Low-cost real-time human motion capturing system

Milutin Nikolić, Lazar Milić, Milutin Studen, Mirko Raković

***Abstract*— In recent years motion capturing technology found numerous applications in industry and research areas like human-robot interaction, medical applications, etc... Those systems can be very expensive and might require a lot of setup time. In this paper, leveraging the advances in deep learning and computer science, the low-cost real-time motion capturing system is presented. The system was designed to use off-the-shelf inexpensive cameras, freely available software, and a gaming laptop. The system design, underlying math principles, reconstruction pipeline, and reconstruction results will be discussed in the paper. The presented motion capturing system can reconstruct a human pose with 33 keypoints in real-time at 17Hz. The whole setup costs less than $2500 including the price of the dedicated PC.**

***Index Terms*— Motion capturing, Human pose estimation, Human-robot interaction, Deep learning**

I. Introduction

Motion capture or "MoCap", for many years now, has been recognized in the majority of industries as the highly advanced technique of recording movements and transferring the results of such recording into digital data. It is widely used in various fields from science, through the film industry, to gaming and video game development, which makes it broadly applicable and remarkably alluring for use in different professional spheres [1].

In medicine and biomedicine, motion capture is used to conduct scientific research and analyses with regard to the understanding of human physiology, particularly the bipedal locomotion. This knowledge can contribute to understanding the effects of injuries and required rehabilitation [2]. In addition, MoCap havs the ability to record facial expressions of humans, which can help us in further understanding of human emotions. In robotics MoCap is used for constructing dynamically stable humanoid movement and for capturing motion trajectory of a human [3, 4].

Furthermore, the filmmaking and video games industry uses motion capture to record the physical actions of the live actors participating in the movies, enabling the real-life movie characters to be "transcribed" into the computer animations, i.e., digital characters. Nonetheless, the MoCap is often used for the creation of special effects in different animated content such as movies, video games, etc.[5] Also, in human-robot interaction, the MoCap is used to record, evaluate and classify human behavior patterns. Also, MoCap is extensively used to record and evaluate human behavior, and modify robot behavior based on recognized human behavior pattern [6].

Milutin Nikolić, Lazar Milić, Milutin Studen and Mirko Raković are with Faculty of Technical Sciences, University of Novi Sad. Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia (email: {milutinn, miliclazar, studen, rakovicm}@uns.ac.rs)

It may be easily concluded that, from its beginnings, motion capture has been a significant addition to the technical processes throughout the diverse industries. However, although many excellent systems leading to the outstanding results are currently present on the market, such as the Vicon Motion[1]

Capturing System, the main setback of motion capture as an asset able to simplify the variety of industrial processes is the fact that its application and practical use are conditioned upon expensive hardware and software. Furthermore, some solutions such as [7] include complex offline postprocessing following the collection of the movement data. For instance, the basic equipment, without any upgrades and advancements, that is necessary for the Vicon setup amounts to approximately $40.000, which is noticeably expensive for the SME.

Although there are many types of motion capture, optical MoCap is most commonly used technology. Optical MoCap use multi-camera setup and triangulation to determine position of markers in 3D world. There are two types of optical motion capture reflective and pulse LED. First technique has reflective markers which are placed on actor's body and because of their reflection it becomes easy for software to determine the position of markers. On the other hand, pulse LED technique has active markers that emission LED light which cameras can capture. Both techniques required 50 plus markers on actor's body for motion to be captured. This process requires a lot of time for markers to be placed on predefined part of body and also markers restrict actor's movement.

Regardless of that, the rapid development of the technology has through the years led to the possibility of the development of low-cost MoCap systems. In addition, the introduction of new methods that can extract the pose from image sequence by using artificial neural networks lowered the cost of new budget MoCap systems. Furthermore, the decrease in the prices of usable equipment for the motion capture processes, including personal computers and cameras, has affected the realization of the more affordable MoCap systems with non-equal but satisfying results [8]. Having in mind all previously mentioned, we are now able to construct budget MoCap system for less than $2500, including all hardware and software, without any markers which minimize required time for preparation, allows actor's to move freely and eliminates location restrictions. Tha low cost would enable further expansion of use of MoCap. Also, removal of required markers and long setup time, enables recording in places where we can't interfere with the subjects, e.g. recording athletes during sport events, behavioral studies of people in pubic spaces, interaction of workers with the robot

 ISBN 978-86-7466-930-3

on the factory floor, etc ... The design and implementation of such a low-cost system is the main topic of this paper.

