



**KENNESAW STATE
UNIVERSITY**
COLLEGE OF COMPUTING AND
SOFTWARE ENGINEERING

Representation Learning for Motion Sequence Retrieval

IT Capstone Project Report
IT4983, Spring 2021
Kennesaw State University
May 2, 2021

Project Website: <https://sharr225.wixsite.com/team3/>
YouTube Link: <https://youtu.be/j1nGOV9XHE0>

Team Members

Ibrahim Mkadmi, Team Leader
Taylor Cuffie
Shondra Harris
Niko Mavridis
Andrew Mumford

Executive Summary

Human pose estimation is a growing field of technology that detects and analyzes human movements and poses. In combination with motion-sensing technology, our capstone project examines how human pose estimation can be used in the fitness industry to help users locate exercise videos online. A video of the user performing exercises is captured from the user's webcam. By implementing extensive Python coding, TensorFlow software (machine learning), and OpenPose/PoseNet software (pose estimation models), the captured video is used as real-time input. Pose estimation algorithms are calculated based on the captured video and several keypoints of the skeleton-based 2D and 3D models are defined as the output. The key points represent the nose, right ear, left leg, right foot, and other joints of a person's pose and are calculated from each video frame to estimate the person's pose. Our ultimate goal was to compare two pose estimation technology models--OpenPose and PoseNet--to determine which model is better. Our team defined criteria for comparing the technologies and performed code testing to analyze the results.

It is the desire of the project sponsor that the preferred technology will be used beyond the scope of this project to use the keypoints to query a database of existing professional trainer's exercise videos and locate a video that matches the input from the video. The retrieved video will be played for the user, thereby providing real-time feedback on the proper techniques for the demonstrated exercise.

Based on our research and code testing, we recommend the use of the OpenPose model with TensorFlow. The combination of these technologies will provide more accurate pose estimations.

Table of Contents

Background	4
Technical background	5
Project Outcomes and Achievements	6
Milestone outcomes	6
Milestone 1	6
Milestone 2	6
Milestone 3	7
Deliverable Outcomes	7
Overall Project Outcomes	7
Technical Summary	8
Project Planning and Management	16
Overview	16
Project Process	16
Team Collaboration Summary	17
Workload Summary	18
□ Milestone 1	18
□ Milestone 2	18
□ Milestone 3	18
Final Deliverable	19
Team reflection	19
Project Success Factors	19
Team Collaboration	20
Challenges	20
Areas to Improve	21
Appendix	21
Project Files List	21
Progress Reports List	22

Background

All team members are seniors at Kennesaw State University (KSU) and are enrolled in IT 4983. In this course, students work in teams to develop or implement a real-world IT solution integrating the knowledge gained from previous IT courses. The emphasis of the course is to help students develop and improve skills in the areas of technical design, research, documentation, project management, leadership, teamwork, and communication. The final IT solution developed in the course should address a typical business or organizational need. There are several teams formed in the course and each team has project sponsors who may be KSU faculty or business professionals. The project sponsors for our team are Dr. Ying Xie and Dr. Linh Le who are both KSU professors in the College of Computing and Software Engineering.

This capstone project is a part of a larger, major project on developing a smart workout assistant. The program will receive video input from a user's webcam and analyze the user's sequence. Algorithms are performed to produce a skeletal model that identifies the keypoint coordinates. Using the data captured from the webcam, the system will query an existing database of professional trainers' videos to locate a video with a matching sequence of motions. The retrieved video is played for the user, thereby providing real-time feedback on the proper techniques for the demonstrated exercise. The user can engage in a professional workout from the comforts of his/her own home while ensuring that proper and safe techniques are being used.

We developed 2 Python codes—one using PoseNet and the other using OpenPose to produce 2D skeletal models and extract the keypoint coordinates. The team encountered challenges producing a 3D, but with academic research of 3D PoseNet pose estimation and the knowledge gained from the 2D testing, we had enough data to compare the 2 technologies.

The video capture is executed from a command prompt or from a Python GUI, but this is not ideal for non-technical users. Future goals of the project include creating and improving a GUI that allows the user to interact with the program more easily and seamlessly.

Technical background

We were given several Github package options to use for our project. To begin the analysis and comparison, team members chose different packages and software applications to determine which was better for the project. TensorFlow, MatConvNet, and CNTK were used for the machine learning processes. We used PoseNet and OpenPose models to perform the pose estimations. In addition, OpenCV and CVUI software libraries were used for the GUI interface.

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, classify human actions in videos, track moving objects, and extract 3D models of objects. It has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS.

