Analysing Real-Time Vision-Based Pose Estimation Algorithms for RITH Purposes on Embedded Devices

ANDREI POPOVICI, University of Twente, The Netherlands

Additional Key Words and Phrases: Telerehabilitation, Pose estimation, Physical rehabilitation, Movement assessment methods

1 ABSTRACT

Rehabilitation programs are vital for the people that suffer injuries. The Covid-19 pandemic made it impossible for these people to follow such programs at healthcare centers. This study aims to develop a new solution for patients opting instead for a Rehabilitation in the Home (RITH) program. The proposed approach is divided into two phases. Firstly, the app tracks and extracts the 2D coordinates of the joints. The second phase uses these extracted coordinates to guide the patient during the exercise. All captured frames are compared to the reference frames, and their similarity is computed. Developing a home-based rehabilitation helper that will work on low-resource devices will bring benefits, including decreased travel time and flexible exercise hours for patients.

2 INTRODUCTION

Telerehabilitation has shown significant results due to the development of new technologies. Nowadays, the greater public health funding allocation and telerehabilitation have improved significantly [1]. The technological advances substituted traditional face-to-face rehabilitation with telerehabilitation. In that way, the patient can follow the entire course from home. Telerehabilitation also helps patients reduce hospitalisation periods and expenditures; for patients and health care providers - provides a remote environment where the patient can follow up the entire rehabilitation program without leaving his home. During the COVID-19 pandemic, the practice of rehabilitation proved impossible for patients, as all hospitals and health care centres were quarantined. Thus, the patients faced a problem that was impossible to solve. In this situation, telerehabilitation is their only solution. The recent developments in artificial vision techniques and machine learning improved the accuracy of human posture estimation, showing a potential practical application in the field of telerehabilitation. To reduce patients' expenditures and ensure greater accessibility to rehabilitation services, some researchers proposed a new way of conducting the telerehabilitation, namely using 2D pose detection libraries such as OpenPose, BlazePose, and wrnchAI [2], which don't require the physical presence of a rehabilitation specialist and gives the patient the opportunity to practice all the rehabilitation exercises alone at home. Although the motivations for the above solution approach sound good. Although the justification for the above-proposed solution approach sounds reasonable, the current state of the art of pose estimation needs more computational power to determine the pose of a human in real-time.

If an embedded device solution like mobile devices can be identified, it will help many in need. Specifically, the proposed research is based on the idea that many patients may face problems getting to a healthcare centre, the lack of available rehabilitation specialists at a particular time in that area, or the patient's schedule. Consequently, the main aim of this research is to develop an app that will allow each affected patient to get the necessary rehabilitation regardless of their location. This is also a first step towards improving the accessibility to rehabilitation indifferent of social status.

In light of the above, the work aims to ease the rehabilitation process for patients by providing them with a new way to practice the exercises wherever they are. However, numerous AI-powered personal trainers use the latest technology to assess the quality of exercises. The need for a post-injury rehabilitation mobile app is still demanding. Given that it is computationally expensive to estimate the pose of a human in real-time, it is pretty hard to run those algorithms on embedded devices. It is necessary to find a way to improve the accuracy of human posture estimation and provide real-time guidance to the patient in the process of rehabilitation. The following research questions (RQ) have been defined as the foundation of the research.

(1) How to track the rehabilitation exercises in real-time with a low-resource device?
(2) How to guide the patient during the exercise?
(3) How to improve the existing keypoint topology to increase the impact of neck pain exercises?

The remainder of this paper is organised as follows. The related work on the topic is described in the third section. The fourth section describes the methodology used to answer the research questions. Following that the fifth section presents the paper's results. The sixth section formulates the conclusions of this paper. Furthermore, the seventh section describes what could be some of the improvements to the current version of the research.

3 RELATED WORK

In this section, we would like to examine some related work on this topic. Over the years, much research has been done on pose estimation and telerehabilitation.

