AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
Wydział Inżynierii Mechanicznej i Robotyki

KATEDRA ROBOTYKI I MECHATRONIKI

Master thesis

Programming the Spot robot in order to obtain unique functionality.
Programowanie robota Spot w celu uzyskania unikalnej funkcjonalności.

Autor: *Dawid Jantosz*
Kierunek studiów: Mechatronic Engineering
Opiekun pracy: *dr hab. inż. Tomasz Buratowski*

Kraków, 2024

List of contents:

Introduction

These days it is almost impossible to imagine worlds without robots. Starting in 1954 with the appearance of the first industrial robot "Unimate", the robot industry has become increasingly influential. With the continuous growth of automatics and robotics the global economy and the quality of life of the average person increases. The quality of work and the safety achieved through the influence of robots is irreplaceable. With the goal of future development we have to impose ourselves greater challenges. The main focus on my thesis was to create new functionality and introduce the possibilities of the Spot robot, created by the company Boston Dynamics. I aimed to make use of its capabilities to implement a function facilitating interaction between humans and the robot, particularly in tracking an outstretched hand.

Spot and similar mobile robots are widely used in industry to automate repetitive, tedious tasks for humans . This approach results in cost savings, and when operating in a hazardous environment, it ensures the protection of the employee’s health. However, there are situations when the human role is irreplaceable and the collaboration between humans and robots becomes problematic. One response to this issue could involve the use of vision systems to detect specific human behaviours, such as outstretched hands, enabling rapid communication without the need for additional equipment.

The first chapter presents types of used mobile robots in the industry. The emphasis was places on presenting the capabilities and specifications of the Spot robot, its applications in industry and research focusing on the robot’s functionality. The second chapter focuses on the presentation of the equipment. The phase of data collection conducted by the robot, the training of a machine learning model, and the Python libraries used were discussed, all of which are key aspects in working with the robot. The last chapter presents data on model training as well as conclusions from the study.

1. Purpose and assumptions of the project.

1.1 The role of mobile robots in industrial automation.

Usage of the robots to automate some tasks has become an increasingly serious issue, drawing attention from both scientists and representatives of companies. The purpose of such action is to increase effectiveness of performed tasks and saving human resources. The process of automation of such repetitive, tedious tasks for humans, has become the main goal enabling the competitiveness the job market. Additional impulse to increase the influence of robotics into the economy of developing countries are demographic factors. The populations of highly developed countries are characterized by an aging society and a birth rate below generational replacement[20].

The usage of the robots is not limited to performing tasks considered unattractive and tedious by humans. In industries requiring sterility and precision, they have become an obvious choice, without which running a business would not be profitable. Such factors caused, after the COVID-19 global pandemic, that strongly limited the economies of many countries, there has been a rapid growth of industries related to robotics and automation. As noted by the International Federation of Robotics (IFR), the robot industry after 2020 is characterized by continuous growth expressed by the annual list of industrial robot installations (Fig.1). The chart shows data from previous years along with predictions for the near future. The conclusion drawn from them is continuous growth, despite the slowdown in the growth of the global economy.

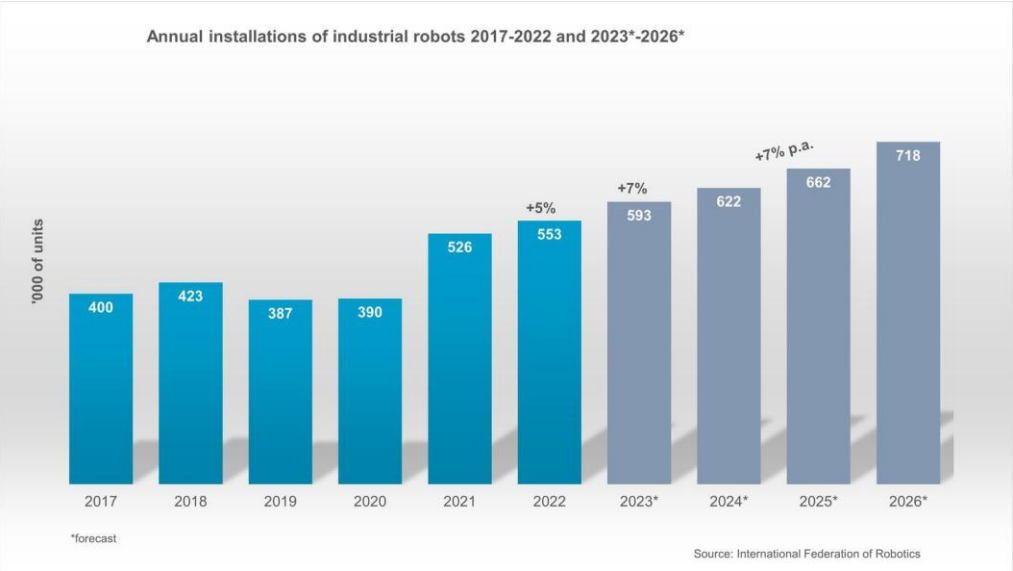

Fig. 1: Annual installations of industrial robots[34].

