TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Artjom Protski 206075IAIB
Danila Romanov 205861IAIB

Human Motion Capture and Analysis Component for Mobile Robotics Platform

Bachelor's thesis

Supervisor: Sven Nõmm
Ph.D
Client: Gert Kanter
Ph.D

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Artjom Protski 206075IAIB
Danila Romanov 205861IAIB

Inimeste jämemotoorika salvestamise ja analüüsimise tarkvarakomponent liikurroboti jaoks

Bakalaureusetöö

Juhendaja: Sven Nõmm
Doktorikraad
Klient: Gert Kanter
Doktorikraad

Tallinn 2023

Authors' declaration of originality

We hereby certify that we are the sole authors of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Authors: Artjom Protski, Danila Romanov

22.05.2023

Abstract

Gross motor tests have been used to diagnose and assess the severity of neurodegenerative disorders for more than a century. Such tests usually include walking, sitting, standing, etc. and are visually assessed by the human practitioner. Advances in motion capture technologies have sparked interest in developing computer-aided systems to support the diagnosis of such diseases or estimate their progress. One of the main obstacles to wider usage of such systems is that either a special environment is required for the motion capture or system setup, and usage may be time-consuming and process also requiring two operators.

The aim of this thesis was to create a software solution for a mobile robot to capture human motion and recognise human poses for possible future improvement with artificial intelligence to use in the medical field to diagnose patients with motor diseases. To achieve these goals following techniques have been used: pose detection with MediaPipe technology, work with the Jetson Nano computers and optimization of framework for them, 2D point triangulation with the DLT method, visualization of recorded movements with Matplotlib library.

As a result of the thesis, software with the ability of capturing and recording human movements and representing recorded motions using stickman model in 3 dimensions was developed.

This thesis is written in English and is 61 pages long, including 8 chapters, 22 figures and 3 tables.

Annotatsioon

Inimeste jämemotoorika salvestamise ja analüüsimise tarkvarakomponent liikurroboti jaoks

Jämemotoorseid teste on kasutatud neurodegeneratiivsete haiguste diagnoosimiseks ja raskusastme hindamiseks juba üle sajandi. Sellised testid hõlmavad tavaliselt kõndimist, istumist, seismist jne ja neid hindab visuaalselt arst. Edusammud liikumisjälgimise tehnoloogias on tekitanud huvi arvutipõhiste süsteemide arendamise vastu, et toetada selliste haiguste diagnoosimist või hinnata nende kulgu. Üks peamisi takistusi selliste süsteemide laiemale kasutamisele on see, et liikumisandmete salvestamiseks või süsteemi seadistamiseks on vaja spetsiaalset keskkonda ning kasutamine võib olla aeganõudev ja nõuda kahte operaatorit.

Käesoleva lõputöö eesmärk oli luua mobiilse roboti jaoks tarkvaralahendus inimese liikumise jäädvustamiseks ja inimese pooside äratundmiseks, et seda tulevikus tehisintellekti abil täiustada ja kasutada meditsiinivaldkonnas liikumishäiretega patsientide diagnoosimiseks. Nende eesmärkide saavutamiseks on kasutatud järgmisi meetodeid: poseerimise tuvastamine MediaPipe'i tehnoloogia abil, töö Jetson Nano arvutitega ja raamistiku optimeerimine nende jaoks, 2D-punktide triangulatsioon DLT-meetodiga, salvestatud liikumiste visualiseerimine Matplotlib'i raamatukoguga.

Selle tulemusena loodi tarkvaralahendus, mis suudab tuvastada inimese poose, klassifitseerida neid ja visualiseerida jäädvustatud liigutusi virtuaalses 3D-keskkonnas. Tarkvara sisaldab kasutajaliidest, mille abil saab kasutaja alustada salvestamist ja salvestatud liigutuste taasesitamist 3D-visualiseerimisega, mille puhul on võimalik sisse lülitada kaadripõhine vaade. Tarkvara saab paigaldada robotiplatvormile ja see töötab selle kaamerate abil, millega see on varustatud.

Käesolev töö katab täielikult autorite etteantud nõuded.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 61 leheküljel, 8 peatükki, 22 joonist, 3 tabelit.

List of abbreviations and terms

2D	2 dimensional
3D	3 dimensional
AI	Artificial intelligence
ARM	Advanced RISC machines
CPU	Central processing unit
CSV	Comma-separated values
DDS	Data distribution service
DLT	Direct linear transformation
GPU	Graphical processing unit
k-NN	K nearest neighbours algorithm
mm	Millimeters
MP4	Coding format for digital audio and video
PAF	Part affinity fields
RISC	Reduced instruction set computer
ROS (2)	Robot Operating System (2)
SD card	Non-volatile flash memory card format
SVD	Single value decomposition
UI	User interface

Table of contents

List of figures

List of tables

1 Introduction

Until recently, the medical community had skeptical views on the integration of computer systems for diagnostics of neurological diseases. However, recently the interest in these solutions has risen abruptly within the community. Ongoing solutions cannot claim to have mobility or need long and tedious process to prepare the equipment ready, that fact alone puts restrictions on equipment usage what makes it unprofitable in the eyes of practitioners in the long run [1].

The goal of this thesis was the development of software that is capable of being launched on the mobile robotic platform Iaso to capture human motions and record those motions as a computer 3D model for the ability of later diagnostics of model regarding the presence of neurological diseases.

This solution can become a good addition to the diagnostics of neurological diseases. As an example, for existing diagnostics of Parkinson's disease, the test named "get up and go test". The "get up and go test" requires patients to stand up from a chair, walk a short distance, turn around, return, and sit down again [2]. The software will be able to record patient movements during the diagnostics and later the generated model can be used in the diagnosis process.

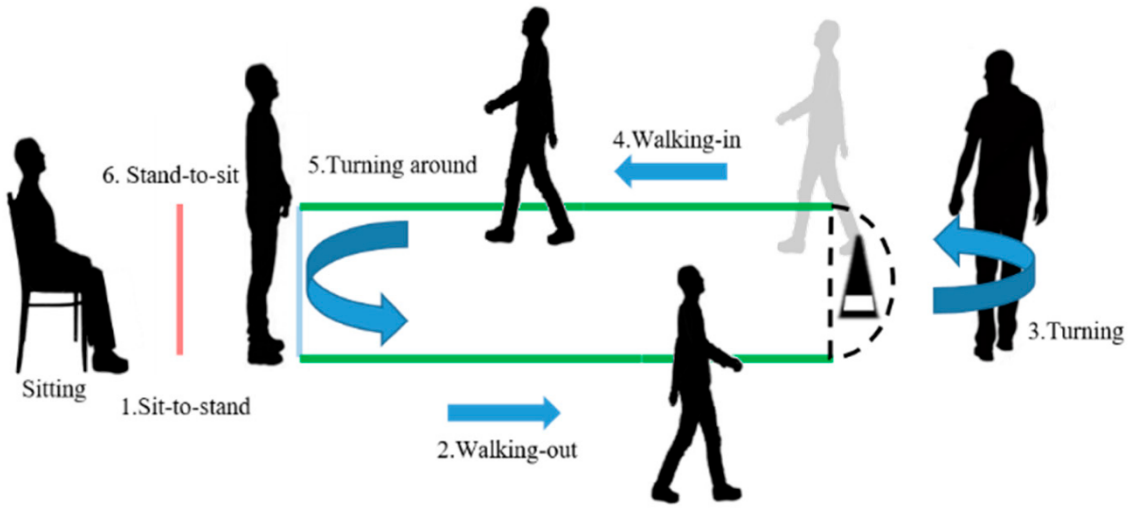


Figure 1. Get up and go test [3].

Although the software was developed with the goal of being used as an additional aid to physicians, its capabilities do not stop there. The software can be used standalone as a PC application in other fields of life where it is important to track the body position, for example, fitness, to count exercise repetition and assess correctness.

1.1 Problem Statement

The proposed project aimed to develop the human motion capture and analysis component for the mobile robotic platform.

Gross motor tests have been used to diagnose and assess the severity of neurodegenerative disorders for more than a century [4]. Such tests usually include walking, sitting, standing, etc. and are visually assessed by the human practitioner. One of the tests is named get up and go and its results can show the differences in motion of patients with Parkinson disease and without [5]. Advances in motion capture technologies have sparked interest in the development of computer-aided systems to support the diagnosis of such diseases or estimate their progress [1]. One of the main obstacles to wider usage of such systems is that either a special environment is required for the motion capture or system setup, and usage may be time-consuming and process also requiring two operators. To answer this concern, a semi-autonomous platform here and later known as Iaso robot has been developed at the Institute of Software Science. The Iaso robot has the ability to move in the room and position itself to the given location; it is equipped with two cameras to capture human motions at different angles. The present project aimed to improve the existing capabilities of the Iaso robot in the area of human motion capture and processing.

The following demands have been specified by the client.

1. Determine the type of posture (sitting, standing) necessary for automatic test segmentation.

2. Provide basic interface functionalities to start and stop the recording.

3. The software answering the above mentioned demands should be executable on Jetson Nano platform and communicate with the motion planning application

4. Provide basic playback functionality for the visualisation of motion records.

The authors planned to develop the software component that will answer the client's demands. There are two candidates for the position of this component's core: OpenPose [6] or MediaPipe [7] engines. It was decided to use MediaPipe (see **Error! Reference source not found.**).

This work required several skills and knowledge in several fields of computer science, mainly programming and robotics. Because of this, authors have decided to split into 2 roles, where one will focus more on application and user side of the project (Artjom Protski), and another on the robotics (Danila Romanov).

The workflow mostly included practical tests of software to assess the results and tune the logic accordingly, such as optimising the calculations or adding new features. Validation milestones consisted of recognising human and building model that closely represents the movements of said person, model saving with the possibility of future replay, and software recognising what actions does the person perform (for example, standing, sitting). Also, the work included validation for the robot part, such as creating a combined model using two cameras recording from different angles, obstacle detection, and ability to move avoiding them.

Validating the functionality that responds to the first demand series of experiments has been conducted by comparing the posture label acquired through calculations with visual assessment of actual posture.

Functionalities such as starting and stopping are the part of the interface, and here a binary system of validation have been used.

The client has validated the final demand related to the integration with the other components of Iaso.

2 Project Description

The practical part consisted of development of software for semi-autonomous robotic platform, that can detect and record model of person in real time for later usage as a part of other systems, for example as disease diagnostics.

The expected duration of development – 4 months. During this time, the team of 2 people planned to develop software prototype that can work on robotic platform Iaso and with the addition of existing solutions, detect pose of a person, classify the pose and save the results for later usage.

The development process can be split in several parts:

1. Development of software
2. Migration of developed solution on mobile robotic platform Iaso
3. Creation of a prototype user interface to interact with the software that is running on the robotic platform
4. Preparations for showcase of the prototype during different parts of development

The plan (see Appendix 2) for the development of software that is focused on the detection and real-time recording of the human model for later use in other systems. Most of the time, the work was done together in pairs with discussion and analysis of progress to acquire the best result possible. The stages of development are closely connected to the dates of prototype presentation and try to optimize the development process to maximize the quality of a result for the day of presentation.

The highest priority task of this thesis was creation of a software part of the application that can detect and record in real time human model, this process was the most time consuming of overall development. The process included research of existing solutions, its analysis and quality assessment with the development of software that is capable of detecting and recording human model in real time. With this came the development process; according to the possible complexity of the main task and the number of subtasks

that were crucial to achieving the goal, it was decided to show only the software prototype for the first demo, without robot integration. This concept would only show the parts that matter most to the client and grading commission. The development process was carried out in a team, with discussion of received information that was acquired during the investigation of source materials related to development. This decision allowed to make the research and analysis processes of necessary development materials faster; nevertheless, this method was crucial to decisions related to choices of necessary technologies and made the choice of the most suitable technology more likely.