Source: <https://opencv.org/about/>

CVUI is a simple user interface library that uses OpenCV drawing primitives. OpenGL is not required.

TensorFlow and CNTK proved to be the most user-friendly package to use for the project. TensorFlow is an end-to-end open-source machine learning platform. It has a comprehensive, customizable, and flexible set of tools, libraries, and online resources that allow developers to easily build and deploy machine learning applications.

CNTK, the Microsoft Cognitive Toolkit, is a framework for deep learning. A Computational Network defines the function to be learned and once this network is appropriately developed, all the algorithm computations required for machine learning are completed automatically.

PoseNet is a machine learning algorithm that estimates human poses in real-time. PoseNet can estimate single or multiple poses, implying that there is a variant of the algorithm that can detect only one person in an image/video and another that can detect multiple people in an image/video. PoseNet is designed to run on low-power computers like a browser or a cell phone.

OpenPose is one of the most widely used open-source pose prediction technologies (Cao et al., 2018), and biomechanics researchers find it simple to use. pass images (or frames) as NumPy matrices. Then, we get the key-points (pose-positions) for that image as a NumPy matrix.
<https://medium.com/pixel-wise/real-time-pose-estimation-in-webcam-using-openpose-python-2-3-opencv-91af0372c31c>

We encountered issues trying to display the pose estimation within the OpenCV GUI. We had to research, locate, and edit code that would allow us to capture the video in the GUI with a reasonable frame rate. Because of the issues we encountered in the OpenCV GUI, we decided to research other GUI options. In doing so, we discovered CVUI which is simpler and does not require OpenGL.

Project Outcomes and Achievements

Our team was able to achieve the goals of Milestones 1 and 2 and a portion of Milestone 3. All team members were able to capture webcam video, produce a 2-D skeletal model, and extract the 2-D keypoint coordinates. We gained valuable experience exploring Python files within the Github projects. The keypoint coordinates were compiled using classes from the Estimator and Common python files. We also learned about the NumPy library. By using the NumPy Python library, we were able to create the 2-D array which yielded the skeleton on the black background.

3D coordinates were extracted using the OpenPose model. However, team members encountered difficulty finding sample code to extract the 3D keypoint coordinates using the PoseNet model. However, performing extensive research, we were able to locate a link to a web browser version of PoseNet. From our research and the link, we performed enough testing and gathered enough

information to make a comparison of the two technologies. We were able to provide the project sponsor with customized Python source code and recommendations that can be used in the larger scope of the project.

Milestone outcomes

Milestone 1

The goal for this milestone was to get an environment set up for as many team members as we could. Our objective was to have at least 3 different environments so that we could compare the technologies and determine which yielded more accurate results thereby, performing a more precise query of existing videos in the exercise video database.

While each environment was slightly different, the recurring platforms among all were OpenCV, Tensorflow, and Python.

In addition, each team member performed different exercises to determine if the software captured data for certain types of exercises better. We learned that floor exercises were less accurate because Ideally, the entire body should be visible. Sitting in a chair or on-the-floor exercises may not produce the best estimation since the entire body is not visible. Finally, we learned that the pose estimation is more accurate when the human is an appropriate distance from the webcam.

Milestone 2

The goal for this milestone was to extract the human pose skeleton from the webcam feed of the user's exercise routine. After the extraction, the coordinates of the keypoints (body parts) are displayed in a side-by-side view with the video from the webcam. The key points mentioned above represent the nose, right ear, left leg, right foot, and other joints of a person's pose are calculated from each frame in real-time. We also learned about the NumPy library. By using the NumPy Python library, we were able to create the 2-D array which yielded the skeleton on the black background.

Milestone 3

The goal of this milestone was to compare OpenPose and PoseNet software and their effectiveness of extracting 3D keypoint coordinates. There were coding samples, resources, and documentation for OpenPose, Using PyQtGraphs and Open GL we created a 3D pose estimation in real-time. However, we encountered challenges locating code for extracting the PoseNet 3D coordinates.

To leverage this, we performed extensive research to learn more about how PoseNet works and utilize code testing from other programmers in order to make a comparison of the two. We were unable to get PoseNet working in our own environments through the original GitHub packages, but we did find a link to a web browser version of PoseNet.

Although PoseNet would be easier for users since it can run on low-power devices such as mobile phones, it was not the ideal pose estimation model. There were numerous inaccuracies in the data because of missing poses in video capture. Therefore, we concluded that OpenPose was the better model until significant improvements are made with the OpenPose model.