The upcoming industrial revolution related to artificial intelligence is closely related to the use of robots and their capabilities. Its pillars and name were defined in detail in the European Commission report entitled: "Industry 5.0"[1.]. One of its assumptions is to improve communication between people and the machines. Efficient communication in this area will be an important factor for the industry and the economy.

1.2 Mobile robots

Mobile robots, historically used in industry to transport heavy, inconvenient loads, with the development of technology have become the key to greater automation of the industry. The development of automated guided vehicles (AGV) and the invention of more advanced mobile robot technology, namely autonomous mobile robots (AMR), contributed to this. Although their function is still limited to moving the load to the desired place, with the progress of automation, the number of applications of mobile robots is increasing dramatically. The combination of mobile robots with manipulators, achievements in the field of vision systems, machine learning and artificial intelligence have resulted in a significant improvement in the robots' response to the surrounding environment, which is crucial for the usage of mobile robots in industry.

Within the category of mobile robots, we can distinguish several subcategories. Robots can be divided according to the way they move, such as wheeled robots, tracked robots, legged robots and others.

1.2.1 Wheeled robots

A wheeled robot is a type of mobile robot that uses a system of wheels to move. The basic criterion for classifying a robot in this category is the number of driven and steered wheels. This type of robot is characterized by an easier structure and control model compared to other types. Optimal use requires a flat surface, due to the difficulty in avoiding obstacles involved with rocky terrain or uneven surroundings. Their design can range from simple single-wheeled robots, such as spherical robots, to more complex, multi-wheeled platforms capable of moving in different directions and overcoming obstacles. One of the representatives of the family of wheeled robots is the Jet Propulsion Laboratory Sample Return Rover (SRR) (Fig.2).

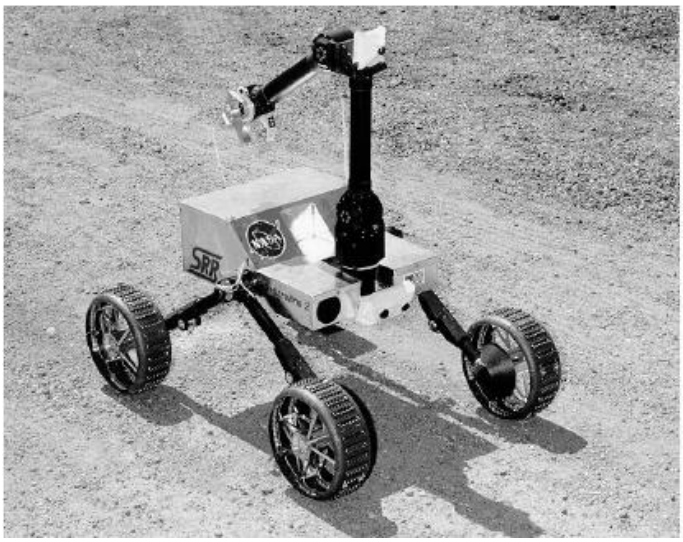

Fig. 2: Jet Propulsion Laboratory Sample Return Rover[21].

1.2.2 Tracked robots

A tracked robot is a type of robot that moves using tracks, similar to some off-road vehicles or tanks. Instead of wheels, the tracks consist of toothed or rubberized belts positioned along the robot's platform, enabling it to navigate challenging surfaces. The higher mobility of tracked vehicles is achieved through greater complexity and a lower coefficient of rolling resistance, making the advantage of such vehicles evident primarily when operating in difficult terrain. One representative of the tracked robot family is the Soryu-C, designed by the company Hibot (Fig.3).

Fig. 3: Soryu-C Hibot[22].

1.2.3 Legged robots.

Another group of robots that is developing dynamically are walking robots, i.e. technical devices designed to perform selected functions similar to the locomotion functions of animals and insects with limbs (vertebrates) or legs (insects)[]. The key feature of these robots is the balance between the robot's stability and its flexibility in movement. The design of these robots is based on complex control algorithms that allow them to maintain balance and moving in a way that is close to natural. An example of a robot that falls into this category is Spot Classic from Boston Dynamics (Fig.4).

Fig. 4: Spot Classic Boston Dynamics[25].

Legged robots can be categorized based on the stability of their movement, the number of legs, and dimensions [36]. A particularly important aspect among the mentioned categories is the ability of robots in this family to maintain stability during motion:

a) Statically Stable:

Statically stable refers to the ability of robots in this category to maintain stability while moving. They are characterized by a significant number of active degrees of freedom, maintaining a constant configuration (posture), with their motion precisely described using kinematic methods. Trajectories of the body and leg movements are predetermined, and precise execution is expected.

b) Quasi-Static Stability:

In comparison to the previous category, robots in this group have fewer active degrees of freedom. Between periods of static stability, there are phases where this stability is lacking. However, the machine does not topple over - dynamic stability is maintained, although just for a limited time.

c) Dynamically Stable:

Robots in this category, possessing from several to several dozen degrees of freedom, stand out due to their constantly changing configuration. The result of these changes is the maintenance of dynamic stability, translating into a smooth progressive movement of the device.