After the first prototype showcase, the main goal that had to be achieved until the second showcase was migration of the developed application to the Iaso mobile robotic platform. This process included research of the robot architecture and research of the practical part that was required to integrate the application to the robot platform, and, lastly, the integration process itself. The process also happened as a teamwork with the discussion of acquired information.

The third development stage included all tasks that were done in previous stages. At this stage, there already existed a rough prototype of the final application that could perform all the necessary tasks that were stated in the requirements of the developed system; however, there also existed several flaws that need to be fixed. The main goal of this stage was correction and improvement of the existing product and, as an addition, the development of a user interface that would be the main component for control of the application in a graphical window. The last part in the development process was creation of different papers and documentations, necessary to the understating of applications by third parties.

3 Project Design

The project included 2 main development processes – development of a software and realization of software working on a robotic platform Iaso. The main areas of responsibility of software are:

1. Human pose (stickman) detection on a frame
2. Saving the data in suitable format for late replay and usage
3. Replay of the results on the screen
4. User interface to use the above functionality

For the migration of the application for usage on the Iaso robot, the architecture of the robot had to be researched and current conventions that took effect regarding software development and methods of migration the application on the robot.

3.1 Human Pose Capture

The goal of this thesis was the development of an application that can work on the Iaso robotic platform. One of the main tasks of the application was human pose (stickman) detection, which was taken from camera, connected to the robot in real time. For real time processing it was required to allocate not more than several milliseconds for frame. Subsequently, the application should be optimized enough and provide the ability to process the video. The model should be detailed enough to provide the ability for late analysis and processing, the application should support the most prominent programming languages and have prominent community. This aspect of the solution and programming language prevalence were important for the simplification of the development process and support of final product. These solutions would be able to simplify the search for new employees and reduce the time necessary to learn the product architecture with the help of the Internet documentation that was necessary in the development process.

At the time of development, there existed several solutions that could meet the application requirement of human pose detection. After the search for available options, 2 candidates were acquired: MediaPipe Pose and OpenPose.

3.2 MediaPipe Pose

In 2020, the solution with the name MediaPipe Pose was released, the main feature of which was lightweight application for human pose detection that is available on mobile devices in real time with the usage of artificial intelligence. It uses the BlazePose and ML Kit Pose Detection API to infer a maximum of 33 keypoints (3D landmarks). It is possible to launch this solution on mobile phone, desktop, or laptop. Now the part of this solution that is responsible for human pose estimation, BlazePose, will be covered.

The BlazePose model, developed by Google, utilizes a two-step detector-tracker ML pipeline [8]. In the case of video, detector launches only on the first frame and finds the zone of person interest inside the frame, after which this zone is used for detection on next frames. If the zone in the video is lost, the detector is launched again for the next frame to find the zone of interest.

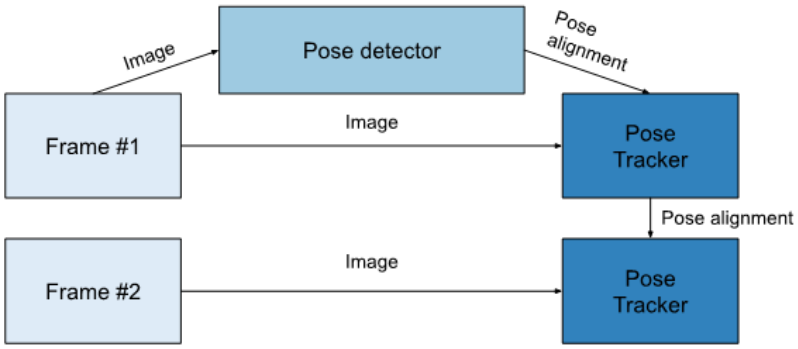


Figure 2. Two-step ML detector tracker pipeline.

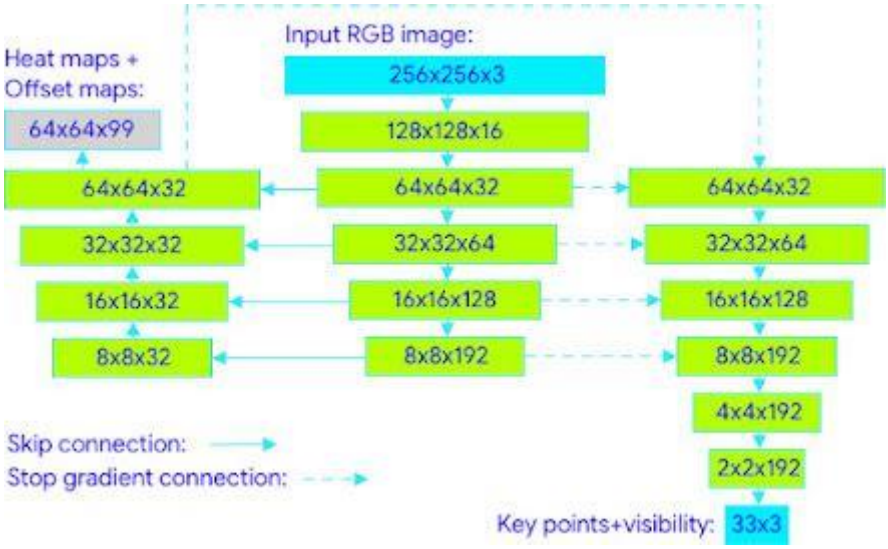
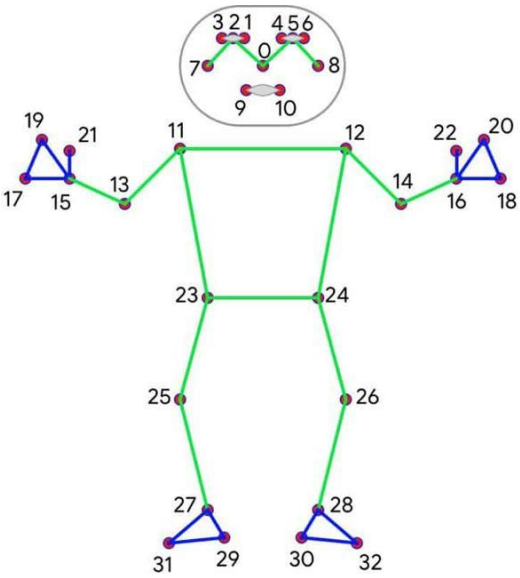


Figure 3. Network architecture [8].



0. nose
1. right eye inner
2. right eye
3. right eye outer
4. left eye inner
5. left eye
6. left eye outer
7. right ear
8. left ear
9. mouth right
10. mouth left
11. right shoulder
12. left shoulder
13. right elbow
14. left elbow
15. right wrist
16. left wrist
17. right pinky knuckle #1
18. left pinky knuckle #1
19. right index knuclke #1
20. left index knuckle #1
21. right thumb knuckle #2
22. left thumb knuckle #2
23. right hip
24. left hip
25. right knee
26. left knee
27. right ankle
28. left ankle
29. right heel
30. left heel
31. right foot index
32. left foot index

Figure 4. MediaPipe topology [8].

BlazePose uses an encoder-decoder network architecture to predict heat maps for all joints, followed by another encoder that regresses directly to the coordinates of all joints [8].

3.3 OpenPose

OpenPose is an open-source library that was made available in 2017 for 2D multi-person human pose estimation. The library uses a nonparametric representation known as Part Affinity Fields (PAFs) to detect the body parts associated with the person in an image. The PAFs describe a list of 2D vector fields and encode both orientation and location of the body [9].

Figure 5. OpenPose demonstration [9].

The architecture encodes global context, allowing a greedy bottom-up parsing step that maintains high accuracy while achieving real time performance, irrespective of the number of people in the image.

3.4 Comparison of MediaPipe Pose and OpenPose

The main criteria for the choice of technology were precision and performance. Application with the main task of 3D pose model detection should have high precision for its later usage in diagnostics of neurological diseases.

During the comparison of two solutions, the advantages and disadvantages of both were found. As a result of the comparison, it was determined that MediaPipe Pose has 2.5 times better performance than OpenPose, which means that the MediaPipe Pose shows more frames per second than its competitor.

This advantage is achieved by using CPU for computations on MediaPipe Pose while OpenPose works on GPU.

It is possible to improve the performance of MediaPipe Pose even further, by launching the solution with GPU support. In that case the performance is improved times 3 and with comparison to OpenPose has 7.5 times more framerate.

Lower performance of OpenPose is counterweighted by its precision, pose detection with MediaPipe Pose is less precise, what shows in “jiggling” of body parts during detection or even false detection.

Data acquired during the work period is also backed by other research that state the same about MediaPipe Pose lower precision than OpenPose but also a system that has more performance.

The performance of OpenPose was assessed as insufficient because of dangers of not getting enough frames with poses for later analysis, that is why it was decided to use less precise, but more reliable system – MediaPipe Pose.

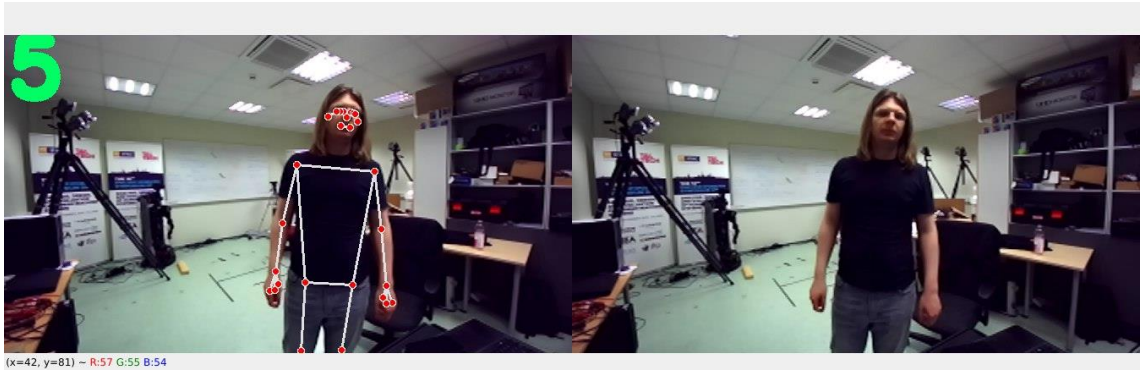


Figure 6. MediaPipe CPU demonstration.

Figure 7. OpenPose GPU demonstration.

The BlazePose keypoints did not always correspond to the respective joint centre. These situations were more likely to occur when the person was moving between frames, the environment had high contrasting backgrounds or additional objects were in the frame, creating a more challenging task for the AI models [8] [10].

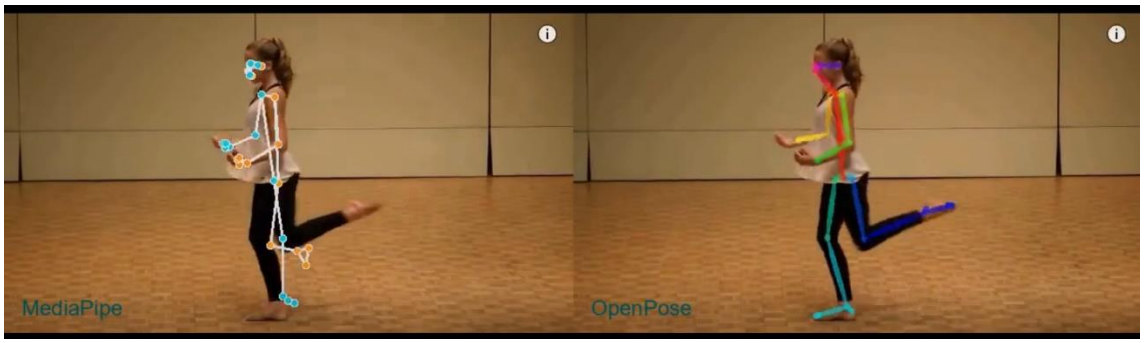


Figure 8. MediaPipe Pose invalid detection [10].

