. This is why I ended up choosing "Exercise Pose Recognition" as the topic for the final project, which allowed me to explore body/motion tracking on my own. Despite not touching upon this in class, a lot of things I learned throughout the semester helped me understand how things work pretty quickly, which I will explain more in the later sections where I talk about the project. To put it in one sentence, I want to learn and study more about human-related applications of Computer Vision, especially on activity/pose detection and recognition of human or humanoid bodies.

Project – Introduction:

The topic of our CS 415 final project is “Exercise Pose Recognition” where we use Computer Vision to detect and classify the exercise that the user is performing. The idea and motivation for choosing this topic came along when my teammate Lucas and I were discussing what to do for our final project, and got around to talking about exercising and working out, which led to us figuring out the topic, problem statement, and goal for our project.

Project – Problem Statement:

The problem statement for our project is that people exercising on their own, especially those new to working out can have very improper postures without anyone to keep an eye on them, which could lead to very ineffective and inefficient workout sessions despite all the efforts. This could lead to those people getting tired and frustrated without seeing much results no matter how hard they exercised, and could also lead to developing long-term bad habits of having incorrect and improper postures which could become difficult to fix down the line.

Project – Goal:

For the goal of our project, we thought it would be a great idea to have an application on laptops that can use the webcam to monitor the user’s workout session, check if their exercise forms and postures are correct and proper, and give them appropriate feedback through the UI to help them not only check on their form and posture in real-time but also if they are doing the right type of exercise just in case they get the different types of exercises confused. In short, we wanted an application that could act as the user’s very own personal trainer or spotter as they exercise and work out on their own that could ensure the users are performing exercises properly with good form, and even enforce it in a way later on as the application gets more flushed out.

Project – General Design:

As we looked into the resources and tools necessary to work on our project, we quickly realized it might be too ambitious for us to develop an application that can detect and recognize too many different exercise poses, as we also still needed to explore at least two different approaches. Therefore, for this project, we decided to narrow down the scope of the application to focus only on two different exercise poses which are pushups and squats. We also decided to make it so that the application can differentiate between the “up” and “down” variations of the same exercise pose to assist with exercise tracking, and also compare the results better.

As for our approaches to this, we came up with two similar yet different approaches with Lucas working on the first approach while I worked on the second. The first approach detects and calculates the angles between the user’s joints. The joint angles are then used to determine the current status of the user’s pose. The second approach uses an image dataset for the two exercise poses for each status to train a prediction model through machine learning. The trained model is then used to predict the current status of the user’s pose.

Project – Libraries Used:

For both approaches, we are using OpenCV to import, capture, and process images and videos. We are also using MediaPipe Pose to detect the person and their pose in the image or video frame, and predict the pose landmarks and segmentation masks for them.

For the first approach, we are using NumPy to assist with calculating angles between joint landmarks, and Flask to give a selection screen between the two exercise poses.

For the second approach, we are using pandas to export training data to CSV files and predict exercise poses. We are also using scikit-learn to train the prediction model through ML.

Project – Design and Implementation of Approach 1:

For the first approach, we would first have the UI showing users the selection screen with the options to track either pushups or squats. This selection screen UI is a webpage created using Flask. Once the user chooses the exercise, a separate UI window should appear to track the selected exercise. That UI window is pretty much a preview of the user's webcam view, but with the pose landmarks and the tracked "skeleton" overlay displayed on top of the user's body. There would also be a text label displaying if the user is performing either the "up" or "down" variation of the exercise that was chosen in the earlier selection screen.

As for how the exercise poses are estimated, the webcam video input would first be captured, and the video frames would be converted to an RGB image and processed using OpenCV. Afterward, the pose landmarks are extracted from the processed RGB image using MediaPipe Pose. The code would then check if the user was present in the video frame using the extracted pose landmarks data. If the user was present in the video frame, depending on which exercise was selected, certain pose landmarks would be used to calculate the angles between the specific joints required to detect the "up" and "down" variations of the exercise with the help of NumPy. The calculated angles are then used to determine if the user is performing either the "up" or "down" variation of the selected exercise. In the end, the exercise pose status would be displayed on the UI window as visual feedback for the user.

As for the code structure of this approach, there is only one main file called "app.py" which contains all the code. In the code, there are three main functions. The first function is Angle() which takes three landmarks as input to calculate the joint angle. The other two functions are pushups() and squats() which are the main code bodies for each selected exercise which are called depending on the user's choice of exercise in the selection screen UI.

Project – Design and Implementation of Approach 2:

For the second approach, there are no selection screens like the first approach as this approach can differentiate the two exercise types in the same UI window without requiring dedicated windows for each exercise type. There is just one UI window that opens up after running the code, which is a preview of the user's webcam view with pose landmarks and tracked "skeleton" overlay on top of the user's body, similar to the UI window from the first approach. However, this UI window for the second approach displays the text indicating the type of exercise the user is performing in addition to the "up" and "down" status.