1.2.4 Quadrupeds robot

Quadrupeds robots belong to the category of legged mobile robots, where each robot consists of four legs, which allows it to move efficiently. The development of four-legged mobile robots has accelerated rapidly in recent years, increasing their use in many human assistance applications.

Robots of this type are characterized by lower stability and movement speed compared to wheeled robots, but they are less susceptible to the influence of terrain and have functions that allow you to avoid obstacles. These features allow this robot to be used in more critical conditions. The design makes the balance problem easier to solve, but the use of a large number of actuators results in increased energy demand. One of the newest four-legged robots is Spot from Boston Dynamics (Fig.5).

Fig. 5: Spot Boston Dynamics[24].

1.1 Spot robot

Spot is a quadruped mobile robot created by the American company Boston Dynamics, based in Waltham, Massachusetts. In the early stages of its operation, the company collaborated with the American Systems Corporation under a contract with the Naval Air Warfare Center Training Systems Division (NAWCTSD) to replace training films for aircraft launch operations with interactive 3D computer simulations featuring characters created using DI-Guy software for realistic human simulation. Eventually, the company began producing physical robots, with the first one, BigDog, being a quadruped robot designed for the U.S. military with financial support from the Defense Advanced Research Projects Agency (DARPA).

The thoughtful construction of the Spot robot allows us to use it in various industries for different complex tasks. In its structure, we can distinguish key components such as:

-dimensions
-environment
-energy
-cameras
-connectivity
-payloads

1.3.1 Dimensions

The Boston Dynamics Spot robot has a length of 1100 mm, a width of 500 mm, and a height of 840 mm when standing, and 191 mm when sitting. Its net weight is 32.5 kg. Spot is equipped with 2 actuators in each hip and one actuator in each knee. Its hip joints, labeled as HX and HY, allow specific ranges of motion: +/- 45 degrees from the vertical for HX, enabling internal and external rotation, and +/- 91 degrees with a 50-degree deviation from the vertical for HY, allowing bending/spreading. Additionally, the knees have a range of flexion/extension from 14 to 160 degrees, providing a total of 12

degrees of freedom, 3 for each leg. Thanks to this construction, it is capable of moving at a speed of 1.6 m/s

1.3.2 Environment

Spot has an IP54 protection rating, meaning it is resistant to dust and moderate amounts of water but is not completely waterproof. It can operate in temperatures ranging from -20°C to 45°C and move on terrain with an incline of +/- 30 degrees. It is adapted to work on stairs compliant with American construction standards, where the typical step has a height of 7 inches and a run of 10-11 inches. Spot can overcome obstacles with a maximum height of 300 mm and requires lighting above 2 lux to optimally utilize its capabilities. The robot is designed to operate in environments where people move, so it is equipped with systems for handling difficult terrain. However, users need to be aware of its limitations[16]. Spot is nominally designed to maintain a minimum distance of 7.5 cm, but this functionality should be disabled in certain environments, such as outdoor areas with high grass or vegetation, for optimal use of Spot.

1.3.3 Energy

Energy management is a crucial aspect when working with a mobile robot. Spot is equipped with a battery (Fig. 6), which can be inserted into Spot by placing it on its back, utilizing the 'safe rolling' feature. The Spot battery provides power for about 90 minutes of normal operation and approximately four hours in standby mode (sitting). Heavy loads, high computational activity, and low temperatures will additionally reduce this operational time. The battery's storage and operating temperature range from -30°C to 25°C in storage and from -20°C to 45°C during use. During operation, the battery temperature can reach up to 75°C. The expected service life is 500 cycles to 80% capacity.

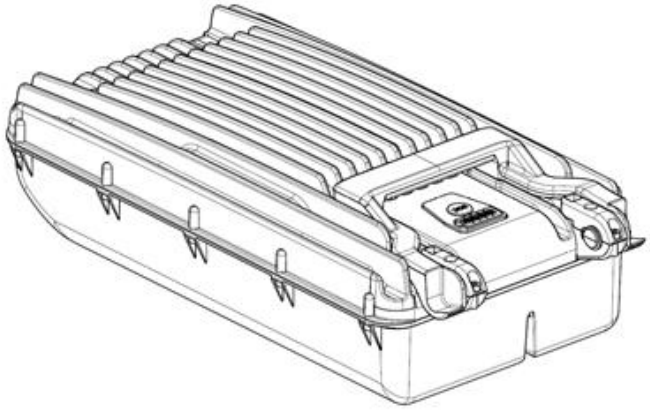

Fig. 6: Spot’s battery[23].