In 2002, Jurgen Broeren et al. [3] used a haptic device as a cinematic utility to assess the rehabilitation exercises of the patients that suffered strokes. In their 2010 paper, Marco Rogant et al.[1] show telerehabilitation's state of the art and its importance to society. In 2010, Portia E Taylor et al. [4] used body-worn tri-axial accelerometers to estimate the quality of the rehabilitation exercises done by the patients. They built a classifier that successfully labels the exercises as correct or incorrect. In 2010, Luis Enrique Sucar et

TScIT 37, July 8, 2022, Enschede, The Netherlands

al. [5] developed a vision-based system to monitor the performance of the patients while following the rehabilitation programme. They used a pressure-sensitive gripper for hand and finger rehabilitation to assess the execution of the exercises. In their work, David Antón et al. [6] developed a telerehabilitation system based on Kinect that helps the user perform rehabilitation exercises by displaying a 3D avatar that shows the correct execution of the movement. In this way, the user is guided through the whole process. Work by Tomasz Hachaj and Marek R. Ogiela [7] describes a new approach to implementing a classifier that can recognise in real-time human body poses and gestures in real time. In their work, Adeline Paiement et al. [8] proposed a new method for human pose estimation based on the Kinect skeleton data. They developed a statistical model for the ideal movements of healthy subjects. Afterwards, they used this model to compute the similarity score between the user's movements and the references. In their work, Ming-Chun Huang et al. [9] develop a framework to supervise the on-bed range of motion exercises using a pressure-sensitive bed sheet. Next, they analyse the results using manifold learning techniques. In their paper, Marianna Capecci et al. [10] provide an accuracy analysis of the Kinect v2 sensor for a rehabilitation program. They use the joint positions and angles for the evaluation of the accuracy. In 2016, Aleksandar Vakanski et al. [11] proposed a new methodology for evaluating human postures. They use the latest progress in neural networks and machine learning technologies to build a parametric model of human motions. Ben Crabbe et al. [12] in their work describe a novel approach for estimating the body pose. Their approach uses a Convolutional neural network (CNN) to map a person's pose space location to their depth-silhouette. There has also been much research carried out on pose estimation and posture correction using Microsoft Kinect. Elham Saraee et al. [13] developed a system called PosureCheck that scores the patient's posture while performing the exercise in front of the camera by using Bayesian estimation and majority voting for classifying the posture, whether correct or not. Lynne V.Gauthier et al. [14] studied the efficiency of a rehabilitation program for stroke patients. The gameplay took place in the home environment of each patient. Wan-wen Liao et al. work [15] investigates the role of Kinect-based upper extremity rehabilitation performance for chronic stroke survivors. The work by Yalin Liao et al. [16] analyses different methods for evaluating a patient's performance in a rehabilitation program. Work by Talal Alatiah and Chen Chen [17] points out the necessity of machine learning in judging an athlete's performance during competition. Previous works focused more on using accelerometer sensors to assess the posture during exercise, work by Meera Radhakrishnan et al. [18] uses an inertial sensor mounted on weight equipment that identifies the mistakes in the execution of the exercises. Furthermore, in 2020 Steven Chen and Richard R. Yang [19] developed a new approach to detecting and correcting posture during workouts using pose estimation. They recorded a dataset of over 100 exercise videos and built machine learning algorithms which can distinguish between a well and a poorly performed exercise. Swakshar Deb et al. [20] use GCN for assessing rehabilitation exercises given skeleton data of a movement.

4 METHODOLOGY

The system developed for this paper is a guide for the correct execution of rehabilitation exercises. It tracks the movements and gives guidance, ensuring that the user completes the exercise correctly and does not miss any phase of an exercise. We think of a rehabilitation exercise as a series of consecutive poses (see Fig. 1). Thus we can easily convert them into a set of phases that the device can understand and process.

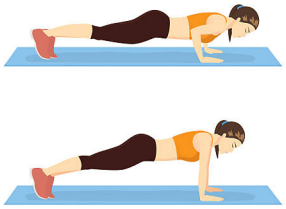


Fig. 1. Push-up phases. (2016, July 25). [Illustration]. Istockphoto.Com. https://www.istockphoto.com/nl/vector/step-to-instruction-in-push-up-gm578104104-99362979

4.1 BlazePose

BlazePose is a pose detection model created by Google that finds and returns the x, y, and z coordinates of 33 skeleton keypoints (see Fig. 4). BlazePose is made up of two different machine learning models: a Detector and an Estimator. The Detector removes the human region from the input image, whereas the Estimator inputs a 256x256 resolution image of the recognised person and returns the keypoints [21].

4.1.1 Architecture of BlazePose. The Detector works as follows, given an image as input, it outputs a bounding box and a confidence score. On the other hand, the estimator employs a regression technique with all keypoints supervised by a combination heat map/offset prediction [21] (see Fig. 2). The estimator's output is the keypoints that are made up of 165 components for each of the 33 keypoints of the model.