We first needed to get the dataset for training the prediction model before running the exercise pose prediction code. For this, we first gathered 20 different images for each exercise, for each "up" and "down" variations from Google Images, resulting in an image dataset with a total of 80 images. The dataset is organized into one main folder, with subfolders with names showing the exercise type, and "up" or "down" variations (e.g. – "Pushup_Down", "Squat_Up", etc…). The images would then be processed, with the pose landmarks data extracted from them using OpenCV and MediaPipe Pose in the same way as the first approach. The extracted data are the xyz positions and the visibility of each pose landmark, along with the name of the subfolder of the image they were extracted from, which is then stored into a CSV file using pandas.

We realized that our dataset might be a bit too small to be reliable at this point, but before gathering the image dataset on our own, we tried looking for publicly available large datasets that are available online on kaggle.com and from other sources we could find. However, we ran into various issues such as the dataset not being in the proper format we needed, being too large for this project, and not having exercise poses separated into "up" and " down" variations.

After getting all the training data in one CSV file in the format that we wanted, we can run the exercise pose prediction code. It will first train the exercise pose prediction model through Machine Learning using scikit-learn. We then get the webcam video input, with the video frames from it processed into RGB images, and pose landmarks data extracted from them using OpenCV and MediaPipe Pose in the same way as in the first approach. Using the extracted pose landmarks data, the code will also check if the user is present or not in the video frame before moving on. This is where it starts to differ from the first approach. If the user is present in the video frame, instead of having the joint angles calculated from a few pose landmarks, all of the user's pose landmarks data will be used to predict both the user's exercise pose type and status. This is done by using the trained prediction ML model and the predict() function from pandas. Afterward, the predicted exercise pose type and status would be displayed on the UI window as visual feedback for the user.

As for the code structure of this approach, there are two main files called "predict.py" and "train.py", alongside the image dataset folder mentioned earlier and the CSV file of extracted pose landmarks data in the same project folder. The "train.py" contains all the code necessary to extract the pose landmarks data from the given image dataset folder into a CSV file. The "predict.py" contains all the code necessary to train the prediction ML model from the CSV file and perform the actual exercise pose prediction. Without the CSV file, it would fail to run.

Project – Note on the Major Difference between Approaches 1 and 2:

Despite being different approaches towards the same goal, the first approach needs user input on the exercise pose to track while the second approach can differentiate it on its own. This is caused by the first approach requiring separate functions for each exercise pose type while the second approach can differentiate whatever was in the image dataset used for training.

Project – Results of Approach 1

During the testing of the first approach by Lucas and I, we found out that the “up” and “down” statuses for the pushups were recognized properly from the front view of the user, but not from the side view. We also found out that the “up” and “down” statuses for squats were recognized properly from both the front and the side views. When it comes to the diagonal and other views, we found the first approach to be quite unreliable.

Project – Results of Approach 2

During the testing for the second approach by Lucas and I, we found out that both of the exercise poses and their statuses were recognized properly for the front view. However, for the side view, pushups and the “up” and “down” statuses of it have some difficulty being recognized properly, while squats and the “up” and “down” statuses of it can be recognized but under very specific situations. When it comes to the diagonal and other views, we found it to be much more reliable than the first approach, but not good enough to be an actual real-world application.

Project – Analysis of the Current Results

As of this moment, I believe the first approach seems to be more reliable for recognizing the “up” and “down” statuses. This could be caused by various factors such as not having a strict recognition confidence level for MediaPipe Pose at the time of testing, the first approach only needing a few certain pose landmarks to be visible to calculate the joint angles for exercise pose estimation while the second approach uses all pose landmarks to predict using the trained prediction ML model, and the first approach not having to differentiate between the different exercise poses like the second approach as it is selected by the user at the start. However, I would say this is as far as it goes for the advantages of the first approach.

Project – Improvements and Future Work

With the right dataset, I believe that the second approach could become a lot more reliable and better than the first approach in many ways. As I mentioned above earlier, I had to gather the images for the training dataset on my own as there were issues with using publicly available large datasets on kaggle.com and other sources, which resulted in a small dataset with only 80 images, with only 20 images for each exercise type for each variation. This could be what led to the unreliable recognition at the moment. With a larger dataset, with data taken from thousands of images of the exercises being performed in different lighting conditions and environments from more different angles, recognition should become far more accurate even in cases where the first approach might fail. As the first approach relies on several key pose landmarks to calculate the joint angles, once one of those pose landmarks is no longer visible due to the lighting, environment, or the direction of the user, it would fail to estimate the exercise pose of the user. However, the second approach uses all pose landmarks and has visibility values for the pose landmarks in the training data. Therefore, even if a few pose landmarks might be no longer visible, it should still be possible to predict the user's exercise pose using all other visible pose landmarks as there are no "key" pose landmarks in this approach.