The battery charging is carried out using a dedicated charging system (Fig. 7), consisting of:

- Charging case.
- Power cable, which connects the charging case to the electrical socket (AC) .
- Shore power cable, which connects the charger to the Spot's shore power port for charging batteries or powering computers using shore power.
- Battery charging tray, where the battery is placed by plugging the charger into the socket and turning it on to start the charging process.

Fig. 7: Spot’s charging system[23].

An alternative method of charging Spot is to use a dedicated charging station (Fig. 8). The spot is equipped with functionalities supporting docking, charging and disconnecting when fully charged. While charging, the robot turns off its motors. Charging time varies depending on battery condition and ambient temperature. However, at 25°C, an almost dead battery can be charged to 80% in 50 minutes. The docking station

has dedicated places for fiducial markers. Spot is equipped with the functionality to detect these indicators, which allows you to automate the process of loading Spot without human assistance.

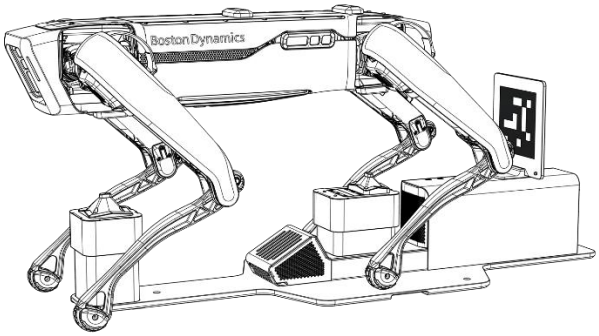

Fig. 8: Spot with a charging station [26].

1.3.4 Cameras

The spot uses different types of cameras for different purposes. It has 5 optical cameras intended for operators, located at the front (left, right), on the sides (left, right) and at the back. The total view for optical cameras is 360 degrees. Additionally, Spot uses 5 sets of depth cameras for the robot's perception and avoiding obstacles. These cameras cover a ~90-degree field of view in each direction, but there are areas of inaccuracy around the robot's hips where the field of view does not overlap.

To optimally use Spot, it is necessary to take full advantage of its ability to detect its surroundings using sensors and cameras in order to efficiently operate in the surrounding terrain and avoid obstacles. To do this, Spot stores information about the environment as a grid-based map, where each grid cell has a parameterized dimension and refers to a specific place in the world [19]. Using this grid of values, Spot is able to assess the dimensions of objects surrounding it, which allows it to move efficiently.

1.3.5 Connectivity

Spot, as a robot designed to operate in diverse environments, is able to use many communication modes. When uploading or downloading files to Spot, a connection via the RJ-45 port is recommended on the back of the robot, which provides a reliable, high-speed communication connection without the need for infrastructure, but limits where the application can be run. An alternative way to connect to the robot is via WiFi (Wireless Fidelity). This connection can be achieved in two ways. The first one is to connect an Access point provided by Spot near the device. The advantage of this connection is the possibility of direct communication without the need for any network infrastructure. The second option is to connect to an existing WiFi network, which allows you to work on a larger area. This option is chosen by plants operating in large spaces.

1.3.6 Payloads

The key aspect of Spot enabling it to work in diverse environments is its adaptive function, utilizing attached payloads. Boston Dynamics provides dedicated payloads such as Spot Arm[2] or Spot Core I/O (Fig. 9), but there is also an option to use equipment designed by other companies after prior configuration and adaptation for Spot. However, when selecting additional payloads for Spot, it must be kept in mind that the maximum load to which Spot is subjected cannot exceed 14 kg, and the number of payloads should not exceed two. An important aspect is the dimensions of the loads applied to the robot. They should be chosen so as not to restrict the robot's movements and not negatively impact the tasks for which the robot will be used.

Fig. 9: Spot Core I/O[10].

1.4 Spot programming

Programming plays a fundamental role in the development of new features for mobile robots. In the case of the Spot robot, using dedicated libraries written in the Python language is a key element of the programming process. Boston Dynamics provides users with an SDK (Software Development Kit)[18], which is a set of programming tools that includes ready-made libraries and a series of examples on which programmers can base their work. This makes the creation of new robot functions more accessible and efficient.

While working with the Spot robot, to use correctly functioning versions of the Python language and compatible libraries, it is recommended to use virtual environments. When working with the Spot robot, Conda[9] is an excellent tool for managing virtual environments. It not only allows the creation of isolated environments but also provides easy management of dependencies, such as libraries for object detection or other tools needed for robot programming.

1.4.1 Base services

Basic services constitute the architectural foundation that is utilized by the rest of the Spot robot's API (Application Programming Interface). These services include authentication, service discovery, and time synchronization. Many applications will leverage these basic services during startup and before using higher-level services such as Robot State, Robot Commands, or Autonomy[4]. These are the foundational elements that provide essential support for subsequent operations, enabling effective utilization of the full functionality offered by the Spot robot's API application interface. Below, I will present key basic services that are fundamental components when working with the robot.

a) Lease

The lease service allows taking control of the robot and maintaining proper communication with its owner. Leasing is essential for issuing commands that control the mobility and movement of the robot, such as powering on or issuing positioning commands. To begin, the lease must be acquired (or accepted) to demonstrate possession of the robot's resources. Subsequently, signals maintaining the lease must be sent throughout the operation to retain control and confirm reliable communication with the lease owner. Ultimately, the lease should be returned to release resources for reuse; however, it can be revoked during operation by another user or in case of a lack of communication signals within a specified time.

