Squat Form Analysis - Using Computer Vision

David Soyele
Mr. Andrew Meehan
Dr. Bryan Hennelly
Electronic Engineering Department,
Maynooth University

david.soyele.2020@mumail.ie
andrew.meehan@mu.ie
bryanh@cs.nuim.ie

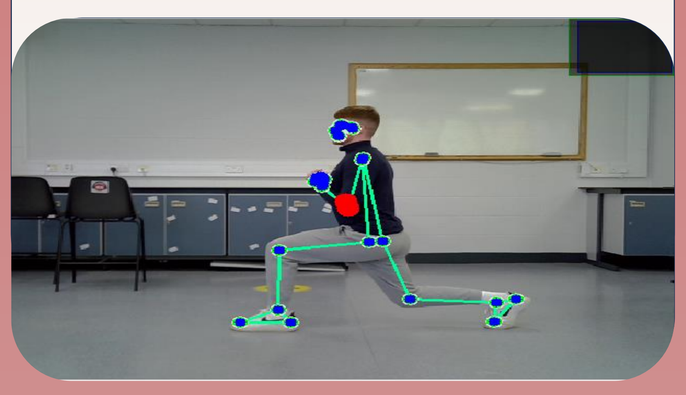
The Aim

The aim of this research project was to program a set of existing cameras, to be used for body detection with assistance from OpenCV. This would then lead to the use of this body detection in exercise, specifically squatting.
We were given a few tools to help facilitate this project:

- The Jetson Nano(a small powerful computer)
- The ArduCam Quad Camera Kit
- A Raspberry Pi

We ended up tackling this aim in a few different ways during the 6 weeks:

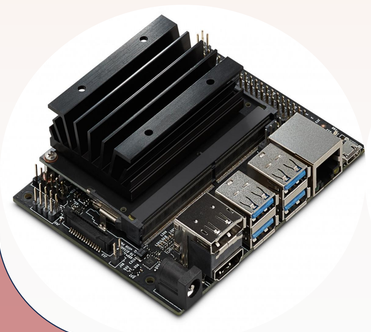
- We were initially working using the Jetson Nano and the ArduCam Quad Camera kit. This proved suitable at the beginning, but we quickly realised using two external web cameras instead would be more efficient. As such, we started programming, using an IDE called Pycharm and OpenCV.
- OpenCV being a python library centred on computer vision.
- Furthermore, we began research into camera calibration, an important tool for cameras to be able to take pictures seamlessly and efficiently. This is particularly important when it comes to computer vision as it makes sure that all cameras running off the code are consistently seeing the same image.
- In the end, we had a very efficient program running able to accurately track human body movements, as I will detail throughout the poster.

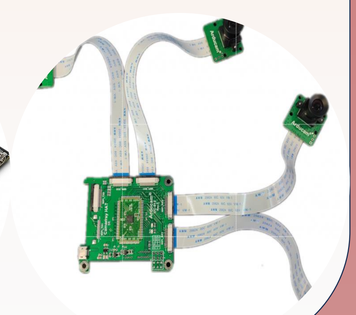


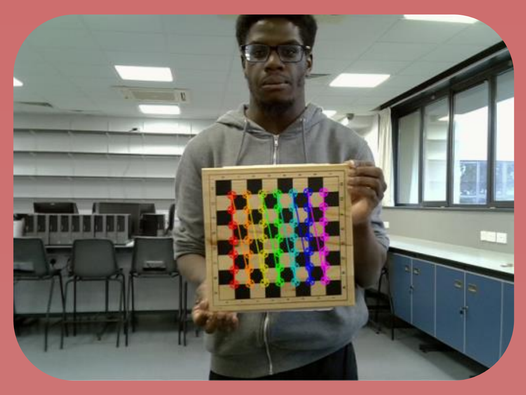
Section 1: The Jetson Nano and the ArduCam 16MP IMX519 Autofocus Synchronized Quad Camera Board :