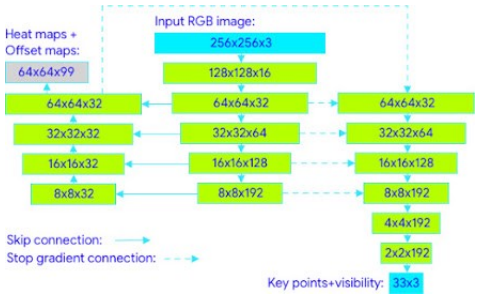


Fig. 2. Bazarevsky, V., Grishchenko, I. (2020, August 13). Tracking network architecture: regression with heatmap supervision [Graph]. https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html

4.2 System Design

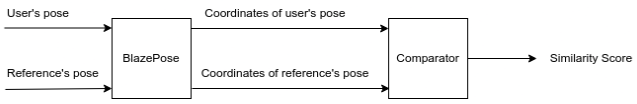
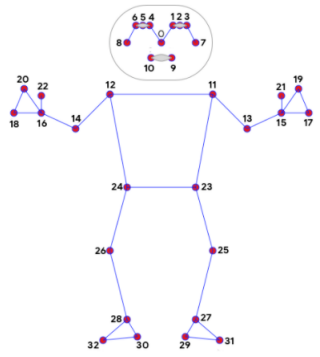


Fig. 3. System design

Our solution was to use BlazePose [21] for tracking human posture because it is the best human pose estimation model at the moment. In Fig. 5 are presented the results of the quality evaluation of 5 models, mainly BlazePose GHUM Heavy, BlazePose GHUM Full, BlazePose GHUM Lite, AlphaPose ResNet50, and Apple Vision. These models were evaluated against three different validation datasets, mainly Yoga, Dance, and HIIT. The performance is evaluated for COCO topology [22]. We see from the Fig. 5 that BlazePose is performing best for the Percentage of Correct Keypoints (PCK), where a detected joint is considered correct if the distance between the predicted and the true joint is < 0.2 * torso diameter. We used BlazePose to collect human poses for each reference image and saved them for comparing them later with the patient's posture. After that, we used those saved coordinates of joints to compute the similarity between the user's pose and the reference image (see Fig. 3). We analysed two comparison methods, namely Weighted Distance and Cosine Similarity. BlazePose returns the coordinates and the in-frame likelihood of the 33 different joints as output. The keypoints returned by the BlazePose have the following format: x and y coordinates and the confidence level of the keypoints. The keypoints returned are shown in Figure 4.



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

Fig. 4. BlazePose 33 keypoint topology. (n.d.). [Graph]. https://google.github.io/mediapipe/solutions/pose.html

4.3 Exercises analysis

In selecting exercises, we decided to choose exercises for specific injuries, mostly related to lower body affections, knee injuries being the most common injuries for older people, athletes, or sports enthusiasts [23]. For this purpose, we selected the most prevalent injuries of the lower body, which are listed below, and selected rehabilitation exercises that are most efficient for these injuries [24]. Besides this, we also picked the Frozen Shoulder injury, as it appeared attractive to study some exercises related to upper body rehabilitation.

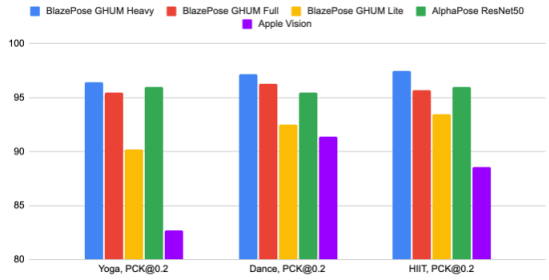


Fig. 5. Quality evaluation in PCK@0.2. (n.d.). [Graph]. https://google.github.io/mediapipe/solutions/pose.html

- Femoral Fracture
  (1) Knee flexion supine
  (2) Knee extension supine
  (3) Hip abduction
  (4) Leg lift
  (5) Hip flexion side-lying
- ACL Sprain
  (1) Heel Slide
  (2) Knee flexion supine
  (3) Hip abduction gluteus medius side-lying
- Baker's Cyst
  (1) Knee flexion prone
  (2) Football kicks with band
  (3) Knee extension prone
  (4) Full wall squat
- Condramalacia Patella
  (1) Football kicks with band
  (2) Hip abduction gluteus medius side-lying
  (3) Quadriceps stretch
- ACL RuptureRecon
  (1) Knee flexion supine
  (2) Knee extension supine
- Frozen Shoulder
  (1) Passive shoulder flexion
  (2) Elevation through abduction
  (3) Drawing the sword

