TALLINN UNIVERSITY OF TECHNOLOGY

MACHINE VISION FINAL PROJECT

Gym workout monitoring method using webcam.

Nikola Tomažin 204454IV, Bruno Marović 204444IV

Tallinn, January 2021.

1. Introduction

For the final project of the course Machine Vision we were supposed to make a *Gym workout monitoring method using a webcam*. In the last couple of years people have become more and more aware about the negative effects our mostly inactive lifestyle causes, more people are starting to go to the gym to workout. To avoid any injuries during the workouts, people often resort to hiring a personal trainer or asking a friend to check out their posture during the workouts to see if they are doing it correctly. However, people sometimes don't have a possibility to do that, be it because of money issues or something else. In that case something like a monitoring system would be of great assistance.

Speaking of AI fitness coach apps[1], the common flow looks as follows:

- Capture user's movements while doing an exercise
- Analyze the correctness of an exercise performance
- Display mistakes to the user interface

Human pose estimation is a computer vision-based technology that detects and analyzes human posture. The main component of human pose estimation is the modeling of the human body. There are three of the most used types of human body models:

- skeleton-based
- contour-based
- volume-based

We split the main problem into few smaller problems. The first one was to detect the joints of the human body and generate a "skeleton", second was to adapt the program to detect different exercises and the third one was to optimise the code so it could be run in real time, but unfortunately we didn't manage to optimise it enough to run it smoothly enough to be usable.

[1] https://mobidev.biz/blog/human-pose-estimation-ai-personal-fitness-coach

2. Human joint detection and skeleton generating

We used the skeleton-based model which consists of a set of joints (keypoints) like ankles, knees, shoulders, elbows, wrists and limb orientations comprising the skeletal structure of a human body.

The process of pose estimation usually involves the extraction of joints on a human body, and then analysis of a human pose by deep learning algorithms. If the human pose estimation system uses video records as a data source, keypoints (joints locations) are detected from a sequence of frames, not a single picture. It allows us to achieve higher accuracy as the system analyzes an actual movement of a person, not a steady position.

There are several ways to develop the 3D human pose estimation system for fitness. The optimal approach is to train a deep learning model to extract 3D or 2D key points from the given images/frames.

Of course, using video streams from several cameras with different views on the same person doing exercises – it will grant us better accuracy. But multi-cameras are often not available. Also, analyzing video from several video streams will take more computer power to process.

2.1. Dataset and the architecture

For the Pose Estimation model two datasets were used COCO 2018 Keypoint Detection Task[1] and MPII Human Pose Dataset[2]. The model used in this tutorial is based on a paper titled Multi-Person Pose Estimation[3] by the Perceptual Computing Lab at Carnegie Mellon University.

[1] https://cocodataset.org/keypoints-2018

[2] http://human-pose.mpi-inf.mpg.de/

[3] https://arxiv.org/pdf/1611.08050.pdf

2.2. Resulting model

The model which gave us the best result was the model trained on the MPII dataset. Results can be seen in 2.1 and 2.2.