It is worth noting that at any given moment, only one person can control the Spot robot. There are two options for obtaining the lease: "acquire lease," which allows gaining access to the robot if no one else currently has access, and "take lease," which enables forcing access to the robot even when it is currently in use by another person. It is also important to emphasize that observing processes performed by Spot, such as taking pictures, does not require using these lease options.

b) E-stop

The e-stop service is based on software pulsation, allowing control over the power of motors and communication with the robot. This function involves confirming the ability to communicate with the robot by sending a special message. The terminology of the service refers to endpoints that must send appropriate messages as part of the registration procedure. Users have the option to determine the level of stopping the robot's actions - from an immediate power cut to no interference with the operation of motors.

c) Time-sync

The time-sync service is used to estimate the difference between the application clock and the Spot's clock, which is crucial because correctly synchronizing time is essential for interpreting sensor data and issuing precise commands to the Spot robot. Time disparity could lead to the incorrect interpretation of new sensor data as coming from the future and could prevent the execution of commands with a specific expiration time, potentially resulting in their rejection by the robot. Therefore, the time-sync service

ensures accurate time alignment between the application and the robot, enabling smooth operation and precise control of the Spot robot.

1.5 Applications of the Spot Robot in Industrial Automation and Inspection

The Spot robot, thanks to its capabilities, offers functionalities for automating both routine and hazardous tasks in various industries and applications. Spot finds particular use in the inspection and patrol industry, where its capabilities are highly desirable. Its ability to navigate precisely, collect data from various sensors, and perform tasks in hard-to-reach places allows for increased efficiency in inspection systems. In the oil and gas industry, where working conditions are often harmful to humans, using the robot to read values from analog, mechanical, and digital indicators becomes an alternative to human operator work. Research conducted at the customer's facility[32] demonstrates a relatively high effectiveness of such robot applications. The route recording and playback feature acts as a safeguard in case of unwanted events or inaccuracies in Spot's performance.

The construction industry is another sector widely utilizing Spot. With a properly adapted payload, the robot can capture 360° images and recordings both indoors and in challenging outdoor terrains. Regularly captured snapshots of construction site progress can be contextualized within construction documentation and used to automate the analysis and reporting of ongoing work using modern artificial intelligence technologies. Spot can create environment maps in offline mode, utilizing Google Cartographer's Simultaneous Localization and Mapping (SLAM) package[1]. This ability is extremely useful for work on construction sites or in hard-to-reach areas.

2. Implementation

This chapter is dedicated to the implementation of the system. In the first subchapter, I will present the equipment used. In the following subchapter, I will explain the available methods of hand detection and the system I applied to my task. In the next subchapter, I will describe the code used to implement the trained model and control Spot's movements. The last subchapter is dedicated to safety issues when working with the robot. Precautionary measures and situations to avoid are presented there.

2.1 Research station

The research station was organized to explore the capabilities of the Spot robot in a specific environment. During the experiments, the Spot robot was used as the main element of our setup. To ensure communication and control of the robot, a Samsung Galaxy Tab 3 tablet was employed, with a dedicated application installed for interacting with the robot. The connection between the tablet and the Spot robot occurred in WiFi access point mode, enabling smooth data transmission and robot control in a small research area measuring 5 meters by 5 meters. Additionally, using this method allowed the visualization of the environment detected by Spot and provided easy access to one of the key functionalities when working with the robot, the e-stop (Fig. 10). In the event of undesired or potentially dangerous situations, this functionality allows for cutting power to the actuators, immobilizing the robot. It is worth noting that a more complex utilization of this system is possible; however, in our case, such application of this functionality is fully sufficient.

Fig. 10: Exemplary image from tablet.

2.2 Hand detection method

In robotics, computer vision is defined as the process of extracting, characterizing, and interpreting information from images of the three-dimensional world. With the latest achievements of artificial intelligence and machine learning, we are able, using a sufficiently large database, to develop our own model that can detect a outstretched hand with high accuracy.

2.2.1 Data collection

The process of capturing images for training machine learning models involves several steps. A crucial aspect is collecting diverse data that reflects various conditions and scenarios the model is intended to learn. In the studied case, an environment with uniform, constant lighting was assumed. The key aspect in data acquisition was the choice of the camera. In the initial phase of the study, a FLIR A70 thermal camera (Fig. 11) was used, allowing image recording in the thermal range.

Fig. 11: Station with camera FLIR A70.