4.4 Image Processing

Processing the reference images is the bottom line of our research, which is why, after finishing the exercise selection, we started searching for videos of the exercises mentioned above. Googling around, we found for each exercise a reference video, where a professional physiotherapist shows how to perform it correctly. Then, we developed a Python script that extracts the frames from a video. Firstly, we are asked to decide on the starting and ending time codes for a repetition of the exercise and the desired saving frames per second (see Fig. 6). Next, for each exercise, we found descriptions of perfect performance. Those descriptions helped us find the phases of an exercise. For an example, see Fig. 1. Knowing the phases, we selected the most relevant frame for each phase. (see Fig. 8). After manually selecting the most relevant frames, we process them with BlazePose.

```
What is the start of the Passive_Shoulder_Flexion.mp4 video?0
What is the end of the Passive_Shoulder_Flexion.mp4 video?5
What is the desired savings frames per second for Passive_Shoulder_Flexion.mp4 ?4
```

Fig. 6. Example of the Python script used to extract the reference frames

.

First, we are asked to choose the keypoints we need for the exercise evaluation. We select them by typing the ID of the keypoint from the BlazePose Topology (see Fig. 7). For instance, in the Elevation through Abduction exercise (see Fig. 8), we have eight keypoints that we are interested in, mainly: Left shoulder with ID 11, Right shoulder with ID 12, Left Elbow with ID 13, Right Elbow with ID 14, Left wrist with ID 15, Right wrist with ID 16, Left hip with ID 23 and Right hip with ID 24. While processing the selected frames we apply a segmentation mask to the image, and then we draw pose keypoints for each image chosen (see Fig. 9). While processing reference images with Blazepose, we select, for each phase, the keypoints mentioned above and save them in a .json file for comparison purposes.

```
Please, select the desired landmarks for the analysis:11 12 13 14 15 16
```

Fig. 7. Example of the Python script used to process the reference frames

.

Fig. 8. Example of reference images. (2015, December 3). [Photo]. https://www.youtube.com/watch?v=cP4LLJie9kw

Fig. 9. Example of reference phases

4.5 Similarity computation

Before computing the similarity between reference pose, and actual pose, we considered factors like a person's height and distance from the camera, as these factors vary a lot from person to person. To tackle these differences, pose vectors were firstly scaled and translated to a pose of size 1*1 and then normalised. L2 normalisation was used, which divides each pair of coordinates by their magnitude (see equation 1).

$$|x| = \sqrt{\sum_{k=1}^{n} |x_k|^2} \tag{1}$$

For pose similarity, we tested two distance metrics to measure the similarity between pose vectors, mainly Cosine Similarity and Weighted distance.

4.5.1 Cosine Similarity. The Cosine Similarity measures the similarity between two normalised vectors. It determines if two vectors are pointing in the same direction by calculating the cosine of the angle between them (see equation 2). It returns 1, if they are the same and -1, if they are opposite.

$$\cos\alpha = \frac{\overrightarrow{a} \cdot \overrightarrow{b}}{||\overrightarrow{a}|| ||\overrightarrow{b}||} \tag{2}$$

It will help us find out how related are the two pose vectors by looking at the angle between them instead of magnitude (see Fig. 10)

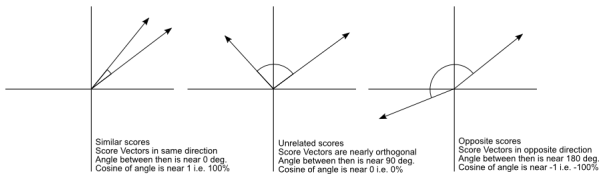


Fig. 10. Visual depiction of cosine similarity. (2013, September 12). [Illustration]. https://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

4.5.2 Weighted Distance. The Weighted Distance technique integrates the in-frame likelihood of each landmark when computing the similarity between pose vectors [25]. The idea is that a high confidence score of a keypoint has a more significant impact on the distance metric than those with a lower score (see equation 4).

$$\frac{1}{\sum_{k=1}^{33} F_{c_k}} * \sum_{k=1}^{33} F_{c_k} ||F_{xy_k} - G_{xy_k}|| \tag{3}$$

In the above formula, F and G are two normalised vectors. F_{c_k} is the in-frame likelihood score of the kth landmark of F. G_{xy}, and F_{xy} are the x and y coordinates of the kth keypoint for each vector [26].

The weighted distance metric showed better results than the cosine similarity metric in detecting whether two poses matched. The weighted distance metric showed better results since it incorporates the confidence scores of each keypoint.