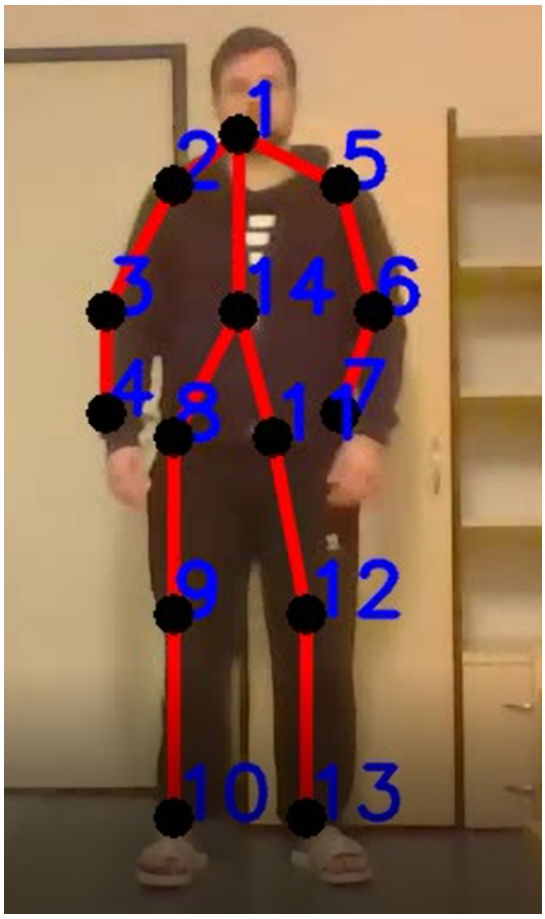

Figure 2.1: Nikola with the detected joints and built skeleton

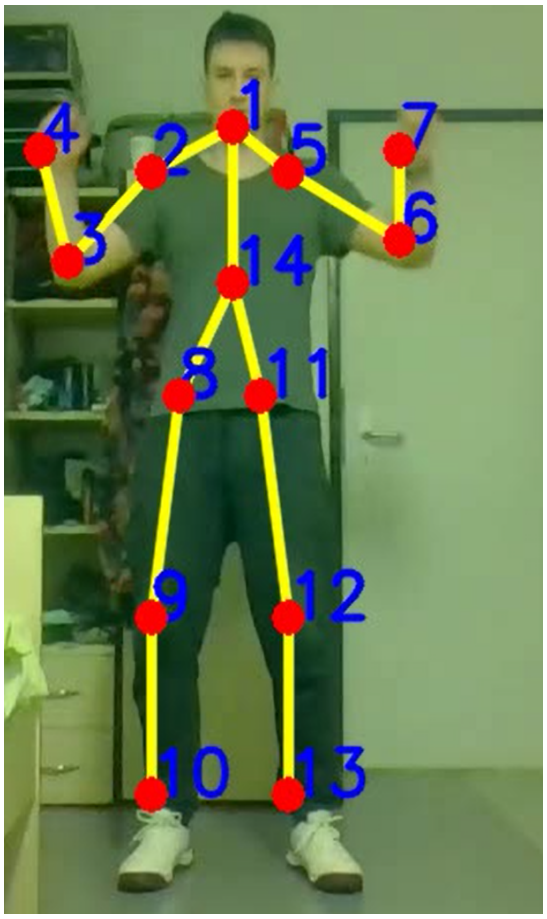

Figure 2.2: Bruno with the detected joints and built skeleton

The points are as follows: MPII Output Format Neck – 1, Right Shoulder – 2, Right Elbow – 3, Right Wrist – 4, Left Shoulder – 5, Left Elbow – 6, Left Wrist – 7, Right Hip – 8, Right Knee – 9, Right Ankle – 10, Left Hip – 11, Left Knee – 12, Left Ankle – 13, Chest – 14.

3. Detection of different exercises

After we got a model who could detect human joints we switched to adapting that data to some real life problems, like exercise detection. We decided for a small number of simple exercises which are easily done in the home environment like squatting, biceps exercises and kettle bell exercises. For the monitoring to begin the user needs to find himself in the region of interest which is predefined for each exercise because for squatting we need to see the whole body, while for biceps exercises we need to see only the upper body.

We planned to generate a dataset of properly and not properly performed exercises to make a model which would be able to detect when an exercise is done correctly, but the model gave poor results and since we didn't have much time we decided for a different approach - manually defined rules of performing the exercise.

3.1. Squatting exercise

For the squatting exercise the rules were that your feet cannot be narrower than the shoulders, that knees shouldn't go to the inside or to much to the outside, and for a squat to be valid you have to go down enough. The initial and final squatting positions can be seen in 3.3 and 3.4, while incorrect squatting position can be seen in 3.5 and 3.6.

3.2. Biceps exercise

For the biceps exercise the rules were simpler, we just had to detect if the biceps is done with a bar or with fitness dumbbells and are both hands work simultaneously or separately. The correct biceps movement would be when the straight hand (palm) reaches the shoulder.