However, the performance of MediaPipe Pose showed superior advantages in comparison with the performance of OpenPose by displaying more frames per second; this comparison was crucial because with low performance (less than 5 frames per second) it is nearly impossible to perform any human analysis in reasonable time. Due to the limitations in the performance of the Jetson Nano computer, it was decided to use less precise but more productive system, MediaPipe Pose [8].

3.5 Saving and Replay of the Received Data

The requirements of the application included the ability to save received data for later processing and analysis and for the ability to represent the model graphically to see recorded movement. This functionality is available for programming in Python language, and this is the main language chosen for the development of application for the mobile robot platform Iaso.

The data acquired from MediaPipe analysis is saved in CSV format and the analysed video feed is saved in MP4 format. The CSV file includes the frame number, Unix time, and 3D coordinates of each keypoint for the current frame. To save the data, Pandas library was used as it is an easy data processing tool, especially for creating and working with CSV files.

3.6 Robot Equipment and Architecture of Software Development for Mobile Robot Platform Iaso

Mobile robot platform Iaso is a metallic construction that has great mobility thanks to the small size of construction and special wheels – omni dimensional wheels, that allow robot to move in all possible directions. The construction is equipped with two Jetson Nano computers and to each computer is connected a ZED 2 camera, which is located on a movable ‘hand’ that can lift itself 90 degrees for additional calculations. The 90-degree angle allows one to record as much data as possible.

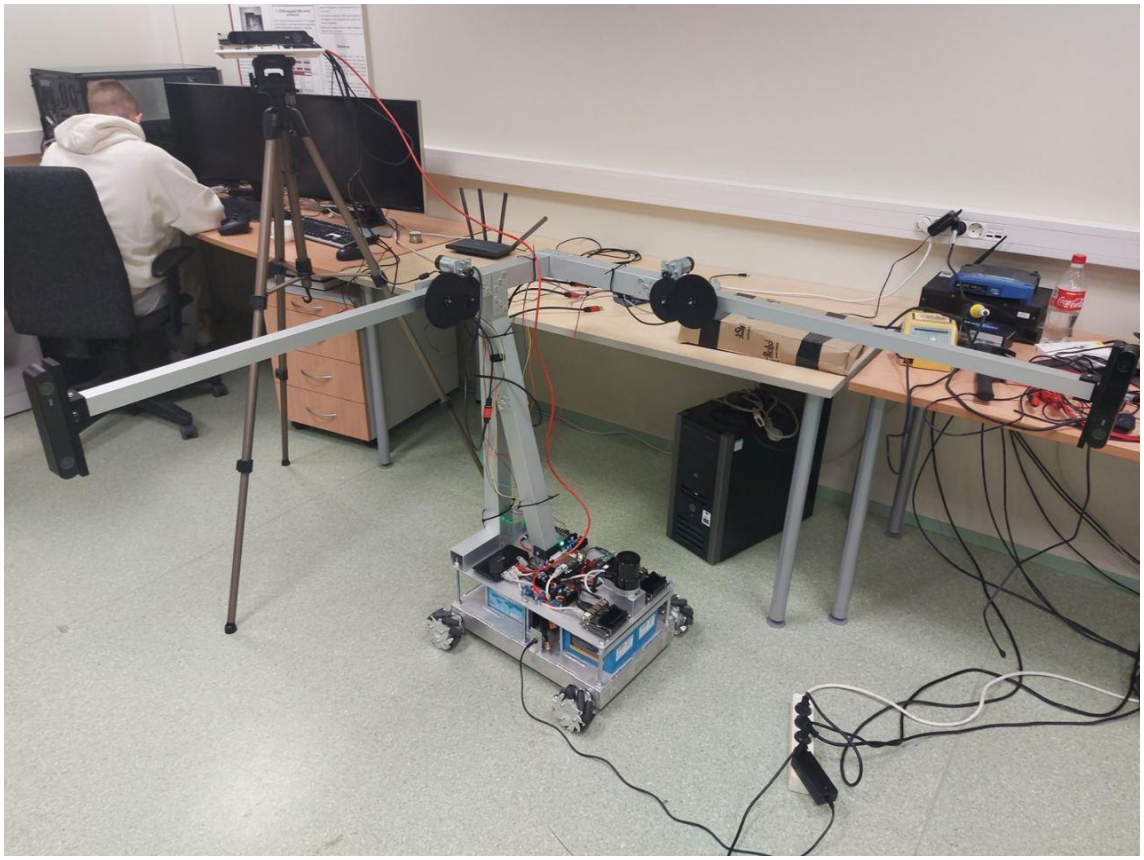


Figure 9. Iaso mobile robotic platform.

3.7 Jetson Nano

The main computing component of the robot is Jetson Nano, which was developed by Nvidia. The advantage of this computer is great performance in comparison to its size (69 mm by 45 mm) thanks to the Maxwell GPU graphical component with 128 cores. The number of cores provides great performance in artificial intelligence computations.

Figure 10. Jetson Nano.

3.8 ZED 2

ZED 2 is a camera that can identify the depth of objects. It became possible due to the usage of 2 lenses that provide binocular vision. Even though this feature has great potential, this thesis will not use it and instead will calculate frame depth with alternative solutions.

Figure 11. ZED 2.

3.9 Robot Operating System 2 (ROS 2)

ROS 2 is a set of libraries designed for robot applications. The use of this technology was justified by the requirements of Iaso robot to the application requirements, specifically to the communication of the different parts, also called nodes.

In ROS 2 there are 3 methods of communication:

1. Topics
2. Services
3. Actions

3.9.1 Topics

Topics are vital elements of the ROS graph that act as a bus for nodes to exchange messages. Topics ensure the ability of nodes to connect to them, after which receive continual updates.

A node may publish data on any number of topics and simultaneously have subscriptions to any number of topics.

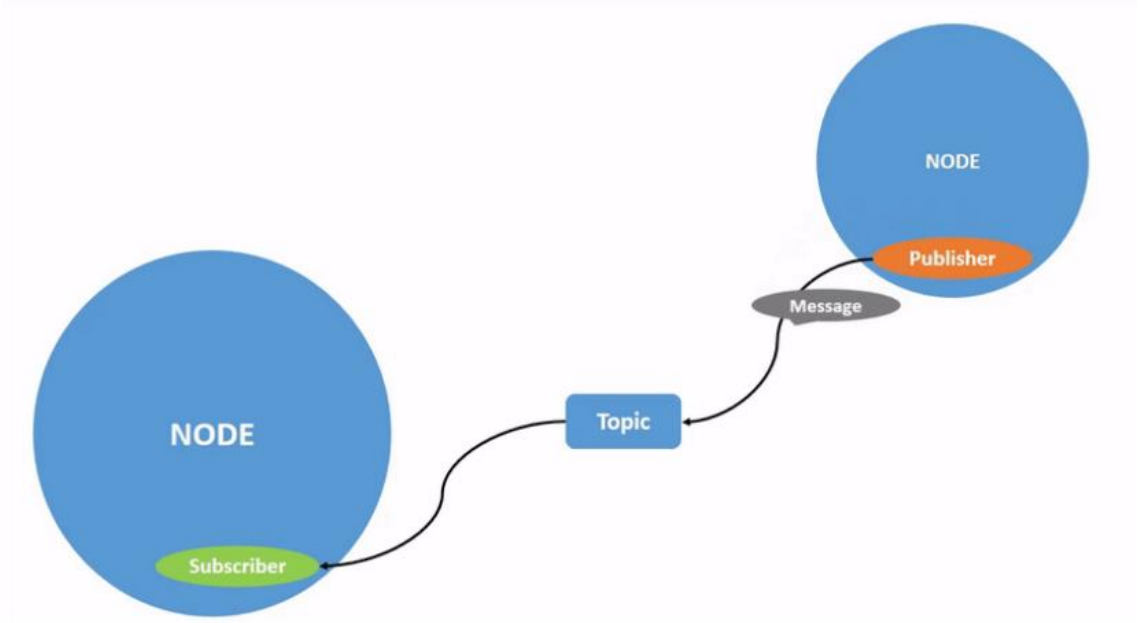


Figure 12. Topics demonstration [11].

3.9.2 Services

Services are based on a call-and-response model versus the publisher-subscriber model of topics. Services provide data when they are specifically called by the service client.

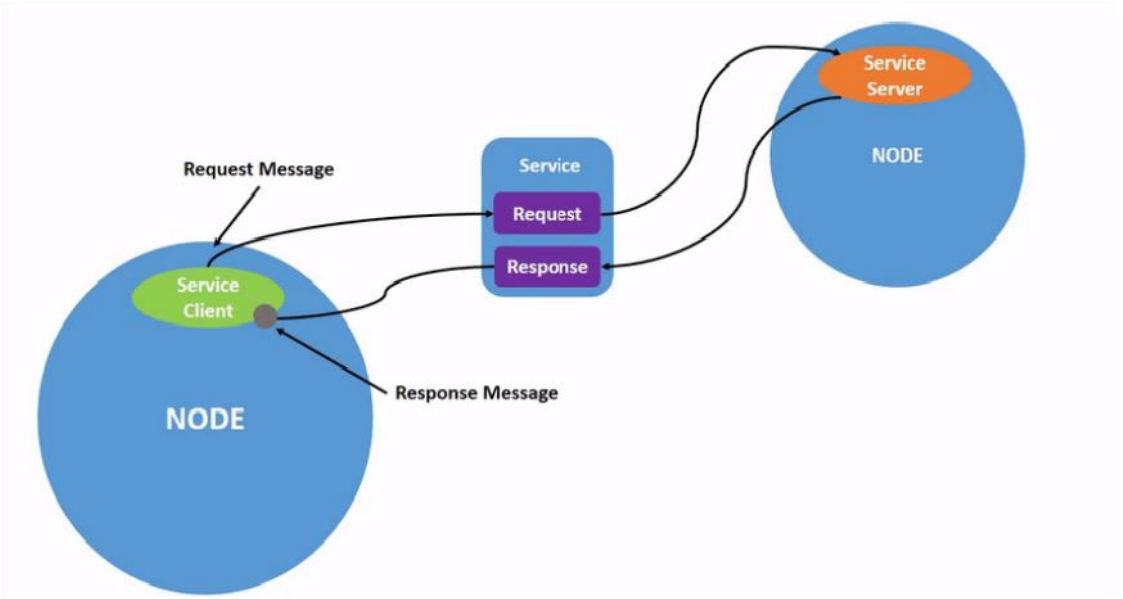


Figure 13. Services demonstration [12].

3.9.3 Actions

Actions consist of 3 parts: a goal, feedback, and a result. It is in a sense a hybrid between topics and services, where functionality is close to the services with the one exception, the process can be stopped during execution. Topics are used for logging of executed processes, which is suited for precise commands having a precise goal.

The communication paradigm in this project will be topics that allow, instead of services and actions, which functionality is close to services, provision of continual data updates; this method is more preferable for motion capture due to the motion capturing process not having end goal or requests, but only provides a stream of frames necessary for data processing.