Deliverable Outcomes

We met regularly via Zoom and had daily communication via the GroupMe app. Team members shared their findings and assisted others with debugging code. Gantt Charts were maintained to monitor the progress towards completing each milestone. A report was provided to the project sponsor documenting our progress. Milestone 3 was completed, but with partial code development.

Source code for capturing webcam video in real-time and extracting 2D and 3D data points in real-time was developed. We created a website to share our project and provide information about our team members.

Overall Project Outcomes

The main goal of this project was to make recommendations for tools, provide customized coding, and offer recommendations for software applications to use for the project. Customized Python code was developed for live webcam pose estimation and 2D and 3D keypoint extraction. The results of our project determined that TensorFlow is the better machine learning software, CVUI is the better library, and OpenPose is the better pose estimation software.

Criteria	PoseNet	OpenPose 3D
Ease of Code Editing	No team member was able to consistently get PoseNet to work in their environment; few or no examples available	Multiple team members were able to get Open Pose working in their environments in varying degrees; several examples available
Processing Power	PoseNet needs very little processing power to work smoothly.	Open Pose takes up a considerable amount of processing power to work smoothly.
Pose Estimation Accuracy	PoseNet has difficulty distinguishing between people and inanimate objects	Open Pose is extremely accurate with keypoint detection
Output Frame Rate	Captured video had an exceptional frame rate	Captured video had a degraded exceptional frame rate

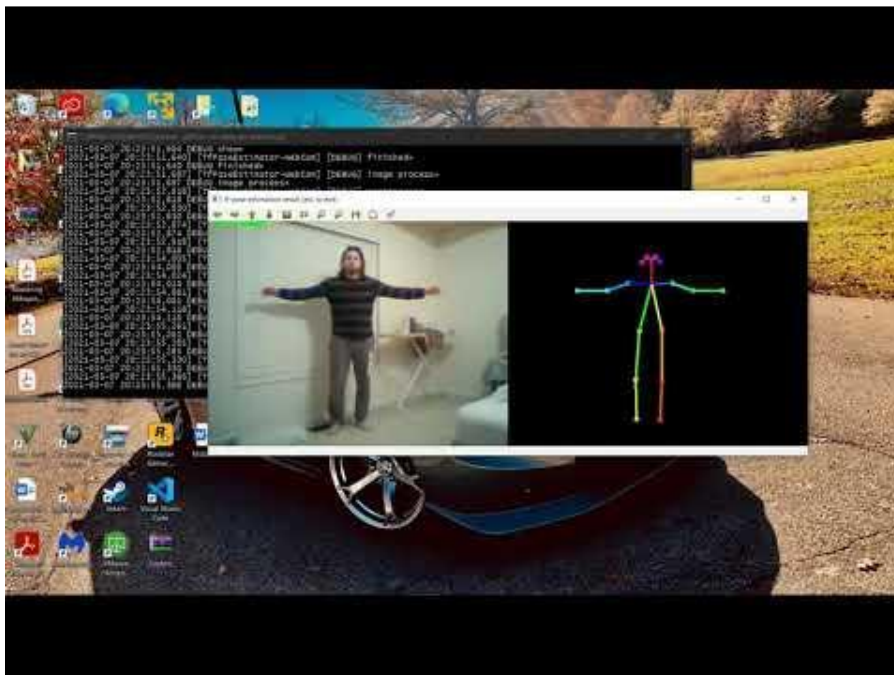
Criteria	CNTK	Tensorflow
Ease of Code Editing	More difficult to learn	Easier to learn; Contains a huge library of functions
Available resources	Microsoft based-product with limited Microsoft support	Numerous reputable developer communities with sample code, scholarly papers