Figure 3.1: Initial biceps position

Figure 3.2: Final biceps position

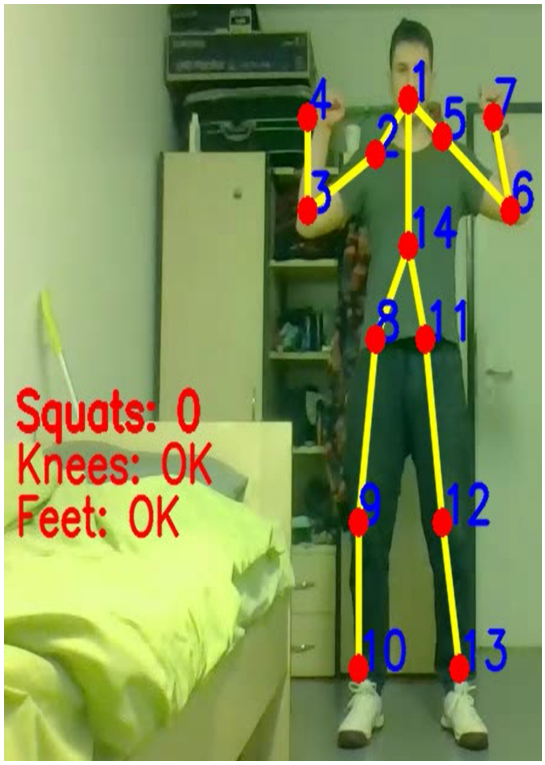

Figure 3.3: Initial squat position

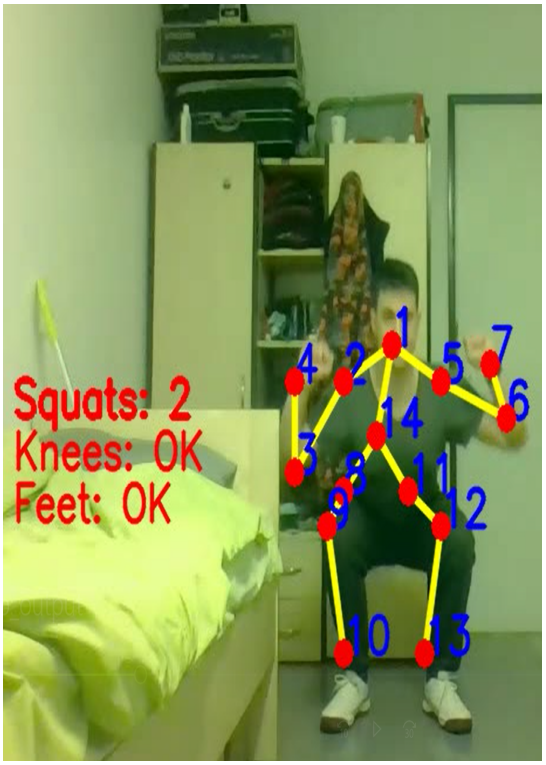

Figure 3.4: Final squat position

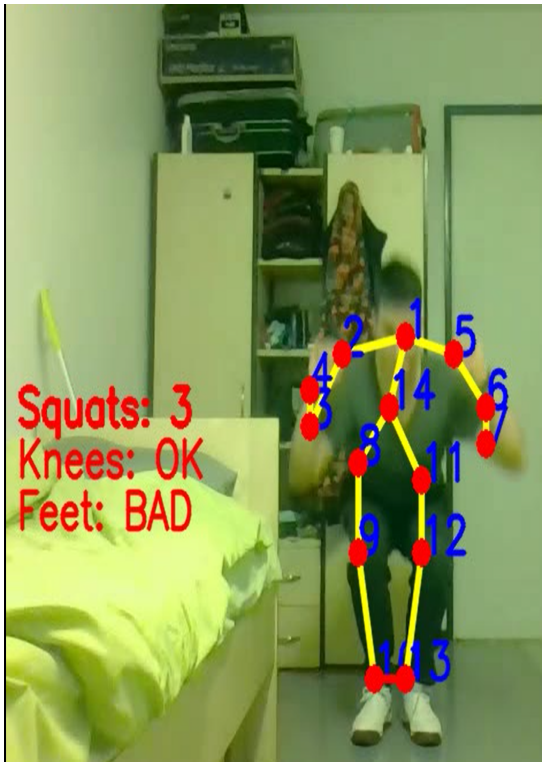

Figure 3.5: Incorrect position because feet are too close

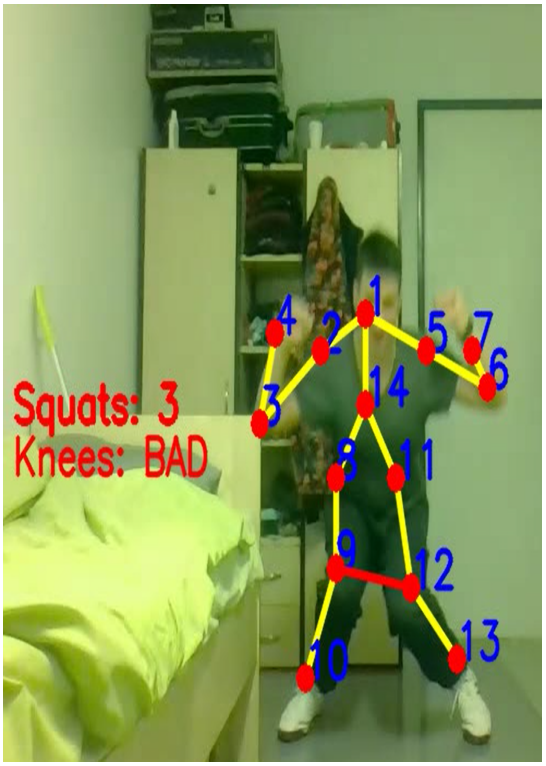

Figure 3.6: Incorrect position because knees are too close (going in an)

4. Code optimization

The main problem was that we didn't have enough computational power to make it work real time. At the beginning we were running the model on a laptops CPU and for the camera input it took 1.3 seconds per frame. We tried different methods like lowering the quality, reducing the number of frames per second of the camera input and generating a region of interest so we don't have to process the whole image, but it didn't help much. Later, after some struggling, we managed to enable CUDA[1] on the laptop, and run the code on the GPU which increased the performances a bit, to 0.18 seconds per frame, and we had 20 frames per second, which was still not good enough.

One workaround is that we take the video with the camera in real time, but then process it asynchronously and get the whole video output later, which is what we decided it is the best approach for the time being.

[1] https://www.nvidia.com/en-gb/geforce/technologies/cuda/

5. Conclusion and future work

In the short time we had we managed to implement a skeleton based detection model for workout tracking of 2 different types of workouts. By using the Human Pose Dataset we trained a deep-learning model with which we were able to detect the 14 different points on the human body. That allowed us to trace the distances and the relative positions of the keypoints, which we used to determine if the posture is correct during the whole duration of the workout, whilst notifying the user if the