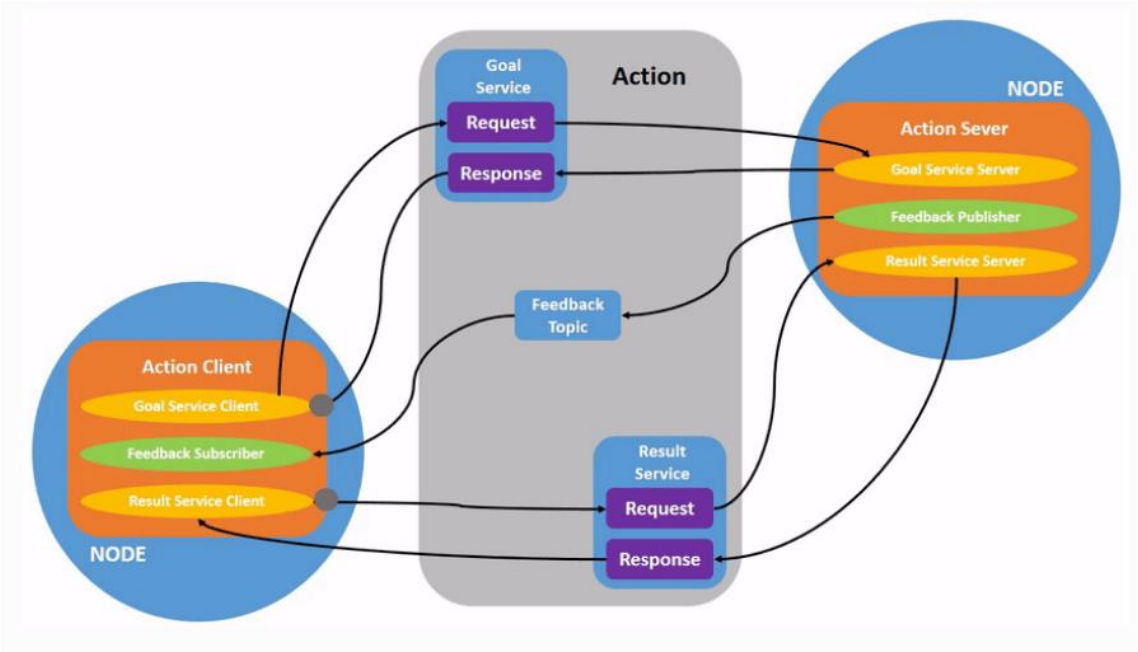


Figure 14. Actions demonstration [13].

3.10 Docker

One of robot architectural requirements is the launch of all applications from Docker container, that allows automatic application rollout.

Docker was designed to simplify the creation, deployment, and execution of applications using containers. Containerization allows the user to run applications in a virtual environment by packaging all necessary elements, such as files, libraries, and other essential components together [14].

4 Implementation

The aim of this thesis was to create a software solution for a mobile robot to capture human motion and recognise human poses for possible future improvement with artificial intelligence to use in the medical field to diagnose patients with motor diseases. It was decided to start with the application first and work with the robot later.

4.1 Setting Up Environment

The first step that had to be done is to set up necessary environment for future development. The work started with empty Jetson Nano and an operational system had to be installed for possible interaction. There were 2 possible images that could be used, the official Ubuntu 18.04 with Jetson Nano installed software, and custom-made Ubuntu 20.04. It was decided to install Ubuntu 20.04 as the version was newer and some of the features may have been unavailable in the older version. The first problem that occurred was that this specific image cut the SD card storage but that could be fixed with additional software that can expand the image to support larger storages. Now the system was ready to install MediaPipe, where the main problems start to occur. MediaPipe has 2 possible ways of working, through CPU or through GPU. Since Jetson Nano comes equipped with the GPU, it would be wise to use it, as it can take an excessive load from the CPU. The problem is that MediaPipe does not have a fast and easy way to install GPU. The normal installation could be done with pip utility but for GPU additional actions should have been made. After several attempts and after accumulating enough information, it was decided to downgrade the operational system version to official Ubuntu 18.04, since all the solutions for current versions did not work. It should be noted that it is possible to install MediaPipe on newer Ubuntu but only the CPU version, and since optimisation is a key component of this application, the GPU variant should be as thoroughly researched as possible. After downgrading, a third-party project was found that showed how MediaPipe with could be installed on Jetson Nano, this solution worked and now it was possible to use and test MediaPipe with full capabilities.

4.2 Software Solution

The development of software was the main goal of the thesis, and the main criteria was to capture human movements and classify the pose on the frame, save the data, and visualise the final result as a 3D model. This process must work with the two cameras installed on Iaso mobile robot platform. Described below is the process of subsystem creation that was necessary for accomplishing all the requirements specified by the client, in the chronological order of development.

4.2.1 Pose detection with 1 camera

First, a small example with MediaPipe Pose library was built that could only detect human poses with one camera. It was used as a base to build on and learn about the library capabilities at the same time. At this stage, work was done with 1 Jetson Nano and 1 ZED 2 camera in a horizontal position. Firstly, the camera feed is read using the OpenCV library, then the frame data is sent to MediaPipe Pose for processing. After processing, the updated frame with body landmarks is shown on the screen in real time. So, the person can move, and at the same time, their pose will be detected on the screen. When application is stopped the video feed is saved in an MP4 format for later replay.

4.2.2 Classification

The next step was to create the pose classifier. The pose classifier was made using MediaPipe Pose methods for machine learning. It is given test data of different states (like sitting, staying) that it learns to detect. K-nearest neighbours (k-NN) algorithm is used to determine the most likely state. The algorithm determines the state on the closest samples in the training set. Several steps had to be taken to build a classifier.

1. Collect image samples of the states and run pose prediction.
2. Convert pose landmarks to the format suitable for k-NN algorithm and form a training set.
3. Perform the classification using camera feed.

To build a classifier as precise as possible, around a few hundred samples are needed for each state, covering different camera angles, lighting, and body position.

4.2.3 Pose detection with 2 cameras

Now the work was done with 2 ZED 2 cameras that the Iaso robot is equipped with but connected to development Jetson Nano computer. Firstly, the camera feed is read using OpenCV library. Then, the frame size is set, 1280 (width) by 720 (height). The cameras are equipped with 2 lenses on left and right front edges, for the purposes of simplification and more stability of detection only one lens is used, the one that is located on top in the vertical position of camera. The MediaPipe Pose then receives the frame data and scans them for human presence. In case it finds one, the keypoint data is returned (position, body part), which can also be drawn on image or camera feed. MediaPipe Pose has a detection confidence that changes how strictly it analyses the feed for human presence, with the parameter ranging between 0 and 1, being close to 1 meaning near impossibility of finding a human. During the detection user sees 3 windows, 1 for each camera with the landmark points and 1 for classifier window. All of the data displayed on the windows is also recorded in several MP4 files for later replay. After the detection, points data is checked to know what body part it belongs to, because for the 3D visualization it is more efficient to not draw the whole body, but only the necessary parts, limbs and torso. If the points are in this category, they are sent to the DLT method for triangulation and the result then is shaped into arrays and written to text file.

4.2.4 Point Triangulation

Since 3D visualisation requires 2 cameras, there should be a method to combine keypoints detected from the first camera with points from the second one, point triangulation model can help with this problem. The 3D point x is projected onto the camera planes through the focal points O1 and O2 and is displayed on the image as y1 and y2. The problem with the displayed model is that it does not reflect the interference that occur. Different distortions and noise affect the model and figure 3 shows realistic state of measurements. The new task then becomes to estimate position of x as close as possible.

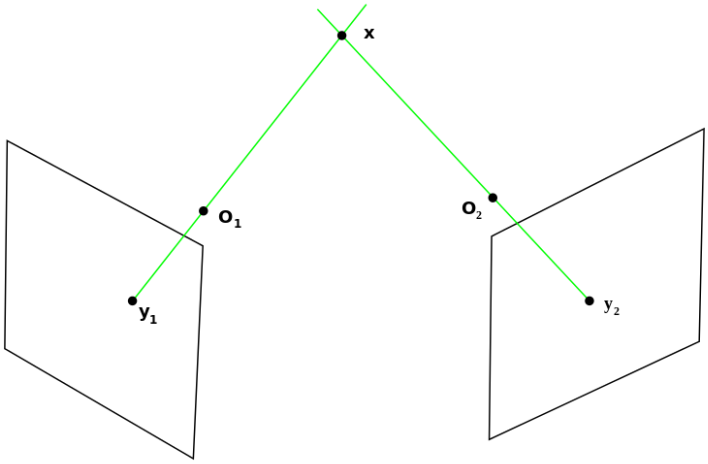


Figure 15. Ideal point triangulation.

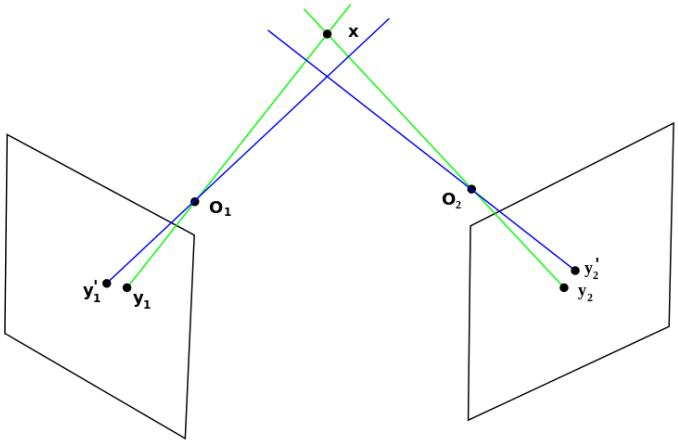


Figure 16. Real-world triangulation.

One possible solution to this problem is to triangulate points with direct linear transformation (DLT). DLT is a method for calculating point triangulation and equation (1) shows its basic form.

$$x = PX \tag{1}$$

Where x is a 2D point, P is a camera matrix and X – estimated 3D point. Several steps must be taken before triangulating points, as DLT method requires individual matrix of each camera, they can be acquired by prior calibration of cameras. Calibration is a process to find the intrinsic and extrinsic parameters of cameras. Intrinsic parameters are the

parameters unique to each camera and they include focal length and optical centres, which together form camera matrix that describes mapping of 3D points in the world to the 2D.

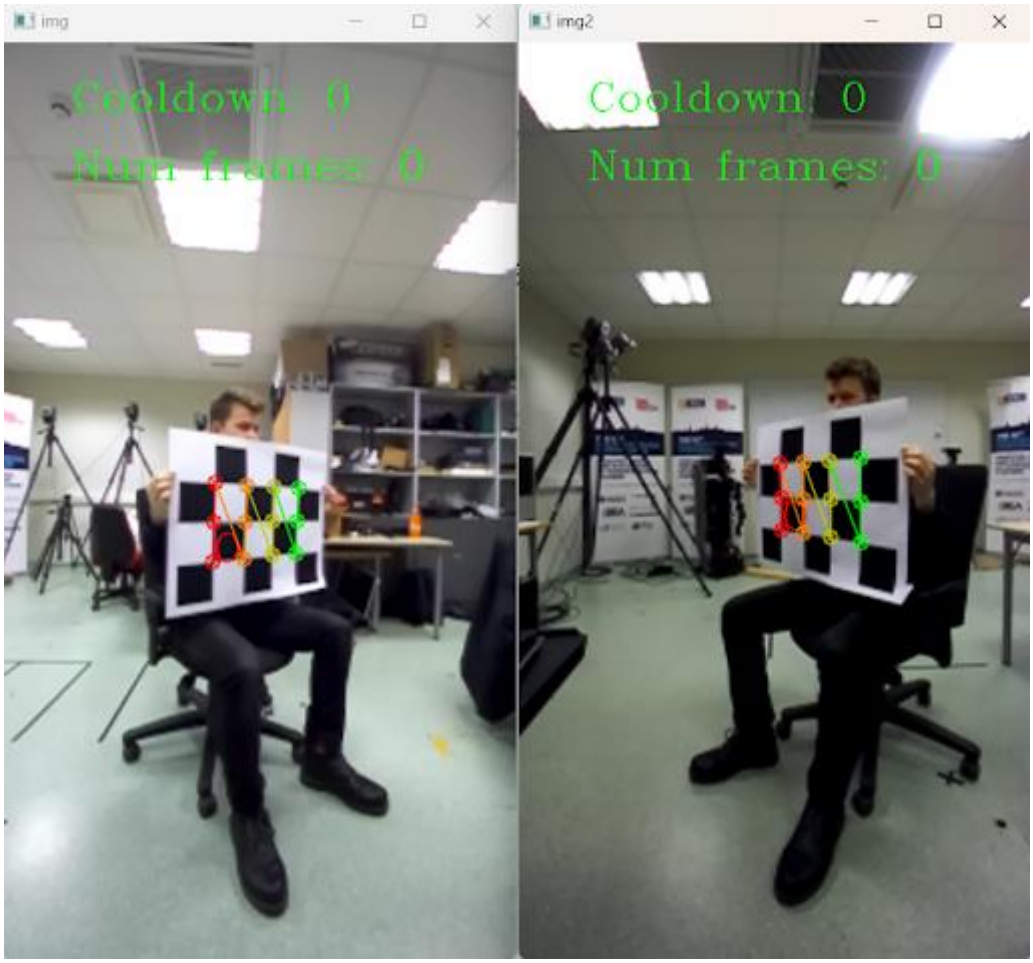


Figure 17. Calibration process.