The paper is organized as follows: in Section II we will describe the hardware components of the system and how it is layed out. In Section III the process for calibrating the system will be described. Section IV describes the whole reconstruction pipeline, while Section V gives experiment results. The paper is concluded in Section VI.

II. System setup

In this section, we will describe the complete system architecture, calibration process, and video acquisition. The video acquisition and recording system consists of three Intel RealSense D415 cameras, which are connected to a PC via a USB interface. The placement of the cameras in the room is so that each camera is oriented towards the center of the room. Since Intel RealSense D415 has narrow field of view, the coverage of the visible area is relatively small. This could potentially disrupt pose estimation. But as shown later in section results this was not a problem.

A. Intrinsic camera calibration

Because of the camera manufacturing process nature, there is a high possibility for it to be assembled imperfectly. This imperfection brings distortions to the image. There are two major distortions, and those are radial and tangential distortions.

Radial distortion has stronger effect on further points in the image and is reflected in such a way that straight lines appear curved. Similarly, tangential distortion occurs when camera lens is not perfectly aligned with image plane.

To describe radial and tangential distortion these five coefficients are used: k_1, k_2, k_3, p_1, p_2, where k_1, k_2 and k_3 are used to represent radial distortion, while p_1 and p_2 describe tangential distortion [9].

Additionally, during camera calibration, intrinsic camera parameters are calculated and these are focal length (f_x, f_y) and optical centers (c_x, c_y). Intrinsic camera and distortion parameters are calculated using OpenCV library. Process of calibration consists of multiple image captures, where each image is showing black and white chessboard with known square size. Images are then processed with the help of OpenCV functions which return all necessary parameters as the end result. Fig 1. shows example of detected chessboard and its corner points during calibration process.

B. Extrinsic camera calibration

After finding intrinsic camera parameters, it is necessary to determine camera positions in real world, also known as extrinsic camera parameters, using chessboard as a reference. Again, we are using OpenCV library to calculate these transformations and Fig **2.** shows example of images containing chessboard and its estimated location in the world.

III. Reconstruction pipeline

In order to reconstruct the pose of the human in 3D space the images coming from cameras have pass through a several processing stages. The flow chart of the raw data obtained from cameras to the final reconstructed pose is illustrated in Fig. 3. Each stage of the reconstruction pipeline will be described in detail in subsequent sections.

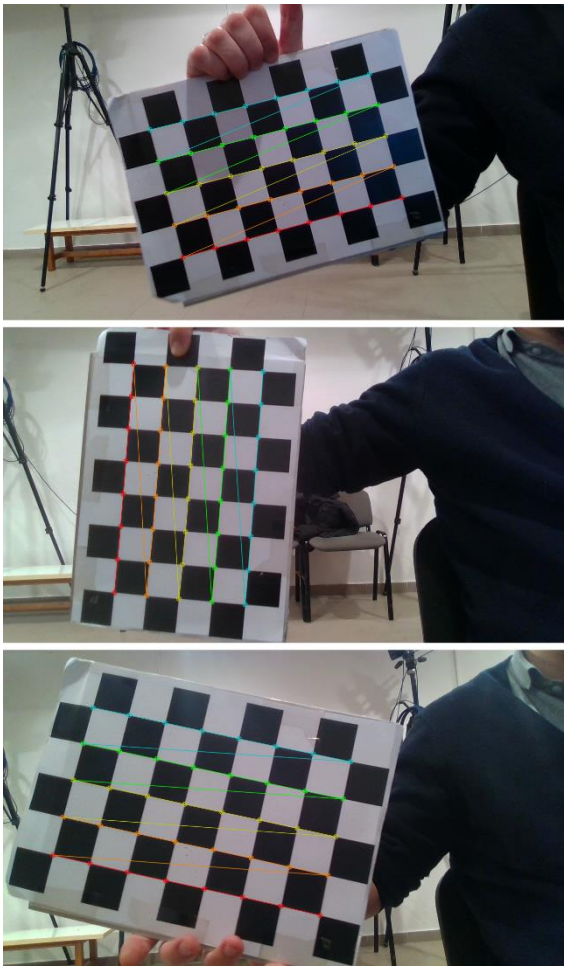

Fig 1. Images acquired for camera calibration with detected chessboards

Fig 2. Images used for extrinsic camera calibration displayed with system origin

A. Image acquisition

The first block in the processing pipeline is image acquisition from the camera. This operation is the simplest one, and the image data from the camera is retrieved by using API provided by the camera manufacturer, in this case python library `pyrealsense2` provided and maintained by Intel.