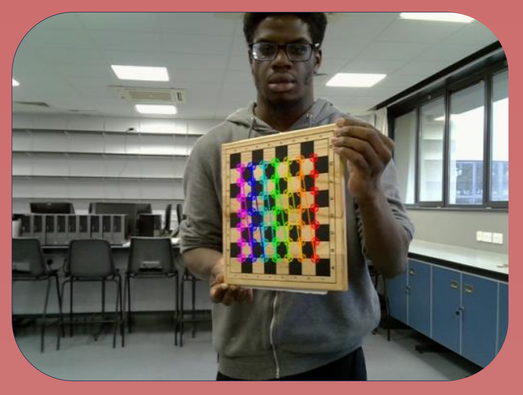
The Jetson Nano is a small computer that is quite powerful for it's size. It is mostly used for embedded applications, but in this case I used it with the ArduCam board. The Jetson Nano has it's own embedded Ubuntu type IDE built into it. Ubuntu being an operating system on Linux So there is no need to download Ubuntu onto a personal laptop or something like that. We ended up having to do a lot of research into Ubuntu for this project, especially with learning certain commands, as well as how the OS(operating system) was structured.
The Jetson website has a guide that goes through the step by step process of getting the Jetson Nano booted up and working, this is what I used in the beginning to get a handle on the machine.

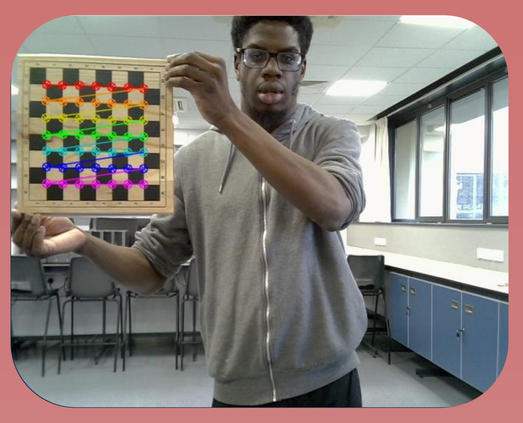
The ArduCam Quad Camera Kit is a circuit board comprised of 4 small external cameras attached to the board with ribbon cables. It theoretically allows for a complete 360 degree of an object, by having each camera at a different point around it.











Camera Calibration

In the pictures above, you can see me in the process of doing something called Camera Calibration. The main central piece for calibrating a camera is some kind of consistent pattern. To facilitate this a chessboard is the most often used pattern, for several reasons:

A. They are easy to construct due to their simple shape.
B. Their planar grid structure means they define several natural interest point(essentially what the human eye naturally falls on) in an image.
C. Simple lines are another major reason for the use of chessboards. The grid structure defines multiple sets of parallel lines meaning that line detection algorithms(or any detection algorithms) can easily detect them.

Calibration is extremely important, especially in sophisticated computer vision project and the like. Most cameras do not have any inherent way to map pixels in images to physical geometry in the real world. This can result in code breaking, or failing due to discrepancies in what different cameras could be seeing. Hence why a consistent calibration process is so important.

Section 2: PyCharm and OpenCV implementation with 2 webcams

PyCharm ended up being the python IDE we used during this project. This was mostly due to its convenience as we could install dependencies directly through the IDE, rather than through the Ubuntu terminal.
We ended up going through several versions of the code during this project, changing due to new things we discovered during the 6 weeks and other such complications.