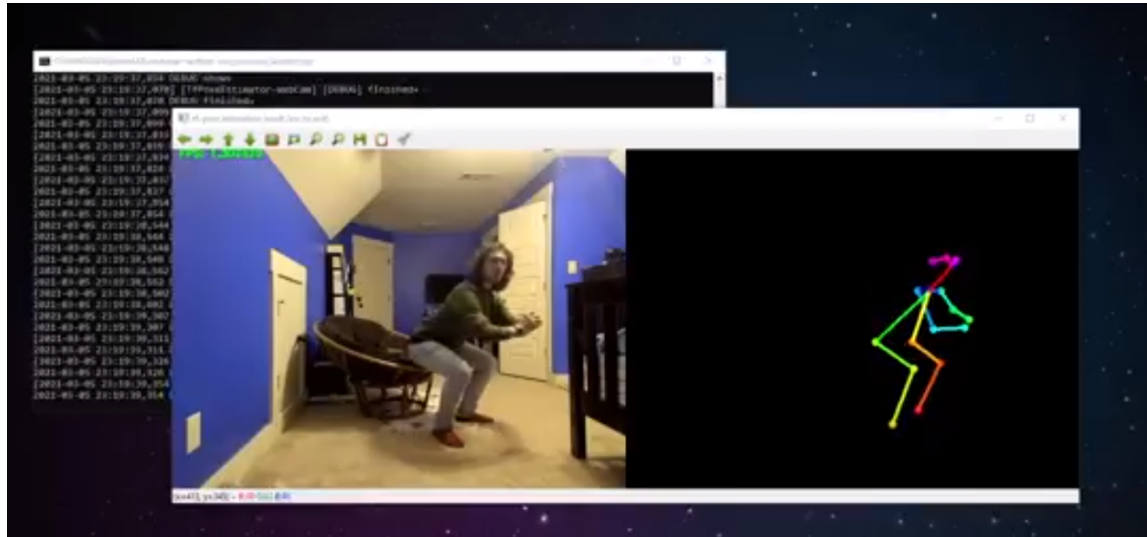
Technical Summary

Each team member completed the tasks individually for Milestones 1 and 2. This allowed us to compare the results for various exercise types and also allow each member to use different packages and software.

Video Capture

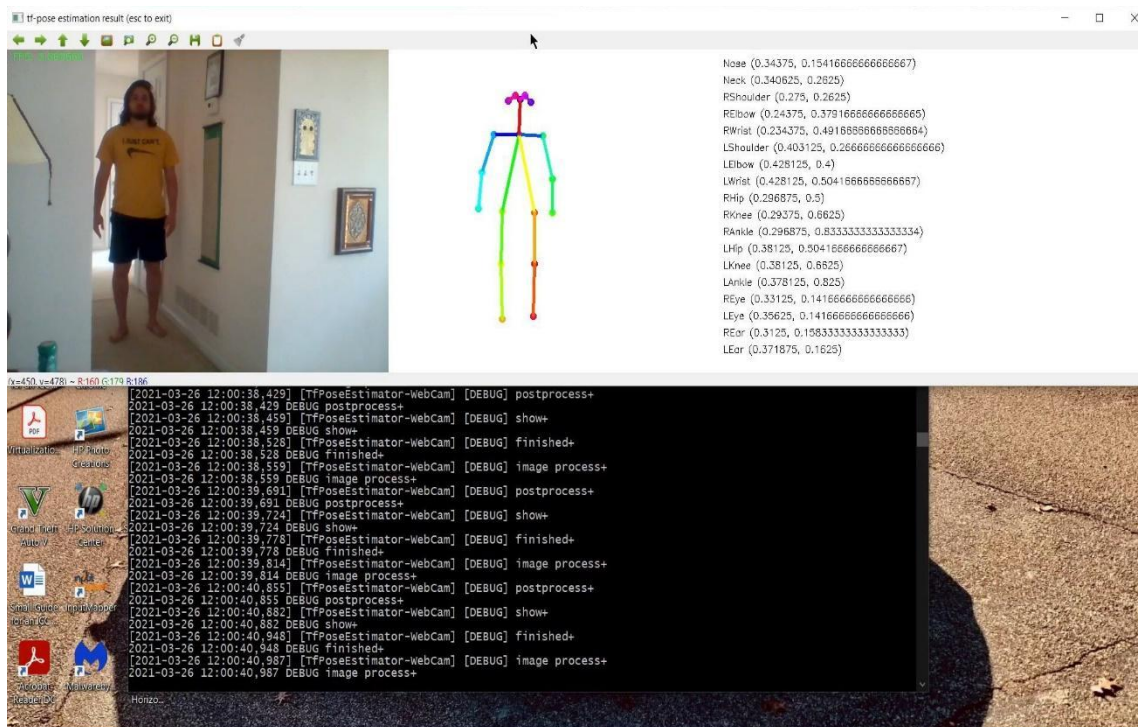
After editing the Python code in a programming environment of choice (Jupyter or Visual Studio), team members used their webcams to capture videos of them performing various exercise routines. The video is captured as individual frames.