 ISBN 978-86-7466-930-3

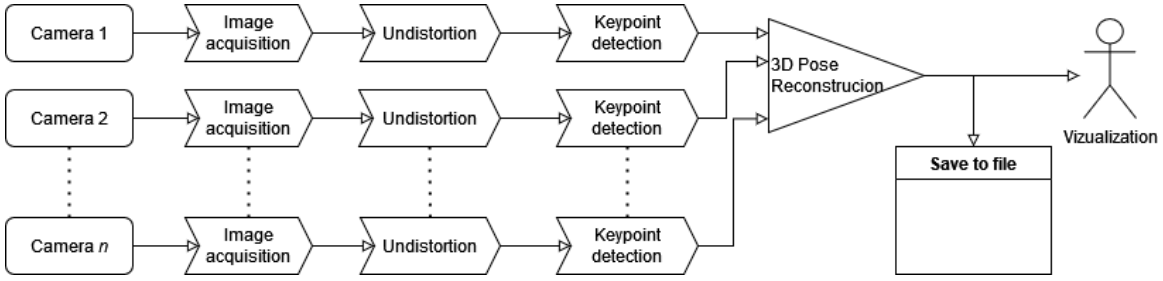

Fig 3. Processing pipeline

[10] Before retrieving the image, user has to provide camera serial number, and desired framerate.

B. Undistortion

The obtained image contains some distortion due to lens imperfection and assembly errors. Such a distortion will introduce errors in the reconstructed pose, so it is extremely important to correct it. In section III we have described the camera calibration process, and result of it is the distortion parameters. In this block, those distortion parameters will be used to undistort the obtained image.

C. Human pose detection

After obtaining distortion-free images the next step is the detection of the human keypoints on the image. Until very recently, without special clothing or markers, accurate detection of body keypoints was almost impossible. In year 2017 OpenPose was developed as a first convolutional neural network based human pose detection framework.[11] It was trained using COCO dataset and it can detect 25 keypoints on the human body. Also, it can detect multiple humans captured by the camera.

Unfortunately, OpenPose, although very accurate, is very computationally demanding. In order to run OpenPose real time, dedicated GPU is required which would significantly increase the total system cost.

Alternative was found in MediaPipe [12] developed by Google as an open-source cross platform, customizable machine learning solution set for live and streaming media. One of solutions provided is used for human pose detection and tracking. It can infer 33 3D keypoints and background segmentation mask on the whole body from RGB video frames utilizing BlazePose [13]. The detected keypoints are given in Fig 4. The MediaPipe can be set up to detect even higher fidelity mode, with total 543 keypoints, with 33 keypoints on the body, 21 on each hand and 468 on the face. To be able to reconstruct face and whole body, the higher resolution and quality camera is required, and for that reason we decided to reconstruct only 33 body keypoints. One limitation of this framework compared to OpenPose is that it can detect only one human in the scene. Multiple humans would inevitably produce detection errors.

3D poses are given relative to pelvis, whose 3D pose is unknown, and, if just one camera is used, can suffer from unknown scale, leading to wrong 3D pose estimation. Hence, we still need multi-camera setup to obtain full 3D position of the keypoints.