4.6 Custom keypoints detector

During the research, we concluded that there are exercises where the standard Blazepose model is inefficient. Analysing different injuries, we found that there is no way to assess the correctness of performing the exercises related to neck pain because the model does not have the necessary keypoints for this. We decided to train a keypoint detection model on a custom dataset. We used an object detection framework called Detectron2 [27]. In the following sub-sections, we will describe how we implemented this.

4.6.1 Preparing dataset. For a good assessment of neck pain exercises, such as "Right side bending," "Left side bending", and "Cervical nod, neutral", we decided to add five keypoints to our model, namely head, neck, right shoulder, left shoulder, and middle of the spine (see Fig. 11). We chose these particular keypoints because the angles that can be derived from them will help us assess the correctness of the execution of the exercises.

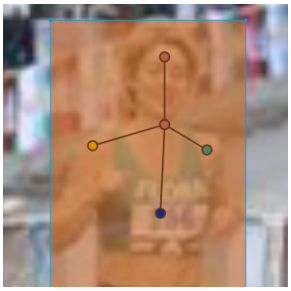


Fig. 11. Custom keypoints representation

We have chosen to use the "MPII Human Pose Dataset" [28] as a dataset. The dataset is a state-of-the-art benchmark for evaluating human pose estimation. All the pictures from the dataset are from YouTube videos and vary widely regarding human positions, surroundings, attire, body size, distance from the annotated figure, and viewpoint. For this task, we have selected 344 images from there that will help us train and test the fine-tuned model. After selecting the images from the dataset, we annotated all the images with the help of the COCO Annotator [29], allowing the images to be labelled efficiently. After successfully annotating the images with the new keypoints, we split the dataset into two parts, 70% into training data and 30% into validation data.

4.6.2 Training the model. After successfully preparing the dataset, we are ready to train the keypoint detection model. We will use the pre-trained R50-FPN Keypoint R-CNN with the following meta parameters used during training (see table 1).

Table 1. Parameters used for training

Parameter	Value
Learning Rate	0.00025
Max. Iterations	2000
BATCH_SIZE_PER_IMAGE	512

4.6.3 Inference. The next step will be to infer the new model on sample images. Trying to predict the newly added keypoints yielded the following result (see Fig. 14). After visually inspecting the results, we see that the developed model predicts the outcomes reasonably well, but still is not accurate enough to predict all the keypoints correctly since the dataset was too small.

4.6.4 Evaluation. For the performance evaluation, we have used two metrics, precision and recall.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

where TP is the number of true positives, FN is the number of false negatives, and FP is the number of false positives. Recall and precision are computed for different threshold values of Intersection over Union (IoU). IoU is calculated based on the overlap between the predicted and ground truth bounding boxes [30]. For instance, if the IoU threshold is 0.5 and the predicted value for the IoU is 0.8, that prediction will be classified as true positive. On the other hand, if the IoU is 0.2, it will be classified as a false positive when it fails to predict the object on the image. The prediction will be classified as a false negative. The recall and precision we got on the validation set for the bounding boxes are illustrated in the figure,12, and for keypoints are shown in the figure13. While analysing the evaluation results for bounding boxes (see Fig. 12), we got mean average precision (mAP) equal to 0.618 for the interval IoU=0.5:0.95, where mAP is computed over all IoU thresholds, and then the average is taken from those values. It is pretty low because the dataset was too small. The mAP@.50IoU metric computes only the average precision for IoU=0.5, the role of this metric is to give an approximate estimation of precision if we are not very strict about the position of the bounding boxes. In this case, it is performing relatively well, with a score of 0.913. The mAP@.75IoU computes the same as mAP@.50IoU, but using IoU=0.75 instead of IoU=0.5, this metric is more strict about the position of the bounding boxes because it requires at least IoU=0.75 to classify the prediction as true positive. In this case, we got 0.747, which means that the vast majority of the bounding boxes are detected correctly, and it is a good score for a dataset of this size. The following three lines in the figure 12 show the results that are computed in the same way as the mAPs above, with a slight difference. The mAPs are divided by the size of the bounding boxes. The small one computes the mAP of the bounding boxes with less than 32^2 pixels. The medium one computes the mAP of the bounding boxes with an area between 32^2 pixels and 96^2 pixels. Moreover, the large one computes the mAP of the bounding boxes with an area greater than 96^2 pixels [22]. For the small and medium sizes of the bounding boxes, we got -1, but for the large sizes, we got 0.619. We got -1 for small and medium sizes of the bounding boxes because the provided bounding boxes while annotating the images were all large, with an area greater than 96^2 pixels. When we look at the mean average recall (mAR), we see that mAR is divided by the number of detections in an image, i.e., 0 <= n <= x, where n is the number of detections and x is the maximum number of detections. COCO evaluator has three values for the maximum number of detections in an image, mainly 1, 10, and 100. Thus the AR@x will calculate the mAR for all images with at most x detections across all IoU thresholds (IoU=0.5:0.95). We see that the score for 0 or 1 detections is equal to 0.658, while the score for images where the number of detections is greater than one is 0.722. The last three lines are the mAR divided by the size of the