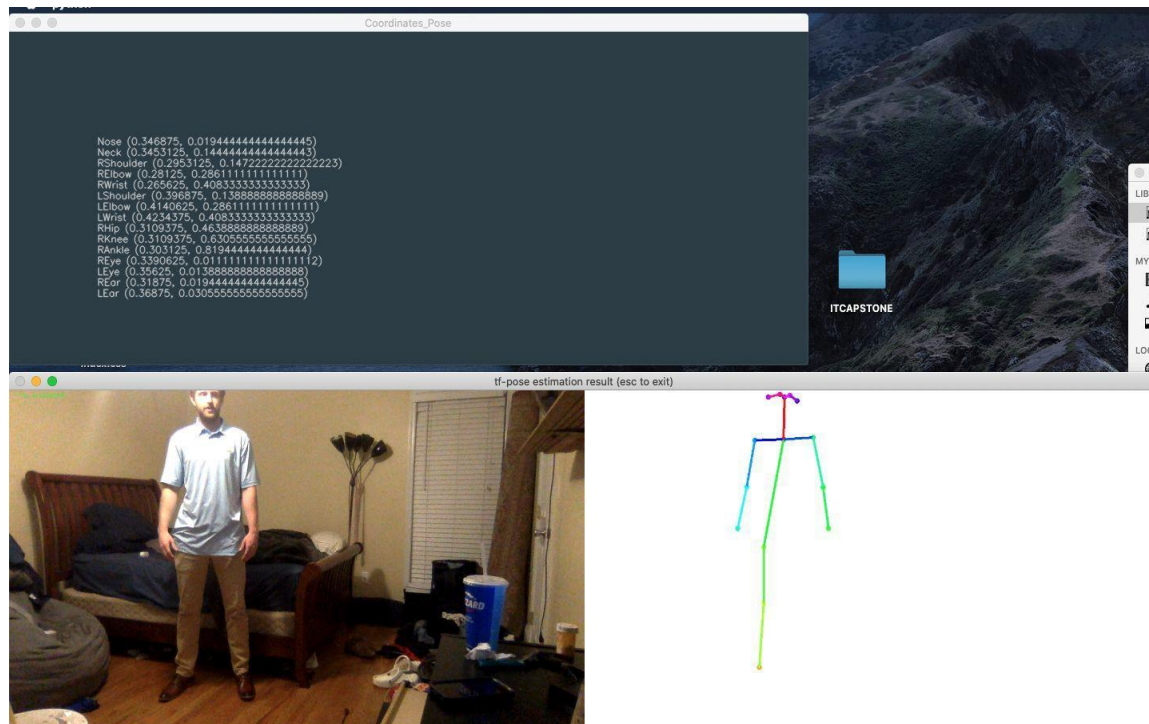




2D Data Extraction

After successfully capturing the video and converting it into usable data, code was edited to successfully extract the human pose skeleton in real-time. We simultaneously extracted the 2D key point coordinates in real-time and displayed the skeleton model and the data points in a side-by-side view.





3D Data Extractions - OpenPose

Using PyQtGraphs and Open GL we can create a 3D pose estimation in real-time. We started by creating our class with five methods: Init, Mesh, Update, Start, and Animation. **Init** will have the creation of the window and initializing all the plot and graph objects. **Mesh** will have all our 3D plots and return all the keypoints. **Update** method will update all our plot objects.

```
This serve as our base OpenGL class.
"""

import numpy as np
import pyqtgraph.opengl as gl
import pyqtgraph as pg
from pyqtgraph.Qt import QtCore, QtGui
import sys

import cv2
import time
import os

from estimator import TfPoseEstimator
from networks import get_graph_path, model_wh
from lifting.prob_model import Prob3dPose
```

```
class Terrain(object):

    def __init__(self):

    def mesh(self, height=2.5):

    def update(self):

    def start(self):

    def animation(self, framerate=10):
```

Creating Objects for an image and Keypoints

```
model = 'mobilenet_thin_432x368'
camera = 0
w, h = model_wh(model)
self.e = TfPoseEstimator(get_graph_path(model), target_size=(w, h))
self.cam = cv2.VideoCapture(camera)
ret_val, image = self.cam.read()
self.poseLifting = Prob3dPose('./src/lifting/models/prob_model_params.mat')
keypoints = self.mesh(image)

self.points = gl.GLScatterPlotItem(
    pos=keypoints,
    color=pg.glColor((0, 255, 0)),
    size=15
)
self.window.addItem(self.points)
```

```
humans = e.inference(image, scales=[None])

for human in humans:
    pose_2d_mpii, visibility = common.MPIIPart.from_coco(human)
    pose_2d_mpiis.append(
        [(int(x * width + 0.5), int(y * height + 0.5)) for x, y in pose_2d_mpii]
    )
    visibilities.append(visibility)

pose_2d_mpiis = np.array(pose_2d_mpiis)
visibilities = np.array(visibilities)
transformed_pose2d, weights = self.poseLifting.transform_joints(pose_2d_mpiis, visibilities)
pose_3d = self.poseLifting.compute_3d(transformed_pose2d, weights)

return keypoints
```