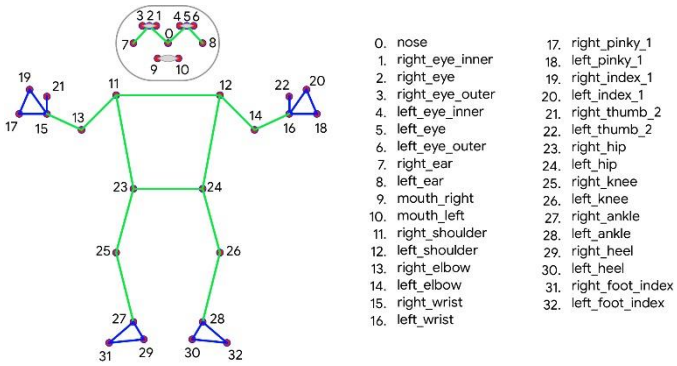

Fig 4. Keypoints detected by BlazePose

D. 3D reconstruction

After getting positions of all keypoints on all cameras, the last step is to reconstruct their positions in a 3D space. To do so, let's assume that we have *n* cameras. The keypoint in space **Q** can be seen on camera *i* at the location $\mathbf{q}_i$ given in pixels. The position and rotation matrix of camera *i* are given with $\mathbf{T}_i$ and $\mathbf{R}_i$. The intrinsic camera matrix is $\mathbf{M}_i$. These vectors and matrices are the result of the calibration process described in section IIIA. The setup is illustrated in Fig. 5. Now, having all the camera positions, rotation matrices and location of the keypoint on all images, the goal is to compute the position of the point **Q** in the 3D space.

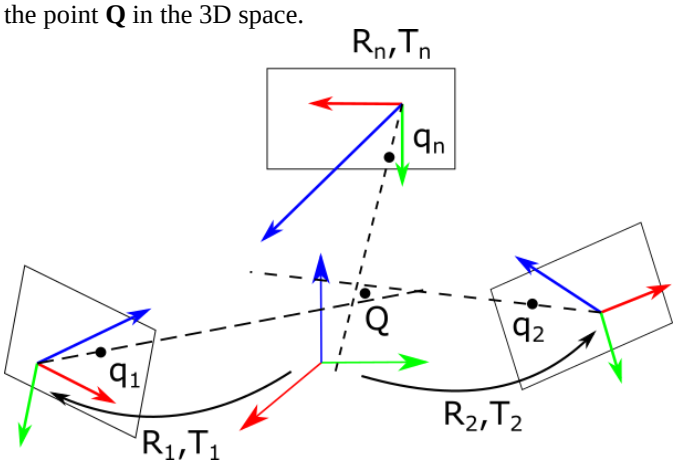

Fig. 5 Setup with multiple cameras observing the same point.

To derive equations, we will start from pinhole camera model, and projection rule:

$$\mathbf{q}_i = s_i \mathbf{M}_i \mathbf{R}_i^{\mathrm{T}} (\mathbf{Q} - \mathbf{T}_i) . \quad (1)$$

 ISBN 978-86-7466-930-3

Where s_i is positive scalar and $\mathbf{M}_i$ is intrinsic camera matrix projecting points from the space to pixels in image. From that equation we can deduce $\mathbf{Q}$:

$$\mathbf{Q} = \mathbf{T}_i + w_i \mathbf{R}_i \mathbf{M}_i^{-1} \mathbf{q}_i = \mathbf{T}_i + w_i \mathbf{u}_i, \quad (2)$$

where we w_i is again positive scalar equal to $1/s_i$. The physical meaning is that a point $\mathbf{Q}$ lies on the ray starting from center of camera i $\mathbf{T}_i$, that passes through point $\mathbf{q}_i$ and follows direction $\mathbf{u}_i$.

Obviously, this is the ideal case, where cameras are perfectly calibrated and keypoint is perfectly detected. In the real world we can't achieve that, so the mentioned ray will pass close, but will not contain the point $\mathbf{Q}$. Including that observation we can rewrite equation (2):

$$\mathbf{Q} = \mathbf{T}_i + w_i \mathbf{u}_i + \boldsymbol{\varepsilon}_i, \quad (3)$$

where $\boldsymbol{\varepsilon}_i$ represents the deviation of point $\mathbf{Q}$ from ray going from $\mathbf{T}_i$, in a direction $\mathbf{u}_i$.

The position of the point Q will be obtained by minimizing the deviation from ideal rays. It is the result of the following optimization problem:

$$\min_{\mathbf{Q}} \sum_{i=1}^{n} \left\| \boldsymbol{\varepsilon}_i \right\|^2. \quad (4)$$

The meaning of this optimization problem is that the point $\mathbf{Q}$ that is the detected on n images at location $\mathbf{q}_i$ is the point in space whose total squared distance to all rays starting at $\mathbf{T}_i$, and passing through point $\mathbf{q}_i$ is minimal.