detected boxes. For these scores, the same logic applies as for the scores for the mAP. There we again got -1 for the small and medium sizes of the bounding boxes for the same reason as above.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.618
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.913
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.747
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.619
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.658
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.722
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.722
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.722
```

Fig. 12. Evaluation results for bounding box

For the evaluation of the results for keypoints detection (see Fig. 13), we used object keypoint similarity (OKS), similar to the Intersection over Union (IoU) in the case of object detection. OKS computes the overlapping ratio between predicted keypoints and the ground truth keypoints [22].

Taking into account that OKS metric shows how close is the predicted keypoint to the ground truth keypoint and its value is between 0 and 1, where OKS = 0 means that the predicted keypoints are off by more than a few standard deviations and OKS = 1 implies that it predicted all the keypoints perfectly. We got an average precision for all IoUs of 0.943 and a recall of 1, getting mAP and mAR of -1 for the size medium for the same reasons that we have already mentioned when analysing the evaluation results for the bounding boxes. It means that the model predicts the keypoints reasonably well. We also see this when we inspect visually the test images illustrated in the figure 14.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets= 20 ] = 0.943
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets= 20 ] = 0.943
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets= 20 ] = 0.943
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets= 20 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets= 20 ] = 0.943
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 20 ] = 1.000
Average Recall     (AR) @[ IoU=0.50      | area=   all | maxDets= 20 ] = 1.000
Average Recall     (AR) @[ IoU=0.75      | area=   all | maxDets= 20 ] = 1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets= 20 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets= 20 ] = 1.000
```

Fig. 13. Evaluation results for keypoints

5 RESULTS

After processing all the rehabilitation exercises, we developed the native app that was written with React Native. We chose this framework because of the outstanding performance it provides. Next, BlazePose was integrated into the app, allowing us to estimate the pose of a human.

The app's home page (see Fig. 15) allows the patient to select the exercise related to their injury. By tapping on the exercise, the patient is redirected to the exercise page (see Fig. 16), where the patient can view the correct execution of the exercise and select how many repetitions they want to do. Finally, the user presses the "start exercise" button and is redirected to the camera view. The app checks in each frame if all the required keypoints for the exercise are visible

Fig. 14. Results of the inference

and have an in-frame likelihood greater than 0.8. If all the keypoints are present, the app starts guiding the user on how to perform the exercise correctly. It tells the patient, using audio messages, how to get to the first phase of the exercise. During the exercise performance, the app computes the coordinates of each keypoint involved in the exercise and compares it with all the reference phases of the exercise (see Fig. 3). If the similarity between those two vectors is more than 0.9, then the phase is considered passed. If the user fails to get to the end of the phase, the app tells the patient that this is wrong and asks the patient to follow the guidelines. If the user passes the phase successfully, the app considers this phase as passed and tells the user the following guideline. If the user successfully passes all the phases, the app increments the repetition counter by one. When the number of repetitions performed is equal to the number of repetitions selected earlier, the app raises a message that tells the patient that the exercise was performed entirely.

6 CONCLUSIONS

This paper was partially inspired by the "Move Mirror" experiment by Google [26]. We took as inspiration how they compute the similarity between two poses by using weighted distance [25]. While "Move mirror" computes a similarity score between two poses and returns the most similar images it found. The novelty aspect of our project lies in the possibility of assisting the user in real-time by providing the needed guidance for the correct execution of the exercise.

The following sub-sections describe the answers to the research questions of this paper.