Despite its ability to capture optimal data in the specified conditions(Fig.12), the decision was made to stick with the cameras built into the Spot robot. The reason for this decision was the difficulty in testing the system. Securing an external camera would require creating special covers or mechanisms to prevent damage in the event of the robot tipping over or unwanted contact with the environment.

Fig. 12 :Exemplary image from FLIR A70.

The process of capturing images using the built-in cameras on the robot was conducted with particular attention to utilizing the entire camera set. Controlling the Spot robot with the tablet, images were sequentially collected from individual cameras by changing the robot's position and its own orientation. Additionally, emphasis was placed

on altering the environment where the hand was clearly presented, considering the diversity of scenarios in which these images might be used.

In the context of the image acquisition procedure, a crucial element was the script controlling the actions of the Spot robot and the image acquisition[7]. An equally important aspect of this code was establishing a connection with the Spot robot, authorizing access, and initializing the client for image handling.

2.2.2 Image labelling

Labeling data plays a crucial role in training a model for hand detection in machine learning. In the case of hand detection, it involves annotating images with information about the precise location of the hand in the picture. For our study, a sample of 750 images was selected.

The labeling of images was performed using labelImg[14], a free, open-source image annotation software using graphics. This tool was released in 2015 and is written in Python. With its help, we can easily and efficiently label objects in images by drawing rectangular boxes around the areas of interest.

2.2.3 Data conversion

The TensorFlow2 Object Detection API uses its own binary file format called TFRecord. After the image labeling process in the previous step, we obtained files in XML format (Fig. 12) containing information about object labels in the images. This format can be easily converted to the TFRecord format, which is more efficient for training models in TensorFlow2. As a binary format, it offers disk space savings and faster data reading, resulting in a shorter machine learning training time. During the study, the following script [13] was used to convert XML files to TFRecord format.

```
<annotation>
  <folder>train</folder>
  <filename>f-0001.png</filename>
  <path>C:\Users\rav12\Desktop\tf2\models\research\object_detection\images\train\f-0001.png</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>1280</width>
    <height>720</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>hand</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>68</xmin>
      <ymin>198</ymin>
      <xmax>410</xmax>
      <ymax>549</ymax>
    </bndbox>
  </object>
</annotation>
```

Fig. 12: Exemplary XML file structure.

2.2.4 Transfer learning

Transfer learning, which involves using pre-trained models for new tasks, is more advantageous in our case than creating a model from scratch. The limited sample of available data would make it challenging to build an effective model; however, with transfer learning, we have access to knowledge accumulated in models trained on large datasets. By adapting the existing model's weights to a new task, we can achieve the desired performance more quickly, and the process requires less computational power than training a model from scratch (Fig.13).

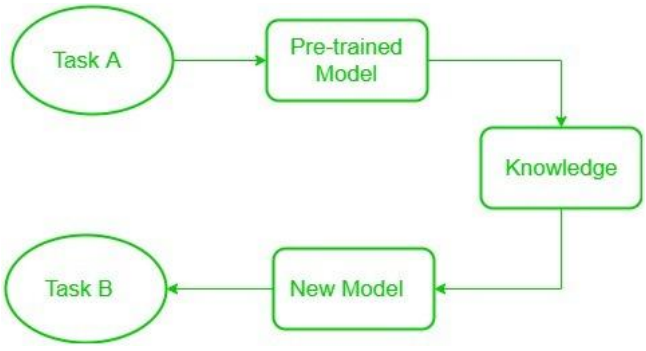

Fig. 13: Simplified scheme of transfer learning[30].

For the study, a model trained on the COCO 2017 dataset was utilized[8]. It has the ability to detect objects using bounding boxes and segmentation masks for 80 object categories. Additionally, it enables generating language descriptions for images and keypoint detection on over 200,000 images and 250,000 instances of humans, marking 17 different body points such as the left eye or nose. This characteristic makes the model previously trained on this dataset suitable for our application.

2.2.5 Model selection

Among the types of neural networks, with the increase in computational power of computers, a special place is occupied by Convolutional Neural Networks (CNN) [33]. In them, three basic types of layers can be distinguished: convolutional layer, pooling layer, and fully connected layer (Fig. 14)

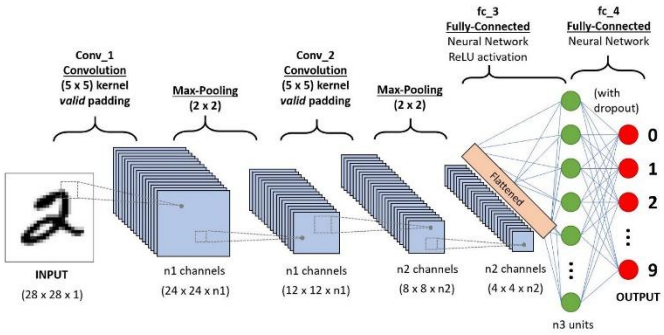

Fig. 14: CNN model[11].