By expressing inserting $\boldsymbol{\varepsilon}_i$ from equation (3) and inserting it into (4) and some matrix manipulation we can the optimization problem as:

$$\min_{\mathbf{Q},\, w_i} \left\| \begin{bmatrix} 1_{3x3} & -\mathbf{u}_1 & 0_{3x1} & \dots & 0_{3x1} \\ 1_{3x3} & 0_{3x1} & -\mathbf{u}_2 & \dots & 0_{3x1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1_{3x3} & 0_{3x1} & 0_{3x1} & \dots & -\mathbf{u}_n \end{bmatrix} \begin{bmatrix} \mathbf{Q} \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} - \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \\ \vdots \\ \mathbf{T}_3 \end{bmatrix} \right\|^2 \quad (5)$$

Such an optimization problem has a well known solution:

$$\begin{bmatrix} \mathbf{Q} \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} 1_{3x3} & -\mathbf{u}_1 & 0_{3x1} & \dots & 0_{3x1} \\ 1_{3x3} & 0_{3x1} & -\mathbf{u}_2 & \dots & 0_{3x1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1_{3x3} & 0_{3x1} & 0_{3x1} & \dots & -\mathbf{u}_n \end{bmatrix}^{+} \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \\ \vdots \\ \mathbf{T}_n \end{bmatrix} = \mathbf{A}^{+} \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \\ \vdots \\ \mathbf{T}_n \end{bmatrix} \quad (6)$$

where we introduced substitute $\mathbf{A}$ for shorter notation and plus (+) represents a MP-inverse matrix.

It is important to note that matrix $\mathbf{A}$ has 3n rows and 3+n columns and for it to be tall, guaranteeing unique solution n has to be at least 2, meaning that we need to see the waypoint at, at least two cameras to be able to reconstruct the pose. Detecting the same waypoint on more than 2 cameras will produce more accurate result. Also, matrix $\mathbf{A}$ is different for each keypoint, so in order to reconstruct model with e.g 33 keypoints, we need to construct 33 different matrices and solve 33 different optimization problems.This process has to be performed for each recorded frame.

E. Data post processing

At this point in pipeline, we have already fully reconstructed the 3D pose of all the keypoints. The last basically depends on the usage of MoCap system. We can either save data for later use, or directly stream it to dedicated video/gaming production software, medical application, etc …

IV. RESULTS

Complete process of image acquisition, undistortion, pose estimation and 3D reconstruction is done on laptop with Intel Core i7 10750H, 6 physical & 12 logical core 10th generation CPU with 16GB of RAM and Nvidia GeForce 1660TI GPU. Even so, we were able to achieve stable 20 FPS. The code has been implemented on Ubuntu 20.04 Focal Fossa in Python3 with extensive use of `multiprocess` library, enabling parallelization. Separate process responsible for image acquisition, undistortion and pose estimation was spawned for each camera, with one additional process tasked with 3D reconstruction, and data post processing.

Fig 6. A frame from all 3 cameras taken at the same time with detected keypoints drawn over

 ISBN 978-86-7466-930-3

In Fig 6, the images from all three cameras taken at the same time are shown. The keypoints detected by MediaPipe are overlaid on top of the undistorted images. I can be noted that not all markers are detected on all 3 images, e.g right arm is not detected on first image. Nevertheless, as seen in Fig 7 it can be observed that the system was able to reconstruct the full human pose, with all keypoints. That comes from the fact that for 3D reconstruction we need a keypoint to be detected at minimum 2 cameras. Detection of a keypoint is more than 2 cameras, increases estimation accuracy. It can be noted that the reconstructed full body pose closely matches the one shown in undistorted images.