This code snippet gets the keypoints and passes an image. Some exception handling is added or an assertion error is displayed if a person is not detected. It will return “ body not in image”. Otherwise, it will update keypoints.

```
def update(self):  
    """  
    update the mesh and shift the noise each time  
    """  
    ret_val, image = self.cam.read()  
    try:  
        keypoints = self.mesh(image)  
    except AssertionError:  
        print('body not in image')  
    else:  
        self.points.setData(pos=keypoints)
```

Using the function **plot_pose**. We save our plot points in a folder called “Lifting” and a file called “.draw”. This is responsible for passing our keypoints and plotting them in 3D.

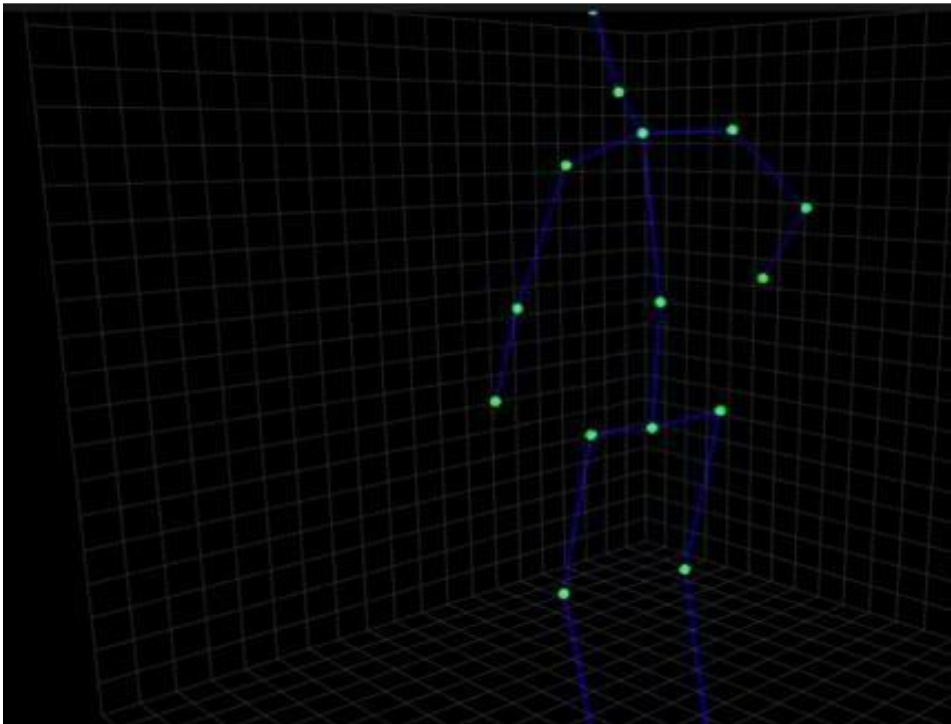
```
def plot_pose(pose):  
    """Plot the 3D pose showing the joint connections."""  
    import mpl_toolkits.mplot3d.axes3d as p3  
  
    _CONNECTION = [  
        [0, 1], [1, 2], [2, 3], [0, 4], [4, 5], [5, 6], [0, 7], [7, 8],  
        [8, 9], [9, 10], [8, 11], [11, 12], [12, 13], [8, 14], [14, 15],  
        [15, 16]]
```

With our points plotted into PyQtGraph with OpenGL, we created lines that connected each joint. We created a *for* loop for this. We enumerated self.connection and add a GL line plot item into our

dictionary. For each line, we added an entry into our dictionary. A NumPy array created two coordinates with every point obtained. The color was specified as blue, and the width of these lines was 3. Everything can be added to our window. [self.lines]

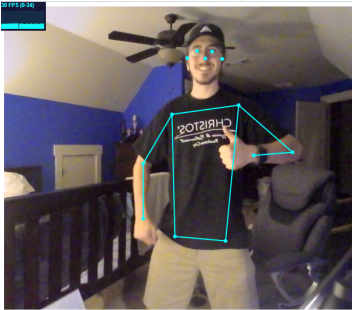
```
self.points = gl.GLScatterPlotItem(  
    pos=keypoints,  
    color=pg.glColor((0, 255, 0)),  
    size=15  
)  
self.window.addItem(self.points)  
  
for n, pts in enumerate(self.connection):  
    self.lines[n] = gl.GLLinePlotItem(  
        pos=np.array([keypoints[p] for p in pts]),  
        color=pg.glColor((0, 0, 255)),  
        width=3,  
        antialias=True  
    )  
    self.window.addItem(self.lines[n])
```


3D Plot Result



3D Data Extractions- PoseNet

We were unable to get PoseNet working in our own environments through the original GitHub packages and instead found a link to a web browser version of PoseNet. PoseNet uses code to capture a 2D representation of a figure and then estimates the 3D position of the figure using machine learning.