This information is used to remove distortion that appears due to the specifics of camera lens. Extrinsic parameters include rotation and translation matrixes and correspond to translation of a 3D point to 2D in the image. The distortions must be corrected first and to solve this problem well defined calibration pattern is used (in this case chessboard). The calibration logic finds the relative positions of the square corners and because it is provided with the specifics of a real-world chessboard (board size, individual tile), it can solve the system for the distortion coefficients [15]. Now, it comes down to the DLT method. The system of 2 equations will have to be solved (1 for each camera), but the problem is that there is usually no solution that satisfies both constraints.

The equation (1) changes to equation (2).

$$x = \alpha PX \tag{2}$$

The 2D coordinate x from homogenous becomes non-homogenous. The ray created by the new equation will have the same direction and the only difference will be in the scale factor α.

Now the matrix from equation (3) is converted to linear system, scale factor, and system is solved with single-value decomposition (SVD).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3}$$

The idea of SVD is that the cross product of the two vectors of same direction is equal to zero (that equality allows the removal of scale factor). In this case equality (4) is SVD.

$$x \times PX = 0 \tag{4}$$

Going through equations (5-9) will give a system of 2 equations that is not enough to find 3 variables, but since this system only accounts for one camera, the second one should be added as well.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} [X] \tag{5}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} p_1^T X \\ p_2^T X \\ p_3^T X \end{bmatrix} = \begin{bmatrix} \mathrm{yp}_3^T X - p_2^T X \\ p_1^T X - x p_3^T X \\ x p_2^T X - y P_1^T X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} y P_3^T X - p_2^T X \\ p_1^T X - x p_3^T X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} y P_3^T - p_2^T \\ p_1^T - x p_3^T \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{9}$$

Equation (10) shows matrix with combine parameters from both cameras and by solving it, 3D point X is acquired [16].

$$\begin{bmatrix} yp_3^T - p_2^T \\ p_1^T - xp_3^T \\ y'p'^T_3 - p'^T_2 \\ p'^T_1 - x'p'^T_3 \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (10)$$

4.2.5 3D visualisation

Matplotlib library is used for visualizations in Python. In this case, the library gets the triangulated points and draws them on a 3-dimensional axis, while also adding limb connections between the points. By giving the points of each frame into sets and providing them one after another, frame-by-frame animation is achieved.

Figure 18. Pose visualisation made with Matplotlib.

4.3 Migration to Robot with ROS2

The usage of ROS2 technology is justified by the Iaso robot architecture that requires it for application development. Due to the application not being developed with the ROS2 architecture in mind, the task was to change (migrate) the application for its usage together with ROS2 and evidently, together with the robot. ROS2 is a set of libraries that were created for use in robot platform applications. For development purposes, the only necessary information was the one that concerns basics and concepts of technology and types of communication of different parts of application between each other. All necessary information was acquired through the official ROS2 documentation (version Galactic) [17].

4.3.1 Choosing ROS2 version

Another question was how to choose the correct ROS2 version for application migration. Currently it had 8 versions [18]. The development process was carried out on separate Jetson Nano, after which the migration of ready application to the robot was taking place to keep other systems that already work on a robot safe. However, the developed application had to have the ability to communicate with the other systems to create a united environment.

Iaso robot platform uses ROS2 version Galactic, during the first installation of the ROS2 for the application was used the latest version Humble. After trying to communicate the application with other systems of the robot, the system did not work correctly. After additional research, it was decided to use the identical to the robot's ROS2 version. As was found, ROS2 uses DDS technology to communicate with the help of topics. Different versions have different DDS that may contain different conventions regarding message exchange [19].

4.3.2 Application architecture in ROS2 context

Application migration to Iaso robot started from creating suitable application architecture and after meeting with the client, the scheme was created that can be observed below.

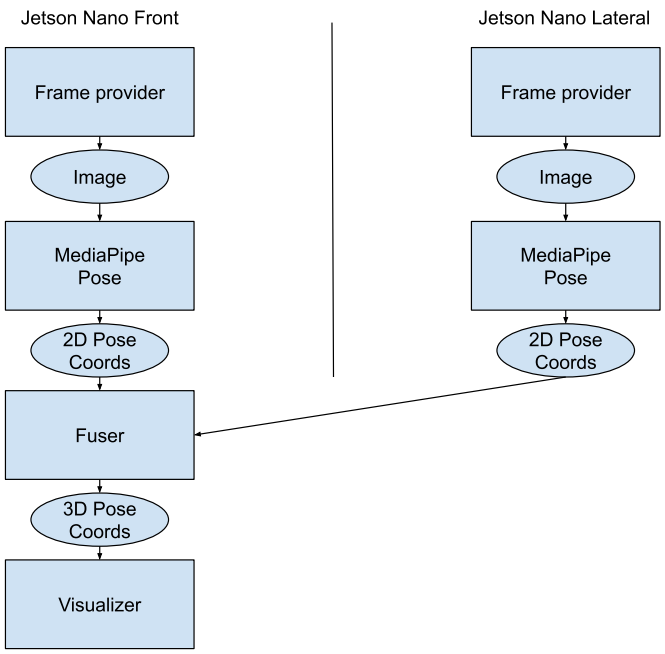


Figure 19. Application structure in ROS.

This architecture considers the usage of two Jetson Nano computers that the robot is equipped with. The two components are duplicated, Frame Provider and MediaPipe. It is necessary because each Jetson Nano is connected to a different camera and each frame acquired is processed separately. Frame provider sends frame to process through Image topic to use in the MediaPipe component. MediaPipe component contains MediaPipe Pose that processes acquired frame and then sends the coordinates of points of human model (stickman). Described process happens in parallel on two cameras and computers to later achieve 3D model with the usage of DLT.

The Fuser launches on one of two Jetson Nano computers and is responsible for 3D human model creation (stickman). It listens to 2 topics that send points' coordinates from 2 cameras, if the received signal has delay lower than described threshold, then the Fuser processes them and sends 3D pose coordinates to Visualizer node. Visualiser node shows the 3D model generated from acquired points in real time. It is possible to turn off the Visualizer to not show the model.

4.3.3 ROS2 message interfaces

To realise the sketched architecture for the application, the ROS2 communication interfaces and how to use them to send the data were researched. Data acquired during

the running of application had to be formatted for ROS2 to send it using the topics. To acquire camera data, the OpenCV library was used, and to translate image data into the format used by ROS2, the CvBridge library was used. This library allows for the translation of the OpenCV frame to the message interface /sensor_msgs/msg/Image in ROS2. Later, it was possible to convert the data from this format to original.

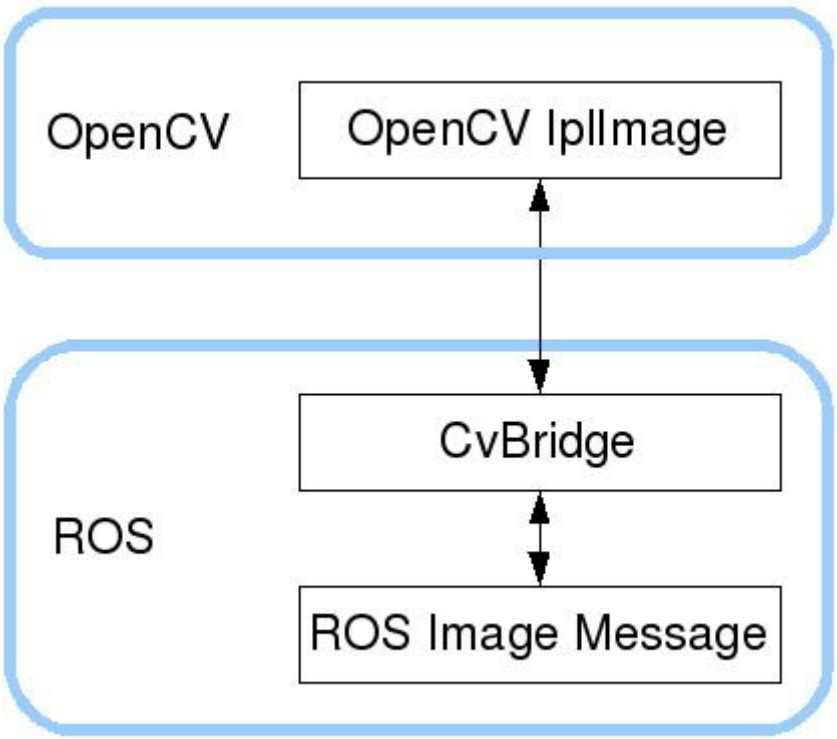


Figure 20. Image translation.

To transfer points data that was acquired MediaPipe Pose estimation it was necessary to create own interface /mediapipe_interfaces/msg/Float64MediaPipeMultiArray to transfer together with the data, information about the size of original frame. This data could be used later to recover original image coordinates and it was needed because the MediaPipe pose normalises human model coordinates.

4.3.4 Application rollout in ROS2 context

The standard launch of components with built-in ROS2 means the launch of each component separately. This approach is unfavorable when we talk about automatic system rollout. To solve this problem, ROS2 capability of writing an additional file was used where the launch sequence is described. After which the file is launched from the terminal, automatically loading all the components.

4.4 Application Containerisation

For automatic rollout of this application on the Iaso robot platform, Docker was used. To launch an environment custom image was used that is generated with the help of a Dockerfile, base for which is Ubuntu 20.04 for ARM system.

4.5 Application User Interface

Future users must communicate with the application without worrying about the inner workings of the application. After analysing several libraries, it was decided to stop on Tkinter library, as it is easy to develop with and provides all the necessary functionality for creation of user interface. The UI allows users to record the video and points necessary for 3D modelling and then later launch that same model for analysis. 3D model has its own interface by Matplotlib library that allows moving the dimensional axes. The Tkinter provides basic functionality, for example events. Events respond to user actions and activate part of the application that it corresponds to.

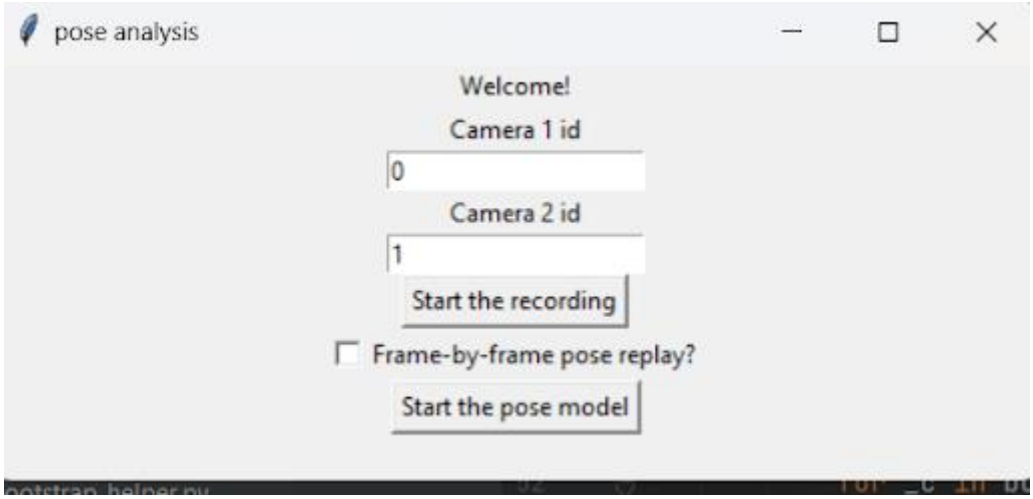
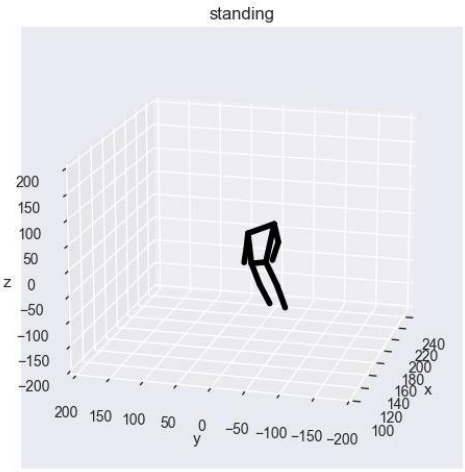


Figure 21. User interface.