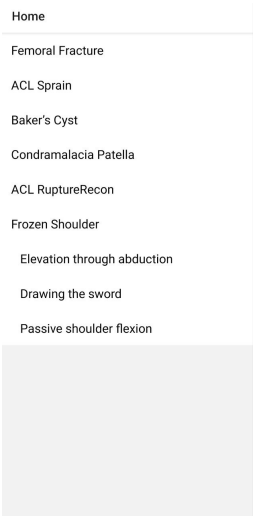


Fig. 15. Home page of the app

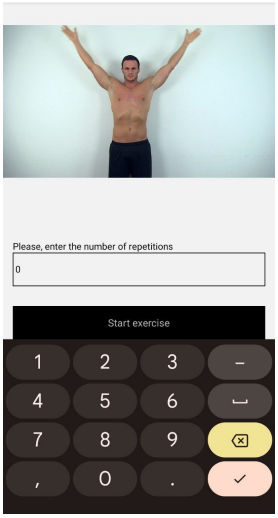


Fig. 16. Exercise page of the app

6.1 Answering RQ1

The first research question was: "How to track the rehabilitation exercises in real-time with a low-resource device?". To answer this question, we searched all other internet body pose models to support low-resource devices. After thorough research, we found BlazePose, the fastest pose estimation model for low-resource devices. It allowed us to estimate the human pose while performing a rehabilitation exercise by returning the position of joints in real time.

6.2 Answering RQ2

The second research question: "How to guide the patient during the exercise?". To answer this question, we processed all the reference images and extracted the coordinates for each keypoint associated with each phase of the exercise. Then, we used the weighted distance technique to calculate the matching coefficient between the current pose vector and the reference vector. If the patient successfully passes all the phases, the app counts the repetition as correct. Otherwise, if the patient fails to pass a phase, the app raises an error, telling the user to follow the exercise guidelines.

6.3 Answering RQ3

The third research question: "How to improve the existing keypoint topology to increase the impact of neck pain exercises?" was answered by finding a category of exercises that the current topology cannot assess. Next, we defined the five needed keypoints, found related images, and annotated them. Afterwards, we fine-tuned the pre-trained model with our new custom dataset. Finally, we trained and tested it with a sample of images (see Fig. 14).

7 FUTURE WORK

Due to a limited time frame, the research goals were relatively small. Nonetheless, there is room for improvement. Namely, in the future version of the app, we plan to include the feature to generate a report at the end of the exercise performance, which will show the mistakes that were made during the exercise execution, with a comment on how to improve it next time. It can be implemented by analysing the angles of the joints involved in the exercise and comparing those to the reference video by using Dynamic time warping (DTW) to compare those two and finding the errors in the execution. Implementing this will transform the app's current version from a guiding app to a correcting one.

Another idea worth implementing in the future is integrating an aufmented reality (AR) trainer into the app. Although we have integrated ARCore in the current version, the display was getting slower, destroying the user experience. In the future, we think we will be able to render a 3D object in the app to help the user perform the exercises. Lastly, we also plan to implement a tracking feature into the app, where the therapist may monitor the patient's progress through the entire rehabilitation program.

REFERENCES

[1] Marco Rogante, Mauro Grigioni, Daniele Cordella, and Claudia Giacomozzi. Ten years of telerehabilitation: A literature overview of technologies and clinical applications, 2010.
[2] Francisca Rosique, Fernando Losilla, and Pedro J. Navarro. Applying vision-based pose estimation in a telerehabilitation application. *Applied Sciences (Switzerland)*, 11(19), 2021.
[3] Jurgen Broeren, Ann Björkdahl, and Martin Rydmark. Virtual reality and haptics as an assessment device in the postacute phase after stroke. *Cyberpsychology and Behavior*, 5(3):207–211, 2002.
[4] Portia E. Taylor, Gustavo J.M. Almeida, Takeo Kanade, and Jessica K. Hodgins. Classifying human motion quality for knee osteoarthritis using accelerometers. *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, (August):339–343, 2010.
[5] L. Enrique Sucar, Roger Luis, Ron Leder, Jorge Hernández, and Israel Sánchez. Gesture therapy: A vision-based system for upper extremity stroke rehabilitation. *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, (June 2014):3690–3693, 2010.
[6] David Anton, Alfredo Goni, Arantza Illarramendi, Juan Jose Torres-Unda, and Jesus Seco. KiReS: A Kinect-based telerehabilitation system. *2013 IEEE 15th International Conference on e-Health Networking, Applications and Services, Healthcom 2013*, (May 2015):444–448, 2013.
[7] Tomasz Hachaj and Marek R. Ogiela. Rule-based approach to recognizing human body poses and gestures in real time. *Multimedia Systems*, 20(1):81–99, 2014.