PoseNet GUI Backend

Algorithm: multi-pose

Input

Architecture: MobileNetV1

InputResolution: 500

OutputStride: 16

InputScale: 0.75

NumBodies: 2

Single Pose Detection

Multi Pose Detection

MaxPoseDetections: 5

MinPoseConfidence: 0.15

MinPartConfidence: 0.1

MinRadius: 30

Output

ShowVideo: ☒

ShowSkeleton: ☒

ShowPoints: ☒

ShowBoundingBoxes: ☐

Close Controls

PoseNet runs with either a **single-pose** or **multi-pose** detection algorithm. The single-person pose detector is faster and more accurate but requires only one subject present in the image.

The **output stride** and **input resolution** have the largest effects on accuracy/speed. A higher output stride results in lower accuracy but higher speed. A higher image scale factor results in higher accuracy but lower speed.

```
[
  {
    pose: {
      keypoints: [{position:{x,y}, score, part}, ...],
      leftAngle:{x, y, confidence},
      leftEar:{x, y, confidence},
      leftElbow:{x, y, confidence},
      ...
    }
  },
  {
    pose: {
      keypoints: [{position:{x,y}, score, part}, ...],
      leftAngle:{x, y, confidence},
      leftEar:{x, y, confidence},
      leftElbow:{x, y, confidence},
      ...
    }
  }
]
```

Project Planning and Management

Overview

Our sponsors separated this project into three milestones, which gave us the room to break down the milestone goals to fit our needs. The information for each milestone was presented at the previous milestone review meetings with the sponsors on Microsoft Teams.

Along the way, our team leader set up separate meetings to check up with our sponsor on the progress we have made and any questions we had.

Our group mainly chatted through text via GroupMe with weekly video meetings over Zoom.

We split up responsibilities based on our strengths, some of us were better at analyzing code, some were good with technical writing, and some floated giving help where they saw fit. Every team member reviewed each other's work and gave insight to improve the work done. This method worked out very well for us, in some cases those who were not strong in coding gave great advice to those who were.

Every week we needed to:

- Meet over video on Monday at 12:00 PM
- Provide a weekly update on the project
- Update the Gantt Chart

Project Process

Our sponsors gave us three goals to achieve. Install a pose estimation platform, extract the 3D coordinate points, and compare to another pose estimation platform were the three tasks that were divided into milestones we were tasked to complete. We chose to attack these same goals individually at first and return with what we found. This process helped us make solutions to errors we ran into, and with our frequent GroupMe conversations we were able to update each other on what progress we made.

Milestone 1

- Research different pose estimation repos “Complete”
- Install Tensorflow “Complete”
- Perform System Testing “Complete”
- Record and Upload Exercise Videos “Complete”
- Compile Video and Pose Extraction Results “Complete”
- Write Milestone 1 Report “Complete”

Milestone 2

- Edit the existing Open Pose 3D Code “Complete”
- Extract Skeleton Coordinates “Complete”
- Compile Video and Pose Extraction Results “Complete”
- Write Milestone 2 Report “Complete”

Milestone 3

- Research PoseNet and 3D Points “Complete”
- Compare Open Pose 3D and PoseNet “Complete”
- Write Milestone 3 Report “Complete”

Team Collaboration Summary

- **Ibrahim Mkadmi** - Our team leader who set up all of our meetings and was our main contact with the project sponsors. He worked most extensively on Open Pose 3D installation and troubleshooting. He also compiled the weekly summary and Gantt chart and turned it in when necessary. Recorded Open Pose footage.
- **Taylor Cuffie** - Was integral to getting our Open Pose 3D looking the way it was supposed to. Solved our issue of getting the live camera feed and separated pose extraction skeleton side by side. Recorded Open Pose footage.