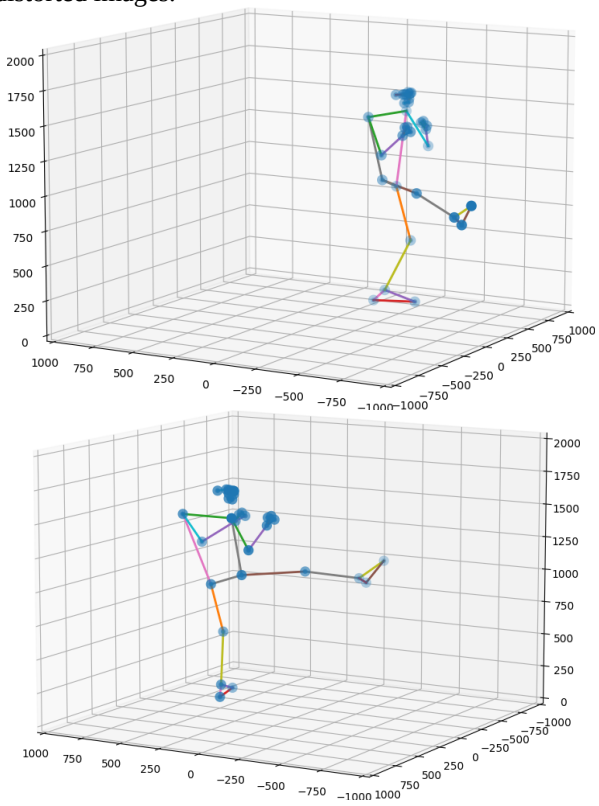

Fig 7. Full body pose reconstructed from images shown in Fig.6

However, during pose estimation some problems occurred. We can observe several frames with incorrectly detected human poses as shown in Fig. 8. Those bad reconstructions can be a result of several issues. Firstly some body parts can go out

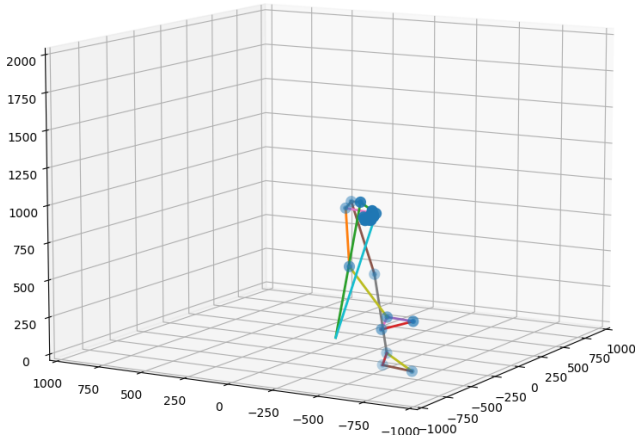

Fig 8. Incorrectly reconstructed human pose

of recording volume or be occluded by objects in the scene, thus preventing MedaPipe to detect keypoints successfully. Other issues can come out of just wrong detection pose detection as shown in the middle of Fig 9. These images show MediaPipes' inability to estimate human pose in non-standing positions. That is a result of MediaPipe being trained on dataset which consists mostly of humans in standing position. That introduces strong bias in the CNN, which expects the torso to be upright. In Fig 9. that was not the case, and MediaPipe has detected a chair in the background as humans face and shoulders. That frame prevents successful reconstruction.

V. Conclusion

In this paper we have presented a budget MoCap system. The system uses inexpensive of-the shelf cameras with a middle-range gaming laptop. Used libraries and software are all open source, with licenses allowing commercial use. Hence, most of the price of the system comes from gaming laptop. The price of the full system was under $2500, but uses common hardware that we had at our disposal, so the MoCap didn't cost us any money. The framerate that we could have achieved was 20FPS. The reconstructed poses closely matched the pose of the actor that was recorded and were stable as long as the actor was within recording volume.

Except low price, the system doesn't require any markers, which significantly reduces setup time. There is no need for