5 Validation

Project validation includes check of MediaPipe Pose detection, pose classifier correctness and application software performance. Pose validation is conducted by comparison of calculated result with factual representation. Based on this data, conclusions were made about the precision of pose classifier (see Table 1).



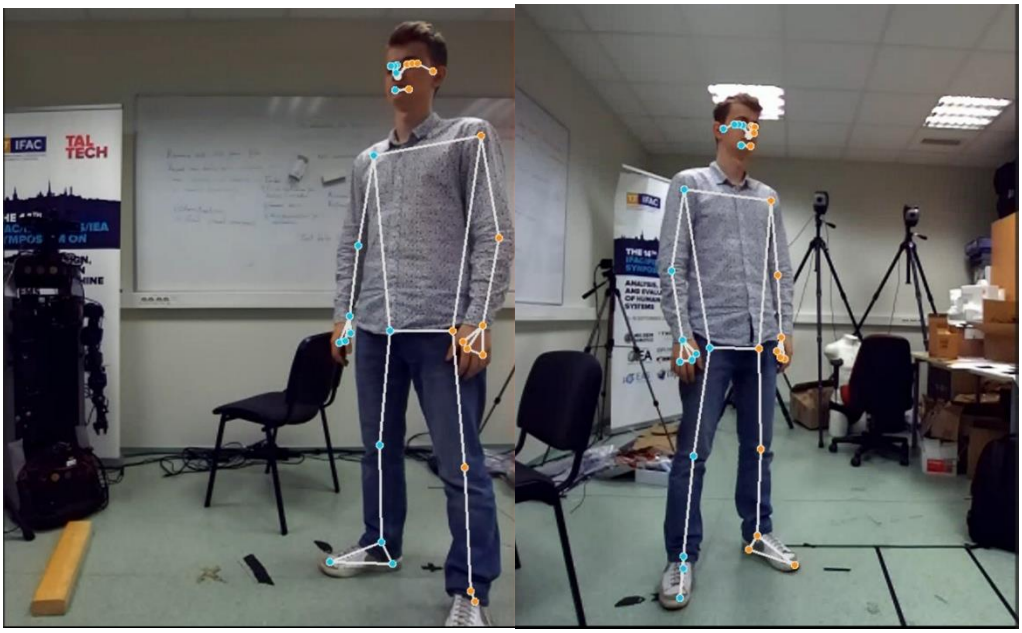


Figure 22. Visualiser work demonstration.

Most of the cases are recorded and displayed precise if not taking into account many of the edge cases where the subject, for example, may make the motions too fast, being face away from cameras and doing unusual, for human standards, poses.

Edge cases include pose detection by analyzing partial data, for example cases where one camera sees a specific body part while another camera does not (see Appendix 5). By checking these cases, software showed excellent result by displaying full body model, the limb coordinates from only 1 camera were enough to display it on the 3D human model.

For example, one of the turn cases where the human turns in the place is the turn while sitting in the chair, after reaching a certain threshold in the body turn angle, the classifier started showing false data regarding the pose. This behaviour could be connected to training data deficiency for correct pose analysis, adding more cases to testing data could fix this issue.

The process of validating the classification was described by analysing the frame delay between the pose classified by software and the factual pose. To determine this delay the frame-by-frame analysis was conducted to calculate the data by visually analyzing the software pose and the human pose in reality.

Firstly, to analyze the data 2 poses were determined that the pose classifier can detect:

1. Sitting – action during which the interior angle of a knee is in the range of 180 to 136 degrees.
2. Standing – action during which the interior angle of a knee is in the range of 135 to 0 degrees.

Test cases were recorded (see Appendix 3) by the results of which several parameters were calculated (see **Error! Reference source not found.**, page **Error! Bookmark not defined.**).

Table 1. Validation data.

Pose change	Minimum delay (in frames)	Maximum delay (in frames)	Mean delay	Standard deviation

Sitting to standing	-2	1	-0.3	0.64
Standing to sitting	0	6	2.65	1.45

This table describes the value of minimal and maximum delay, mean and standard deviation for each case of pose changing – from sitting to standing, from standing to sitting.

The collected data shows that the classifier determines the pose change from sitting to standing relatively fast and precise; in this case the mean delay goes closely to 0 and most of the numbers are close to 0 delay, the data that standard deviation shows.

The classification of sitting to standing change shows a worse result, mean delay is 2.65 frames, and also the difference in the values is much higher in comparison to the first case.

An assumption was made, that this behavior of a classifier can be connected to the small amount of training data and that making the training set bigger can improve the speed and the precision of the classifier.

In addition, anomalies were found during the testing period, during which the classifier started to periodically change certain poses while the subject remained still. This event may often coincide when the subject turns on the place.

The F1 score was counted that measures the overall accuracy. By assigning different poses to states (standing – true positive, sitting – true negative) and counting the frames with correct or incorrect pose classifications precision and recall values were calculated. The F1 score is a harmonic mean of precision and recall.

$$Precision = \frac{True\ positive}{True\ positive + False\ positive} \quad (11)$$

$$Recall = \frac{True\ positive}{True\ positibe + False\ negative} \quad (12)$$

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (13)$$

Table 2. Model precision.

Precision	0,77
Recall	0,97
F1 score	0,86

Another thing that should be mentioned is the speed of the application on Jetson Nano.

Table 3. Application performance on Jetson Nano.

Application type	Frequency (in hertz)
MediaPipe Pose with 2 cameras and visualizer	0.62
MediaPipe Pose with 2 cameras only	2.61
MediaPipe Pose with 1 camera	3.66

6 Related works

While researching the topic, several other projects were found that have already implemented some parts intended for this thesis. Two projects stood out mostly among all: first included human motion capture using 2 cameras with MediaPipe and visualisation with Matplotlib, second one was a web application that captured live feed from 1 camera and showed captured keypoints in real time with the possibility to adjust settings. Most inspiration was taken from the first project since it included detailed explanation of every part of application and provided research that backed up created software [20]. The disadvantage of this project is that it provided an already made solution calibrated for personal cameras of the author, so several parameters had to be changed or recalculated to work with the new setup, especially camera calibration parameters. This solution also only works on desktop and does not have any user interface, so migration to mobile robotic platform Iaso and latter had to be added.

7 Discussion

The developed application that can launch detection and save human 3D model is a great demonstration of the ability to run these types of systems on the Iaso robot platform that has restrictions with computation capabilities. However, the solution still has a list of problems that need to be fixed, the Jetson Nano computers that the robot is equipped with have performance too low for precise 3D pose detection, even though MediaPipe Pose is an excellent option for pose detection because it provides high performance and higher than average precision. The current results that MediaPipe Pose provides on this architecture are insufficient for possible use as an instrument for medical purposes. Better precision in this case shows OpenPose. Even though this option provides more precision, the robot computers do not allow launching this solution with acceptable performance, leaving MediaPipe the only viable solution. What could solve this problem is equipping the robot with more powerful computers, for example, more powerful Jetson models from the same company Nvidia.

The choice of ZED 2 as the main camera for frame capture for later processing with the usage of 3D modelling technology was an excessive choice for the goals of this thesis. The camera capabilities of object depth perception that was achieved due to the presence of two lenses that simulate binocular vision are an excess for this thesis, as depth perception is achieved using other methods. In the application, depth perception is achieved by placing 2 ZED 2 cameras on 90-degree angle that allows adding third dimension to the model by combining data from 2 frames.

Many problems could not have been accounted for before, as an example different version incompatibilities. Some time was also spent on the topic of cameras being placed vertically, since that also makes the frames acquired vertical. For easier analysis of 3D model and possible video replay the frames had to be turned horizontally. This is not a problem when the system is whole and everything is in one place but that becomes a big problem when parts become scattered, for example in Iaso robot with ROS and Docker containers. The current architecture does not provide enough information about cameras,

and the source code has to be changed to know which camera is rotated in what way. Another possible solution can be usage of mathematics and the property of a frame that it is a matrix of numbers, so in theory, by rotating the matrix, the frame will rotate as well.

8 Conclusions

As the result, software solution was created that can detect human poses, classify them, and visualise captured motions in virtual 3D environment. The software contains a user interface with the help of which the user can start the recording and replay the captured motions in 3D visualisation with the capability to turn on frame-by-frame view. The software can be mounted and works on a Iaso robot platform using the cameras it is equipped with.

During the development a software was created that can capture human model using MediaPipe Pose and also visualize 3D human pose in real time using several cameras and DLT method for triangulation. Received 3D pose model can be recorded and saved for later replay. Also, a classifier was developed that can detect 2 types of poses on the frame – standing and sitting, and small user interface with the help of which software behavior can be changed it can be launched.

All the functionality described can be launched on mobile robotic platform Iaso.

Described functionality fully covers the goals set up for the authors of this thesis and, after conducted tests, shows good results in the precision of pose detection and precision of pose classification.

References

[1] A. Krajushkina, S. Nõmm, A. Toomela, K. Medijainen, E. Tamm, M. Vaske, D. Uvarov, H. Kahar, M. Nugis, P. Taba, “Gait Analysis Based Approach for Parkinson’s Disease Modeling with Decision Tree Classifiers,” in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, Oct. 2018, pp. 3720–3725. doi: 10.1109/SMC.2018.00630.

[2] S. Mathias, U. S. Nayak, and B. Isaacs, “Balance in elderly patients: the" get-up and go" test.,” *Arch Phys Med Rehabil*, vol. 67, no. 6, pp. 387–389, 1986.

[3] C.-Y. Hsieh, H.-Y. Huang, K.-C. Liu, K.-H. Chen, S. J.-P. Hsu, and C.-T. Chan, “Subtask Segmentation of Timed Up and Go Test for Mobility Assessment of Perioperative Total Knee Arthroplasty,” *Sensors*, vol. 20, no. 21, 2020, doi: 10.3390/s20216302.

[4] D. O. Stewart A. Factor and W. J. Weiner, *Parkinson’s Disease: Diagnosis and Clinical Management*. Springer Publishing Company, 2007. [Online]. Available: https://books.google.ee/books?id=zUp54Dm-Y7MC

[5] S. Nõmm, A. Toomela, M. Vaske, D. Uvarov, and P. Taba, “An Alternative Approach to Distinguish Movements of Parkinson Disease Patients,” *IFAC-PapersOnLine*, vol. 49, no. 19, pp. 272–276, 2016, doi: https://doi.org/10.1016/j.ifacol.2016.10.546.

[6] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,” Dec. 2018.

[7] “MediaPipe,” 2021. https://developers.google.com/mediapipe (accessed Mar. 02, 2023).

[8] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, “Blazepose: On-device real-time body pose tracking,” *arXiv preprint arXiv:2006.10204*, 2020.