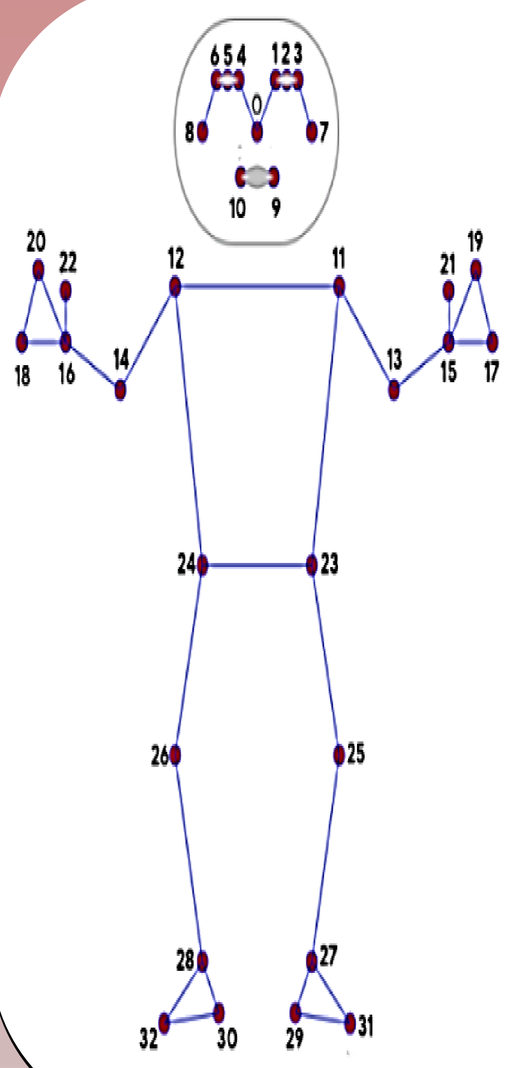
- The original form of the code consisted of a simple two webcam camera picture taking setup. The cameras would periodically take pictures with the press of a button, and save them in an external file
- We then implemented OpenCV into the code. Now the code would overlay a human pose estimation model(using mediapipe) over anyone in the camera's lens. We were then able to code some calculations into a function that would calculate the angles of certain parts of the body(arm to forearm, legs to hip etc.) for use in analysis
- Finally, code was implemented that would store the angle values and such into a .txt file format. This would then be converted into a .csv file which would allow the values to be viewed on excel.
- There is absolutely more room to grow with the python code. We considered perhaps even using machine learning to train an AI to recognize when someone was performing a squat. However with the time we had, I think we achieved more than enough.

Section 3: Theory

We encountered a lot of theory and recurring topics during my research over the 6 weeks.

- For example, one topic that came up and actually proved to be an important part of the programming was confidence intervals. This refers to a range of different estimates for a particular thing or parameter you are cooking for.
- This topic is used extensively in statistics and other fields, but we managed to find a use for it concerning the two webcam cameras. We were using two cameras for our purposes, but we had to base our angle calculation off only one of them.
- To rectify this we implemented a confidence value calculation in the code. Essentially, whenever either webcam looked at a certain target they would calculate how much of the target they could see. And whichever one felt like they had the more accurate value(99% vs 98.89% for example) the angle calculations would be based off that cameras view.

We also found some useful theory regarding human body position tracking itself.
The figure on the right is the chart that most computer vision algorithms follow when it comes to human body position. Because computers only read numbers, each human body part is represented as a number as opposed to their name. This lets the algorithm track the body more efficiently and it also helped me write better code. If I wanted to find the angle between the hip and the ankle for example, I could use the numbers 24 and 28 as points for my calculation.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Conclusions:

Considering the amount of time we had to conduct this research project, I think we made an excellent amount of progress. In hindsight, I don't think the task was too difficult to manage, but I did end up being very creatively challenged by the project. We did a demonstration in front of Andrew Meehan on the second last day of the project, and he seemed to really like how much progress we had made.
In the end the two programmed webcams were able to accurate detect someone's body, overlay a human pose estimation model over it, and calculate various angle pertaining to the different parts of the body. It would then print these results onto an excel sheet for later analysis. We did have some other ideas however. Given more time, We had plans to implement machine learning in some way, shape or form. This would add a level of automation to the project, something it currently lacks. Essentially, an AI would be trained on several images and videos of people squatting at different angles. This would include people intentionally squatting incorrectly, to give the algorithm a larger dataset. This would result (hopefully) in a neural network capable of accurately recording a squat being performed, as well as be able to tell when it was completed. Overall, I think the project was a success, and I would not mind pursuing it further in the future.

Acknowledgements:

Special thanks to Dr. Bryan Hennelly and Andrew Meehan for guiding and co-ordinating this project. Also special thanks to Sam McCormack and Jake O'Brien, the two other SPUR candidates that did this project with me. Finally, thank you to Maynooth University for giving me this opportunity in the first place.

References:

Code with Aarohi: https://www.youtube.com/watch?v=P-EZr0zy53g
JetsonHacksNano: https://github.com/JetsonHacksNano
Nicolai Nielsen: Camera Calibration in less than 5 Minutes with OpenCV
Nicholas Renotte: AI Learns to Do Deadlifts
OpenCV Scale Resolution: https://www.codingforentrepreneurs.com/blog/open-cv-python-change-video-resolution-or-scale/

Ollscoil Mhá Nuad
Ollscoil na hÉireann Má Nuad

MU SUMMER PROGRAMME FOR UNDERGRADUATE RESEARCH SPUR