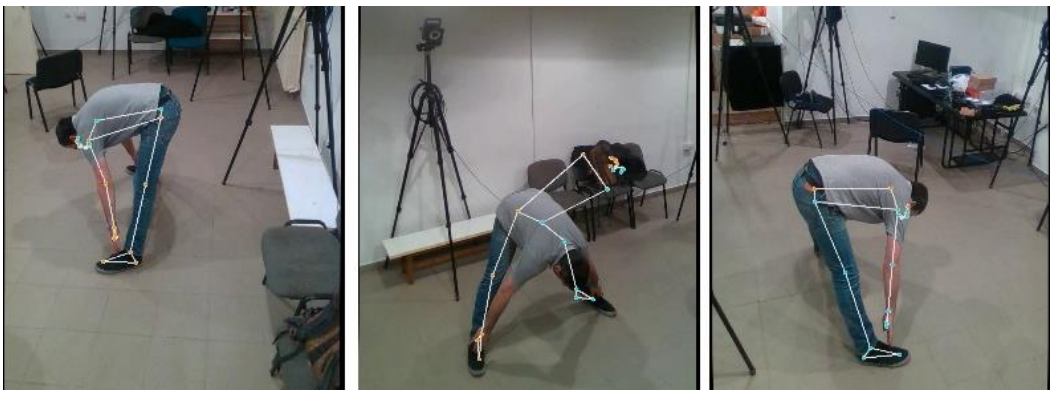

Fig 9. A frame from all 3 cameras taken at the same time with detected keypoints drawn over with incorrect human pose detections

 ISBN 978-86-7466-930-3

tedious process of attaching markers to the actor. The other benefits from this is that people can be recorded while performing their task without interference. That means, the person's motion isn't modified by the fact that there are markers attached to it's body. For behavioral studies, person's motion can be recorded subject's knowledge, reducing the chance for the person to unconsciously alter the behavior. This makes developed MoCap, extremely potent for studies in sport science and behavioral studies with potential HMI application.

Although presented MoCap has a lot of potential there are a few issues that would be the topic for the future research. The current approach doesn't include the temporal continuity of the recording, but each frame is considered independently. The system doesn't consider human model, which would make detection more accurate and more robust to false detections and outliers. Finally, current deep learning backend, MediaPipe can detect only one person. OpenPose, can be used instead, but that comes at the cost of framerate and ability to run system in real time or at the price of higher performance and higher price hardware.

REFERENCES

[1] Menolotto, M., Komaris, D. S., Tedesco, S., O'Flynn, B., & Walsh, M. (2020). Motion capture technology in industrial applications: A systematic review. Sensors, 20(19), 5687.

[2] Schönauer, C., Pintaric, T., Kaufmann, H., Jansen-Kosterink, S., & Vollenbroek-Hutten, M. (2011, June). Chronic pain rehabilitation with a serious game using multimodal input. In 2011 International Conference on Virtual Rehabilitation (pp. 1-8). IEEE.

[3] Ogrinc, M., Gams, A., Petrič, T., Sugimoto, N., Ude, A., & Morimoto, J. (2013, May). Motion capture and reinforcement learning of dynamically stable humanoid movement primitives. In 2013 IEEE International Conference on Robotics and Automation (pp. 5284-5290). IEEE.

[4] Ito, T., Ayusawa, K., Yoshida, E., & Kobayashi, H. (2020). Simultaneous Control Framework for Humanoid Tracking Human Movement with Interacting Wearable Assistive Device. IEEE Robotics and Automation Letters, 5(2), 3604-3611.

[5] Duarte, N. F., Raković, M., Tasevski, J., Coco, M. I., Billard, A., & Santos-Victor, J. (2018). Action anticipation: Reading the intentions of humans and robots. *IEEE Robotics and Automation Letters*, *3*(4), 4132-4139.

[6] Sharma, S., Verma, S., Kumar, M., & Sharma, L. (2019, February). Use of motion capture in 3D animation: motion capture systems, challenges, and recent trends. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) (pp. 289-294). IEEE.

[7] Ikegami, Y., Nikolić, M., Yamada, A., Zhang, L., Ooke, N., & Nakamura, Y. (2020). Whole-Game Motion Capturing of Team Sports: System Architecture and Integrated Calibration. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 10256-10261). IEEE.

[8] Thewlis, D., Bishop, C., Daniell, N., & Paul, G. (2013). Next-generation low-cost motion capture systems can provide comparable spatial accuracy to high-end systems. Journal of applied biomechanics, 29(1), 112-117.

[9] Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

[10] Intel RealSense SDK2 https://dev.intelrealsense.com/docs/python2 Accessed: May 8. 2022.

[11] Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7291-7299).

[12] MediaPipe Pose Module, https://google.github.io/mediapipe/solutions/pose, Accessed May 8, 2022

[13] Bazarevsky, V., & Grishchenko, I. (2020). On-device, real-time body pose tracking with mediapipe blazepose. *Google AI Blog*.

 ISBN 978-86-7466-930-3