[8] Adeline Paiement, Lili Tao, Sion Hannuna, Massimo Camplani, Dima Damen, and Majid Mirmehdi. Online quality assessment of human movement from skeleton data. *BMVC 2014 - Proceedings of the British Machine Vision Conference 2014*, pages 1–12, 2014.

[9] Ming-chun Huang, Jason J Liu, Wenyao Xu, Nabil Alshurafa, Xiaoyi Zhang, and Majid Sarrafzadeh. On Bed Rehabilitation Exercises. 18(2):411–418, 2014.

[10] M. Capecci, M. G. Ceravolo, F. F. Ferracuti, S. Iarlori, S. Longhi, L. Romeo, S. N. Russi, and F. Verdini. Accuracy evaluation of the Kinect v2 sensor during dynamic movements in a rehabilitation scenario. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2016-Octob(March):5409–5412, 2016.

[11] Vakanski A, Ferguson JM, and Lee S. Mathematical Modeling and Evaluation of Human Motions in Physical Therapy Using Mixture Density Neural Networks. *Journal of Physiotherapy & Physical Rehabilitation*, 01(04):1–10, 2016.

[12] Ben Crabbe, Adeline Paiement, Sion Hannuna, and Majid Mirmehdi. Skeleton-Free Body Pose Estimation from Depth Images for Movement Analysis. *Proceedings of the IEEE International Conference on Computer Vision*, 2016-Febru(December 2015):312–320, 2016.

[13] Elham Saraee, Saurabh Singh, Kathryn Hendron, Mingxin Zheng, Ajjen Joshi, Terry Ellis, and Margrit Betke. ExerciseCheck: Remote monitoring and evaluation platform for home based physical therapy. *ACM International Conference Proceeding Series*, Part F1285:87–90, 2017.

[14] Lynne V. Gauthier, Chelsea Kane, Alexandra Borstad, Nancy Strahl, Gitendra Uswatte, Edward Taub, David Morris, Alli Hall, Melissa Arakelian, and Victor Mark. Video Game Rehabilitation for Outpatient Stroke (VIGoROUS): Protocol for a multi-center comparative effectiveness trial of in-home gamified constraint-induced movement therapy for rehabilitation of chronic upper extremity hemiparesis. *BMC Neurology*, 17(1):1–19, 2017.

[15] Wan-wen Liao, Sandy McCombe Waller, and Jill Whitall. Kinect-based individualized upper extremity rehabilitation is effective and feasible for individuals with stroke using a transition from clinic to home protocol. *Cogent Medicine*, 5(1):1428038, 2018.

[16] Yalin Liao, Aleksandar Vakanski, Min Xian, David Paul, and Russell Baker. A review of computational approaches for evaluation of rehabilitation exercises. *Computers in Biology and Medicine*, 119:1–29, 2020.

[17] Talal Alatiah and Chen Chen. Recognizing Exercises and Counting Repetitions in Real Time. pages 1–13, 2020.

[18] Meera Radhakrishnan, Darshana Rathnayake, Ong Koon Han, Inseok Hwang, and Archan Misra. ERICA: Enabling real-time mistake detection & corrective feedback for free-weights exercises. *SenSys 2020 - Proceedings of the 2020 18th ACM Conference on Embedded Networked Sensor Systems*, pages 558–571, 2020.

[19] Steven Chen and Richard R. Yang. Pose Trainer: Correcting Exercise Posture using Pose Estimation. 2020.

[20] Swakshar Deb, Md Fokhrul Islam, Shafin Rahman, and Sejuti Rahman. Graph Convolutional Networks for Assessment of Physical Rehabilitation Exercises. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 30:410–419, 2022.

[21] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. BlazePose: On-device Real-time Body Pose tracking. 2020.

[22] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):740–755, 2014.

[23] T. Malone, T. A. Blackburn, and L. A. Wallace. Knee rehabilitation. *Physical Therapy*, 60(12):1602–1609, 1980.

[24] Rehabilitation Exercises – Stables Therapy Centre.

[25] Personlab Person, Pose Estimation, George Papandreou, Tyler Zhu, Liang-chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model.

[26] Move Mirror: An AI Experiment with Pose Estimation in the Browser using TensorFlow.js — The TensorFlow Blog.

[27] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. 2019.

[28] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3686–3693, 2014.

[29] Justin Brooks. COCO Annotator, 2019.

[30] Ying Huang, Bin Sun, Haipeng Kan, Jiankai Zhuang, and Zengchang Qin. Follow me up sports: New benchmark for 2d human keypoint recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11859 LNCS:110–121, 2019.