- **Andrew Mumford** - Worked to understand the Open Pose 3D files and explain how it worked to the rest of us. Worked on GUI for Open Pose. Recorded Open Pose footage.
- **Shondra Harris** - Set up the team website and organized any documents needed to be turned in. Worked on installing PoseNet. Recorded Open Pose Footage.
- **Nikolas Mavridis** - Worked on PoseNet results. Attended sponsor meetings with Ibrahim. Recorded PoseNet and Open Pose footage.

Workload Summary

This section outlines how long each milestone took and summarizes the overall goal in each milestone.

- **Milestone 1:** Goal was to choose between Tensorflow, Pytorch, Caffe2, Chainer, MXnet, MatConvnet, and CNTK to install correctly and use. This took a total of 166 man hours.
- **Milestone 2:** The goal in this milestone was to have two windows, one to have a live camera feed and one next to it to show the skeleton over a black background then extract the live coordinates of the skeletons. This took a total of 52 man hours.
- **Milestone 3:** For this milestone we needed to research another pose estimation software and we chose PoseNet. We could not get it properly installed in our environments so we used a web browser version to compare it to Tensorflow. In addition to this we also kept trying to make our Tensorflow software better. This took a total of 166 man hours.

Final Deliverable

The final deliverable will be a zip folder containing all of our weekly logs, every milestone report, every Tensorflow file and this final report. After Milestone 3 we spent time creating our milestone 3 report, recording the final presentation, submitting a C-Day application and a companion project website. This took 65 man hours. The total time spent on this capstone project ended up taking 384 hours.

Team Reflection

Project Success Factors

1) Collaboration: Our team worked well together, and everyone was willing to assist where they were needed. Everyone stayed up to date on team communications, and participated in the team's chat regularly. Meetings between the professor and the groups were also beneficial. This afforded us the opportunity to ask questions of the professor and gain insights from other groups. Each member was aware of our team's progress, and was ready to call a meeting if necessary. All members were willing to share any struggles they were having.

2) Organization: Planning the project from the beginning clarified our understanding of the project's scope. Our team lead organized the team well, and ensured that we were aware of deadlines.

3) Research: Our team learned some relevant information throughout other courses. However, a large part of our success relied on effective research. Without taking out the time to conduct this research and apply it effectively this project would have been vastly more difficult.

Team Collaboration

Our group held regular meetings to address issues, and to assist one another. These meetings were arranged primarily via our group chat, and set at the most convenient time for all members. These meetings were primarily via Zoom. Periodically, our team lead and perhaps another group member met with our project sponsor. These meetings were primarily via Microsoft Teams. The meeting applications allowed participants to share their screen. This was a vital helpful feature.

In order to complete documents our team utilized a variety of collaboration tools. These included Google Docs, Google Slides, and Github. These enabled us to collaborate in real time.

Challenges

This sort of project was new to most of the team. Downloading and installing packages from Github was our first hurdle. This meant utilizing the command line, and required much research. It was often difficult to pinpoint what issues were taking place. We overcame this hurdle by taking out the time to help one another.

Once the packages were installed, the next issue was how to manipulate the program in order to achieve further milestones. We overcame this struggle by being willing to share any resources that we found, and by adding to one another's work. Our sponsor was also helpful in this regard, and clarified our goals.

Additional challenges of this project involved implementing PoseNet. Our experience implementing OpenPose provided some insight into the needed coding. However, with few examples available, there were still program nuisances that we had to overcome.

Another challenge involved deciding on the criteria to use for comparing the two technologies. The frame rate of the output decreased with both technologies. Open Pose utilizes and depends on advanced GPU and CPU to get a better frame rate.

Pose Net frame rate decreased dramatically. In addition, it was not accurate and targeted non human objects.

Areas to Improve

Future improvements would naturally include implementing Pose Net more fully. We could also improve our GUI so that it is more user-friendly and seamless for the user.

Another improvement could involve capturing a pose sequence. This is one way that we could improve the utility of our program and get the most out of pose estimation technologies.

Appendix

Project Files List

tf-pose-estimation

- `_init_.py`
- `cvui.py`
- `cvui2.py`
- `run.py`
- `run_checkpoint.py`
- `run_directory.py`
- `run_image.py`
- `run_video.py`
- `run_webcam.py`
- `run_webcam_skeleton.py`
- `setup.py`

Progress Reports List

Project Plan

- Project Plan --Project 3.docx

Three Milestone Reports

- Milestone #1 Report
- Milestone #2 Report
- Milestone #3 Report

Final Gantt Chart

- Project Schedule and Tasks Planning -- Project 3

Last Activity Log

- Complete Activity Log

Final Report

- FinalCapstoneReport