The first layer in the convolutional network is the convolutional layer, followed by subsequent convolutional layers, interspersed with pooling layers. The final layer is the fully connected layer, which is not utilized in the transfer learning process. Earlier layers focus on basic features such as colours and edges, while subsequent layers aim to extract increasingly complex features and structures. This architecture allows for assigning objects to appropriate classes, which is crucial in the context of our application.

The classifier (the part of the network responsible for assigning objects to specific classes), except for the last two layers, consists of a structure called the Feature Extractor

(F.E.). There are many variations of the F.E. architecture (e.g., ResNet, DenseNet, MobileNet, etc.), but essentially, F.E. is a sequence of convolutional layers with activation functions, generating multiple feature maps that flow through the network[31].

We have at our disposal a rich set of pre-trained neural networks provided by TensorFlow 2. For our task, we will utilize a specialized section called the TensorFlow 2 Detection Model Zoo[29], created for tasks related to efficient object detection. The list of available models consists of the following families:

a) Fast-RCNN.

Faster R-CNN is a widely used and versatile object detection model. It combines a Region Proposal Network (RPN) to generate potential bounding box proposals around objects and a convolutional neural network for classifying and refining these proposals. Faster R-CNN achieves impressive accuracy but may require larger computational resources compared to some other models.

b) CenterNet.

CenterNet is a one-stage detection model with high accuracy. Target features are extracted from input images by the base network and then passed to fully convolutional networks to obtain a heatmap. The peak points on the heatmaps are the centroids of the targets. The positions and categories of the targets are obtained by regressing the location of the centroid[35].

c) SSD.

SSD (Single Shot MultiBox Detector) is known for its speed and efficiency, while maintaining competitive accuracy. It utilizes a single neural network to simultaneously predict object classes and bounding box offsets for multiple default boxes of various sizes and aspect ratios. This model is used in cases where speed is the most crucial factor.

d) EfficientDet.

EfficientDet is an advanced object detection model that leverages the efficient neural network architecture EfficientNet, characterized by high accuracy with minimal computational resource consumption. It enables efficient object detection across various

scales and aspect ratios, providing both precision and computational efficiency. Considering the available models and their characteristics, the EfficientDet-d0 model was chosen for the study, as it offers the most favourable performance-to-resource ratio[15].

2.2.6. Model training

The foundation for effective training of the model for detecting extended hands is the proper preparation of training data, including dividing the set of 750 images into training, validation, and test sets. The training set is directly used to train the model. During training, the algorithm evaluates the correctness of predicting classes and object locations in images and adjusts the weights of the neural network through backpropagation. The validation set is used by the algorithm to monitor training progress and adjust hyperparameters. Compared to the training set, elements of the validation set are used periodically every few training steps. The test set contains data not used by our network during the model training process. Its purpose is for human evaluation of the correctness of the utilized model. In my study, I applied a data set split into training, validation, and test parts in an 80-10-10% proportion.

The tool used to track the training progress of the investigated model is TensorBoard. With its help, visualization of charts is possible, which is crucial in optimization problems. Using this tool, the following charts were obtained: Loss/classification_loss (Fig.15), Loss/localization_loss (Fig.16), Loss/regularization_loss (Fig. 17), Learning rate (Fig. 18), and Loss/total_loss (Fig.19). Particularly important is the Loss/total_loss chart. To achieve a highly effective model, the training process should be carried out until the flattening of this chart[3].

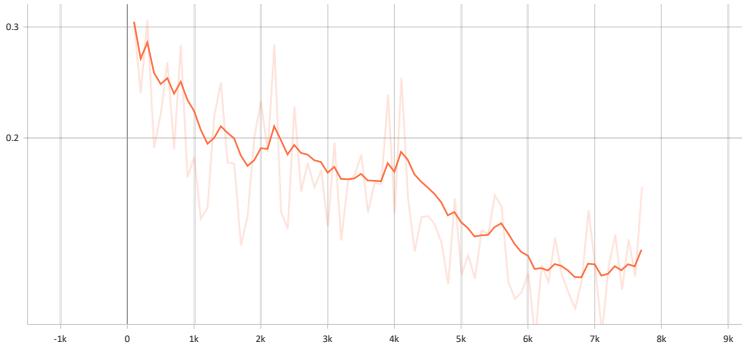

Fig. 15: Loss/classification_loss graph.

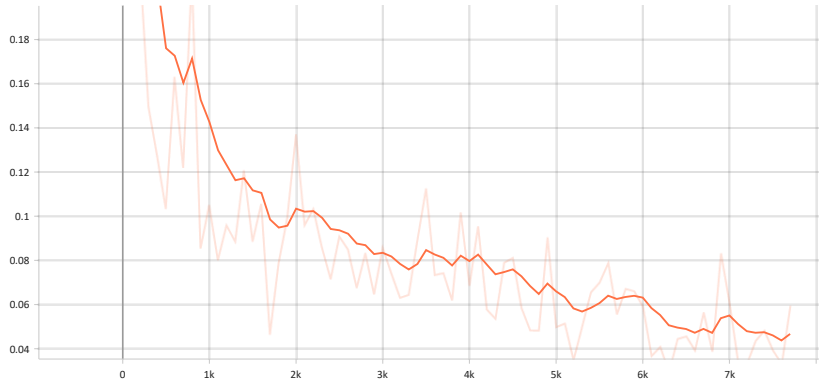

Fig. 16: Loss/localization_loss graph.