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

[10] M. Reza Soheili, “MediaPipe vs OpenPose (Human Pose Landmark Detection).” Feb. 13, 2023. Accessed: May 22, 2023. [Online]. Available: https://www.youtube.com/watch?v=X471QY9n7dA

[11] Open Robotics, “Understanding topics,” 2023. https://docs.ros.org/en/galactic/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html#understanding-topics (accessed May 22, 2023).

[12] Open Robotics, “Understanding services,” 2023. https://docs.ros.org/en/galactic/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Services/Understanding-ROS2-Services.html (accessed May 22, 2023).

[13] Open Robotics, “Understanding actions,” 2023. https://docs.ros.org/en/galactic/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Actions/Understanding-ROS2-Actions.html (accessed May 22, 2023).

[14] I. Docker, “Docker,” *lnea]. [Junio de 2017]. Disponible en: https://www. docker. com/what-docker*, 2020.

[15] “OpenCv Camera Calibration,” 2023. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (accessed Apr. 15, 2023).

[16] Kris Kitani, “Triangulation,” *Carnegie Mellon University*. Accessed: Apr. 13, 2023. [Online]. Available: http://www.cs.cmu.edu/~16385/s17/Slides/11.4_Triangulation.pdf

[17] Open Robotics, “ROS2 (galactic) documentation,” 2023. https://docs.ros.org/en/galactic/index.html (accessed Apr. 19, 2023).

[18] Open Robotics, “ROS2 distributions,” 2023. https://docs.ros.org/en/humble/Releases.html (accessed Apr. 19, 2023).

[19] T. Foote, “ROS2 different distribution version compatibility.” 2020. Accessed: Apr. 18, 2023. [Online]. Available: https://answers.ros.org/question/341372/can-nodes-from-different-ros-2-distributions-communicate-compatibly/?answer=341385#post-id-341385

[20] T. Batpurev, “bodypose3d,” 2022. https://github.com/TemugeB/bodypose3d (accessed May 19, 2023).

[21] M. Sajith, “How-to-Install-Mediapipe-in-Jetson-Nano,” 2022. https://github.com/Melvinsajith/How-to-Install-Mediapipe-in-Jetson-Nano (accessed May 22, 2023).

[22] G. Montamat, “Pose detection in python with CUDA support,” 2021. https://github.com/google/mediapipe/issues/2041 (accessed May 19, 2023).

[23] L. Zhensheng, “ros2_jetson,” 2023. https://github.com/ZhenshengLee/ros2_jetson (accessed May 19, 2023).

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

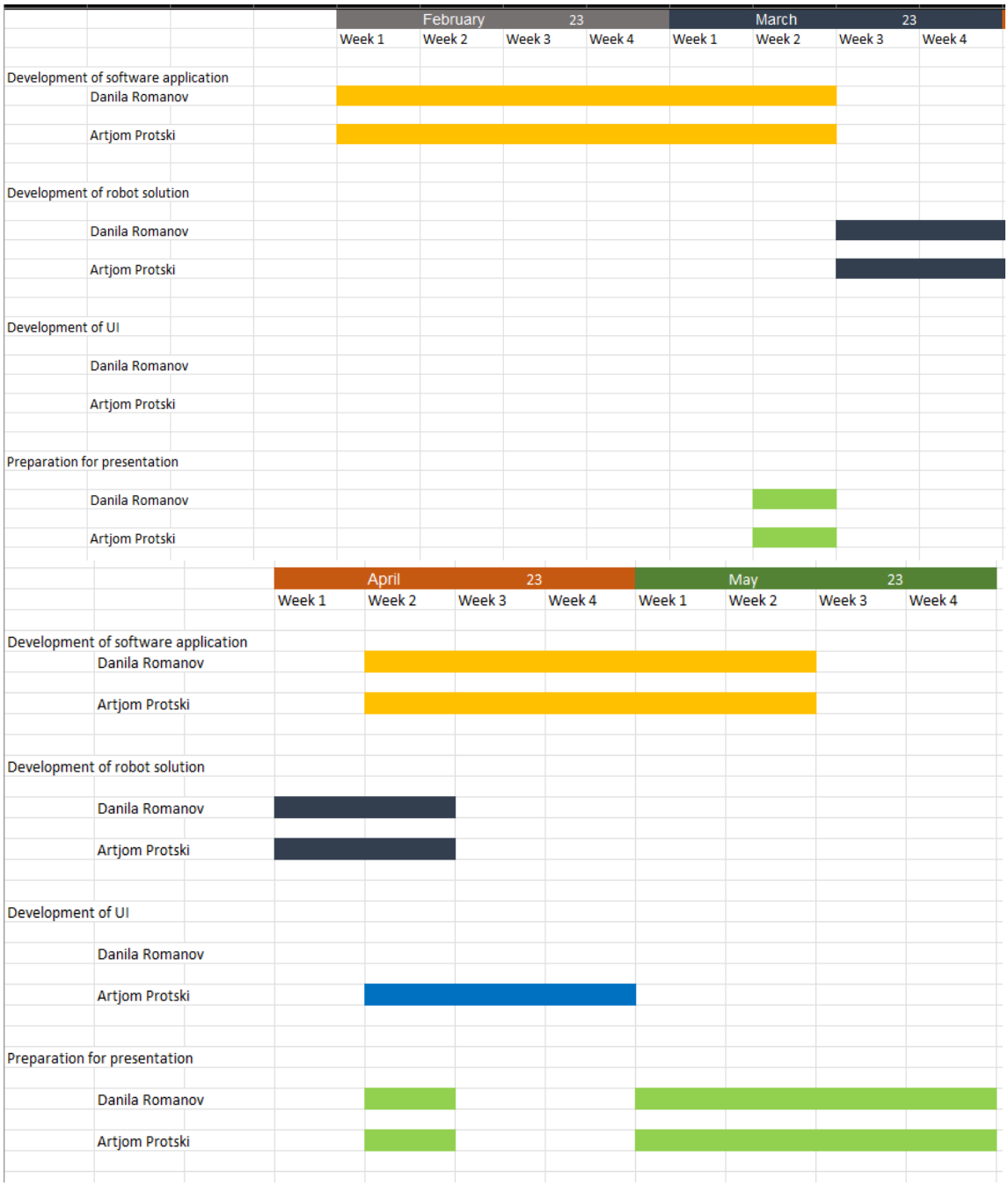
We Artjom Protski, Danila Romanov

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for our thesis “Human Motion Capture And Analysis Component For Mobile Robotic Platform”, supervised by Sven Nõmm
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. We are aware that the authors also retain the rights specified in clause 1 of the non-exclusive licence.
3. We confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

22.05.2023

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Gantt chart



Appendix 3 – Validation data

Case number	Delay
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	1
10	0
11	0
12	0
13	0
14	0
15	-1

16	-1
17	0
18	1
19	-1
20	0
21	0
22	0
23	-1
24	-2
25	-1
26	-1
27	-1
28	1
29	0
30	0
31	-1

Case number	Delay
1	2
2	2
3	3

4	2
5	2
6	3
7	2
8	2
9	2
10	1
11	1
12	0
13	1
14	1
15	1
16	2
17	1
18	3
19	0
20	2
21	5
22	4
23	4
24	6

25	3
26	2
27	5
28	5
29	2
30	2
31	2

Appendix 4 – GPU

During development, it was considered to launch MediaPipe Pose with GPU support. However, many problems occurred with the integration to work together with Python Framework and GPU. At the time of writing this thesis, the official support of Python Framework and GPU was absent (MediaPipe Pose v0.8.9) because of absence of official graphs that support the GPU in Python. The image of MediaPipe Pose v0.8.5 with the GPU support was discovered, however the launching process of the image is difficult due to the condition that it must work with ROS2 because the image demands specific Python version (v3.6) [21].

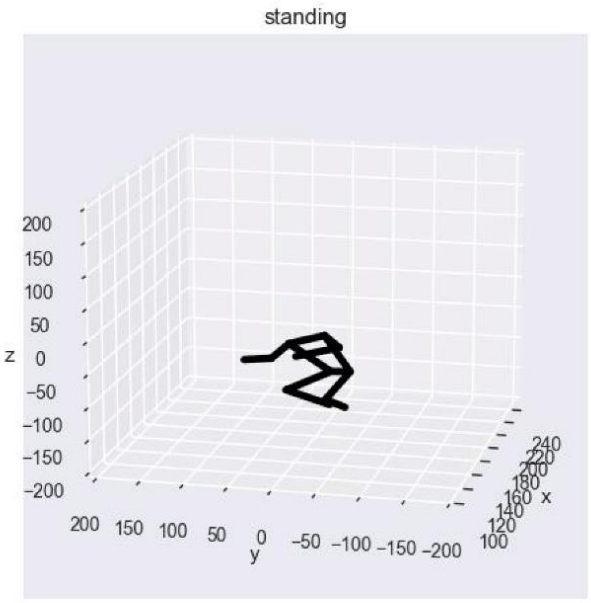
Installation of this specific version for Ubuntu 20.04, the version necessary for ROS2 (Galactic) to work, was problematic, also because the ROS2 version cannot be changed due to the robot being equipped with this specific version. It was assumed that most of the problems are connected to version compatibility problems.

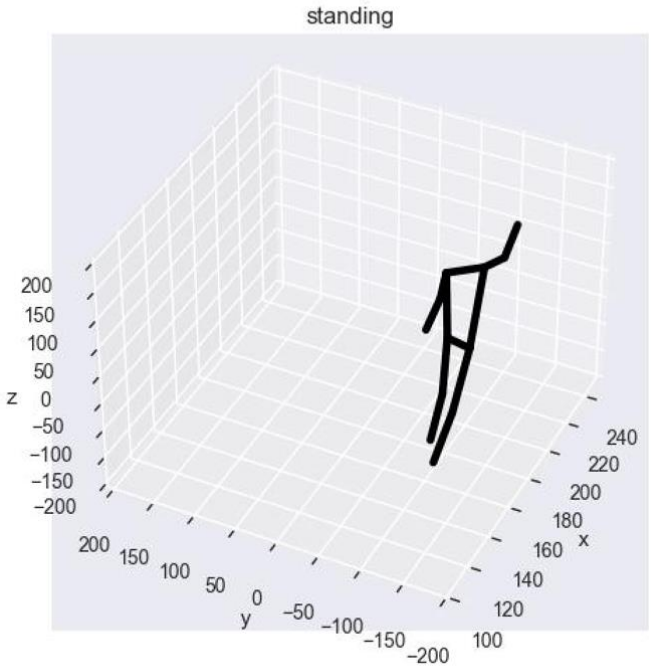
Additionally, several handmade solutions were checked to launch MediaPipe Pose with GPU support. An attempt was made to create graph for GPU support in MediaPipe Pose but it was unsuccessful [22].

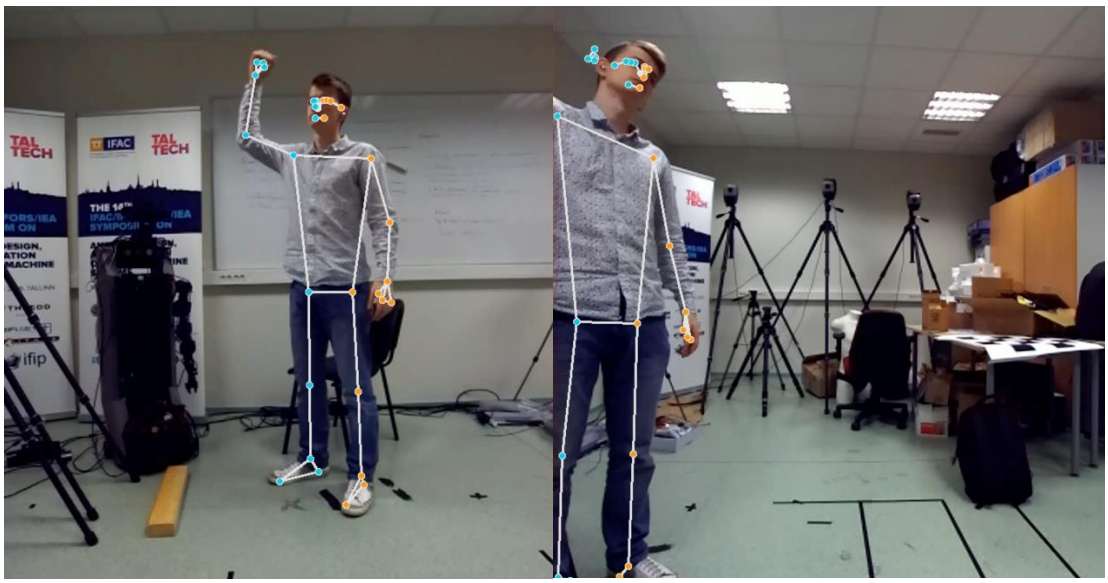
It was also attempted to launch ROS2 Galactic on Ubuntu 18.04 which has no problems with the Python v3.6 instalment [23].