In addition to the recognition accuracy improvement, the second approach is also much better for future expansion and improvements than the first approach. If we wanted to add new exercises for the first approach, we would have to hardcode a dedicated function for that exercise pose and also add an option for that exercise in the selection screen UI. However, with the second approach, there is no need to change the code at all to add more exercises. We would just need a properly formatted image dataset folder with appropriately named subfolders, and once the prediction model is trained, it should be able to differentiate all exercise types in the dataset.

There are also several general improvements we could make for the overall project and application regardless of the approach. As we did not have too much time for this final project, our application is very barebones, with the emphasis only on the recognizer. If we are to turn this into an actual real-world usable application, there are several features we could add such as an exercise counter that tracks the number of exercise reps and sets the user has done with a way to also input the target exercise reps and sets by the user. We could also add something like a "difficulty" slider that directly modifies the confidence level for the recognizer used by MediaPipe Pose, which would make the exercises either less or more intense. We also did not get to thoroughly test the recognizer for this project before the presentation, so if we wanted to focus on improving the recognition accuracy further regardless of the approach, we could conduct user studies in the future to evaluate the recognizers under more conditions with more people. We could also consider the use of external webcams for higher-quality video input and better positioning of the camera relative to the UI screen. We could even look into improving and optimizing the current approaches themselves given the time. I am certain we could find better approaches and solutions on the internet that we may not have discovered yet, or even figure out on our own from scratch, which is something we could not do before this project.

Project – What We Should Learn More

When we were working on this project, despite our interest in the topic and referring back to our lecture notes, we had a difficult time getting the project started initially as we lacked the experience working on video inputs, and knowledge when it comes to body/motion tracking. Therefore, I would say we should have done mini-projects with more different types of inputs, and also at least briefly touch upon the topic of body/motion tracking. I would have preferred it if there were lab-like class sections too instead of all classes being lectures only.

Project – Credits for Approach 1

For the first approach, it was Lucas who mainly worked on it so I might not be as knowledgeable as him on how he exactly used the references, but I will try my best to explain it as best as I can from my understanding and from the discussion I had with him.

- Main Reference: https://github.com/M-A-D-A-R-A/Sports_py
    - I believe this GitHub repository was used as the main reference for the first approach in structuring the code, differentiating the exercise pose estimation into different functions, and also figuring out how to process video input.
- Angle Calculation: https://stackoverflow.com/questions/58953047/issue-with-finding-angle-between-3-points-in-python
    - I believe this stack overflow thread was used to figure out the proper joint angle calculations using NumPy, with the specific three neighboring pose landmarks depending on the exercise type being used as the inputs for the function.

Project – Credits for Approach 2

For the second approach, I was the one who mainly worked on it, so I can give more in-depth explanations on how I used the references than I did for the first approach. I will try to make it as clear as I can to explain how I used these references to develop the second approach.

- Main Reference: https://github.com/pereldegla/yoga_assistant
    - This was the GitHub repository I used as the main reference when coming up with the idea of Machine Learning for the second approach. I used this to figure out how to extract pose landmarks data for training into a CSV file using pandas, and how training the prediction model using scikit-learn was done.

- Exercise Recognition Dataset:

  https://www.kaggle.com/datasets/muhannadtuameh/exercise-recognition

  - This was the large publicly available dataset on the internet that suited the needs of our second approach the best. It also gave me an idea of how the format of the CSV file for the training dataset of exercise pose recognition should look like. This dataset is a CSV file of pose landmarks data extracted from 500 videos of people doing exercises using MediaPipe Pose. So it was already in the exact format I needed without having to download large image and video files and having to go through the process of extracting the pose landmarks data myself. It also had the exercises classified into “up” and “down” statuses just like I needed. However, there was one big issue. The code I had at that point was using 4 extracted values per pose landmark which are xyz positions and the visibility value. Meanwhile, this dataset is missing the visibility values for all the pose landmarks and only has the xyz positions. Regardless, I modified my code accordingly to fit the format of this CSV dataset file and use just the xyz positions of the pose landmarks without the visibility values. The prediction model was trained and the code was running properly, but the recognizer could not detect the exercise pose changes at all regardless of all the optimization and changes I made at the time. My code probably needed a different ML implementation for training the prediction model to make this dataset work as my implementation relied quite a bit on the visibility values to account for the pose landmarks that are not fully visible within the webcam video frame. However, I lacked the ML knowledge to pinpoint the exact issue to fix, which led to me creating the image dataset myself.