posture is correct or not. Also we used that information to calculate how many reps a person did based on the distance of a certain keypoints which are essential for that specific workout. The main issue we faced throughout the project was that we weren't able to optimize the model enough to have the application work in real time. That's why we resorted to processing a prerecorded video instead.

In the future, a more optimised model could be created which would allow the application to work in real time. One of the methods with which we could improve it is by running the program on a device with a much higher computational power. Although we think the biggest improvement would come out of additional model optimization. The improvements to the workout can also be implemented. For example, if we had a multi-camera system with cameras recording the user from different angles we could also be able to check some other aspects of the exercise, like if your back are straight during the squats, if you're leaning too much or something similar. And of course there are a lot of different exercises that can be done in the gym and for a lot of them this kind of system could be implemented. There are a lot of possibilities and a lot of potential for this kind of systems, so we leave the rest to the future generations.

6. Sources

1. Deep Learning with PyTorch: Build, Train, and Tune Neural Networks Using Python Tools, 2020. Eli Stevens, Luca Antiga, Thomas Viehmann

2. Learning OpenCV, 2008., Adrian Kaehler, Gary Bradski

3. An Overview of Human Pose Estimation with Deep Learning, Bharath Raj https://medium.com/beyondminds/an-overview-of-human-pose-estimation-with-deep-learning-d49eb656739b

4. How to Build a Real-time Hand-Detector using Neural Networks (SSD) on Tensorflow, Victor Dibia https://medium.com/@victor.dibia/how-to-build-a-real-time-hand-detector-using-neural-networks-ssd-on-tensorflow-d6bac0e4b2ce