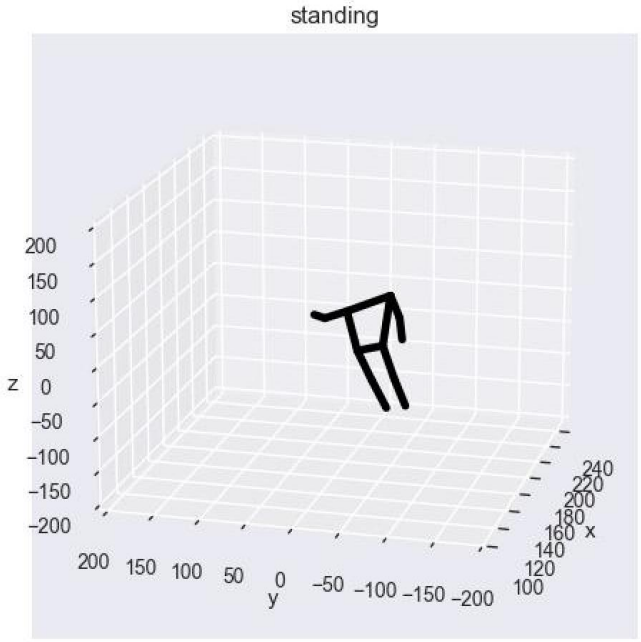
Most of the problems with the compatibility were in some way connected to OpenCV that showed different errors during each of the attempts.

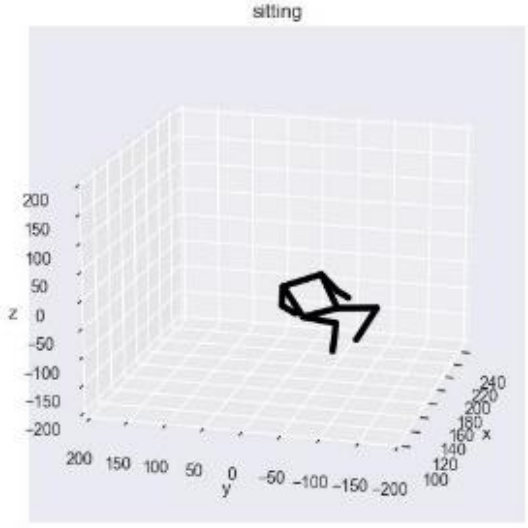
Appendix 5 – Pose estimation edge cases

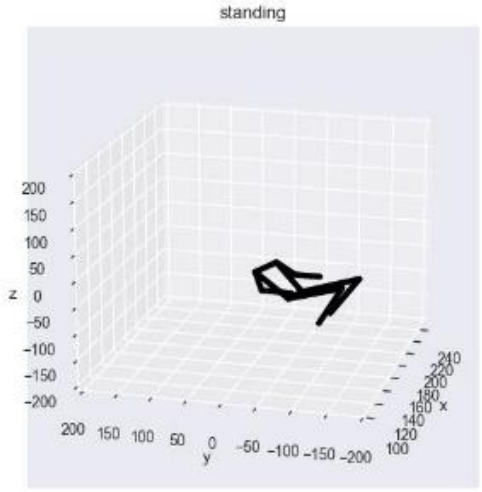


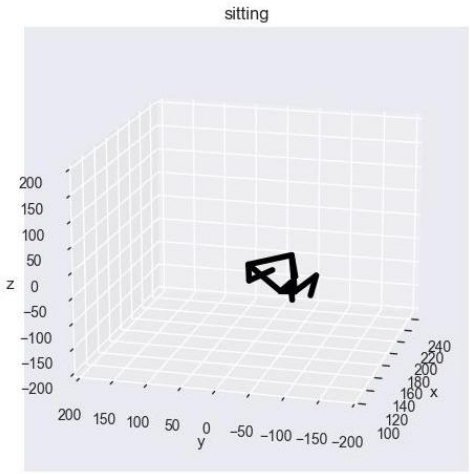

standing
200
150
100
50
z 0
−50
−100
−150
−200
200
150
100
50
0
y
−50
−100
−150
−200
100
120
140
160
180
x
200
220
240

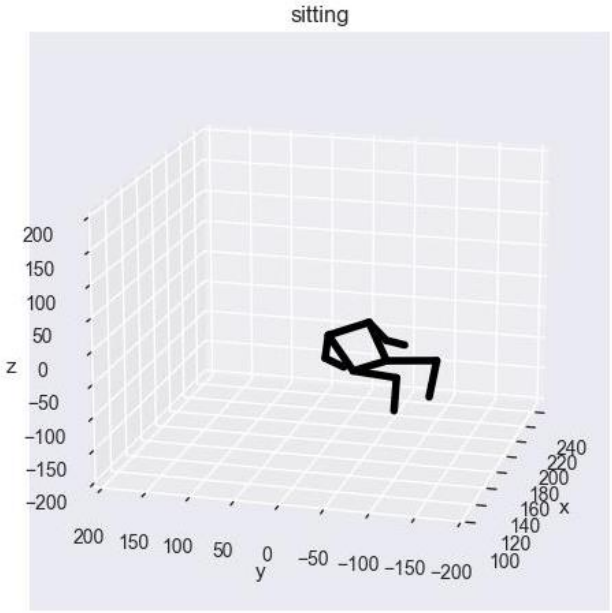


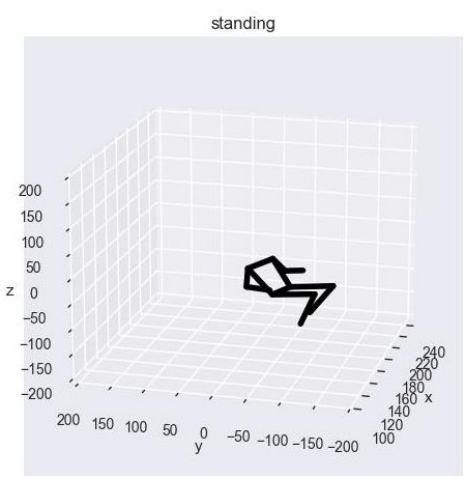

standing
z
200
150
100
50
0
−50
−100
−150
−200
y
200
150
100
50
0
−50
−100
−150
−200
x
240
220
200
180
160
140
120
100

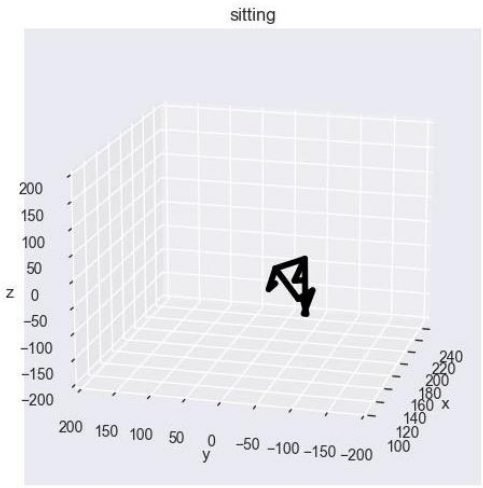

sitting
z
y
x

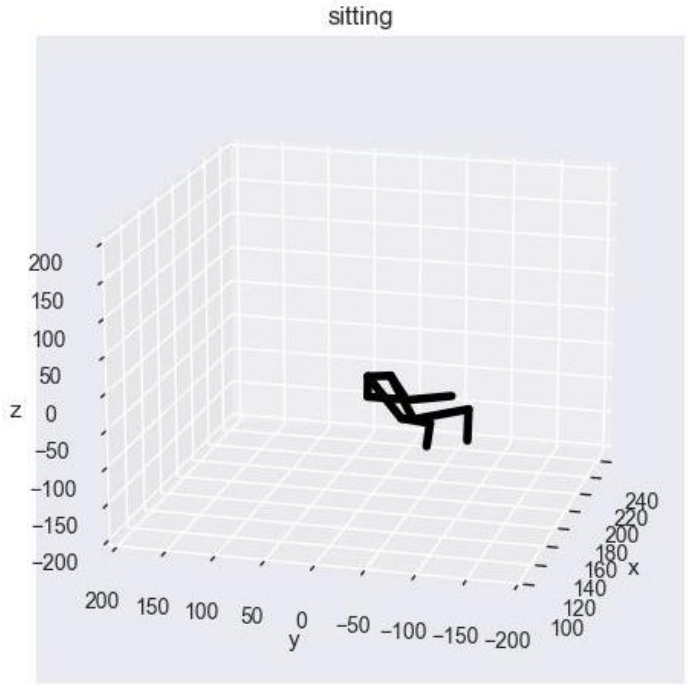

standing
z
y
x

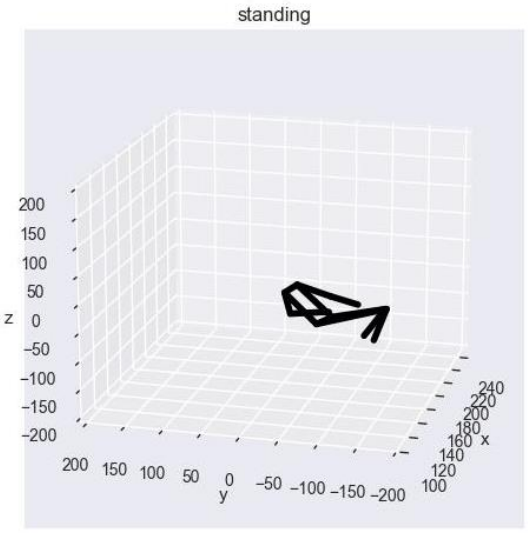

sitting
z
y
x

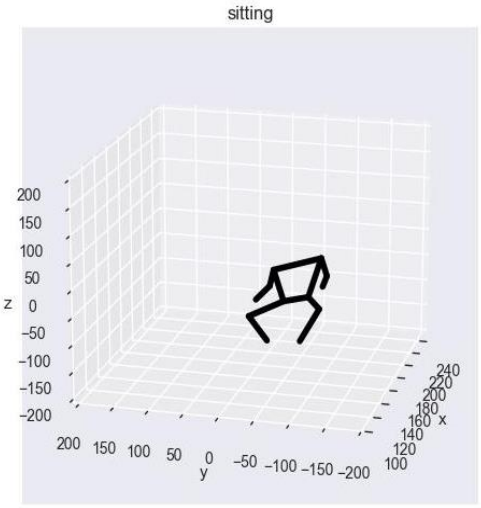

sitting
z
y
x


standing
z
y
x


sitting
z
y
x


sitting
200
150
100
50
z 0
−50
−100
−150
−200
200 150 100 50 0 −50 −100 −150 −200
y
240
220
200
180
160 x
140
120
100


standing
200
150
100
50
z 0
−50
−100
−150
−200
200 150 100 50 0 −50 −100 −150 −200
y
240
220
200
180
160 x
140
120
100


sitting
200
150
100
50
z 0
−50
−100
−150
−200
200 150 100 50 0 −50 −100 −150 −200
y
240
220
200
180
160 x
140
120
100