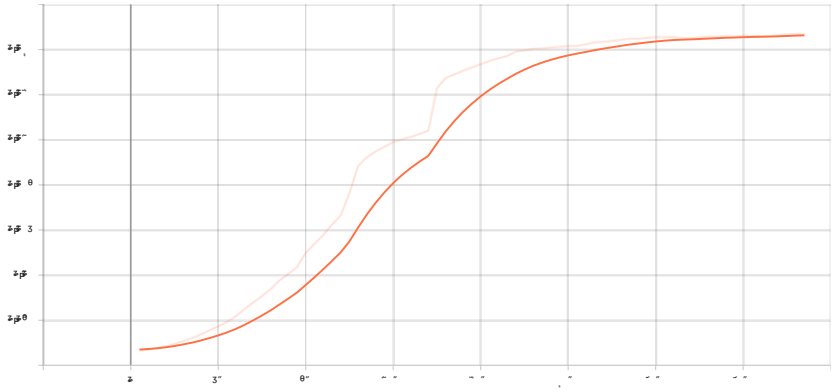

Fig. 17: Loss/regularization_loss graph.

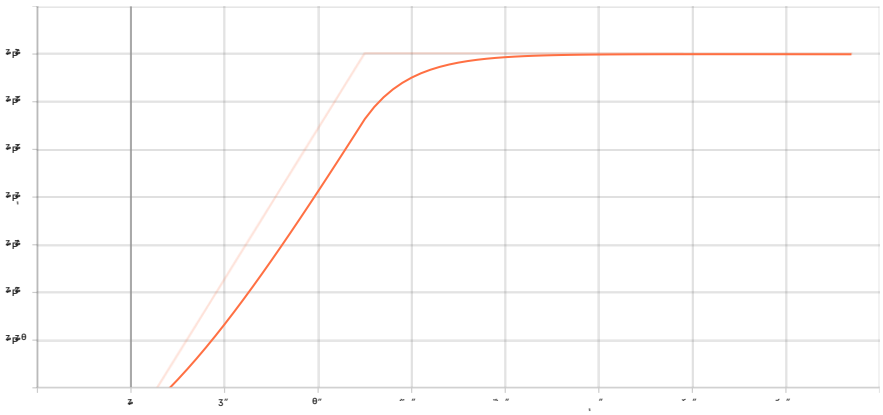

Fig. 18: Learning rate graph.

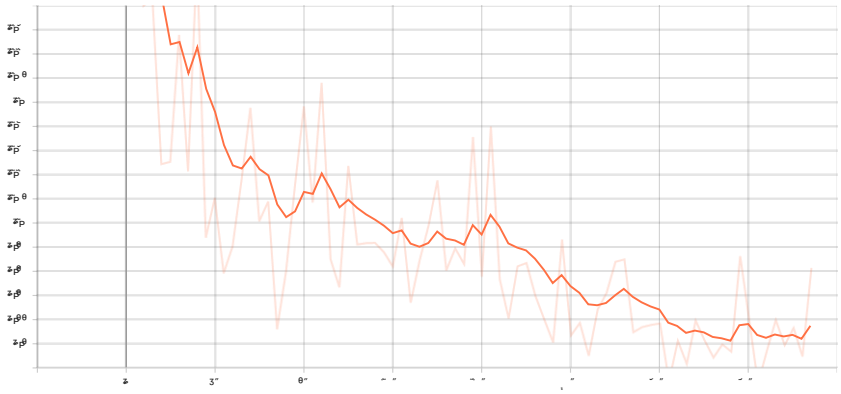

Fig. 19: Loss/total_loss graph.

2.3 Principle of operation

In the previous chapter, a hand detection model was developed and trained using transfer learning on a set of images. This model has been successfully utilized for the identification and localization of hands in images processed by Spot.

In this chapter, the focus will be on expanding the application of this model by implementing hand motion tracking functionality. The main emphasis will be on a thorough exploration of available methods for monitoring hand movements, with a particular focus on accurately determining the distance between the user's hand and the robot. The significant goal will be to enhance the system to allow for precise distance determination, which plays a crucial role in the communication between the robot and the human

2.3.1 The usage of the stereo camera in the distance measurement

Stereo cameras are widely used in tasks related to accurately determining the distance between a detected object and the camera[5]. Stereo cameras operate by using two lenses, simulating the human eye's visual process. This dual perspective allows stereo cameras to precisely record differences in the position of objects in the image, known as parallax (Fig.19). This mechanism forms the foundation for the accurate three-dimensional reconstruction of space.

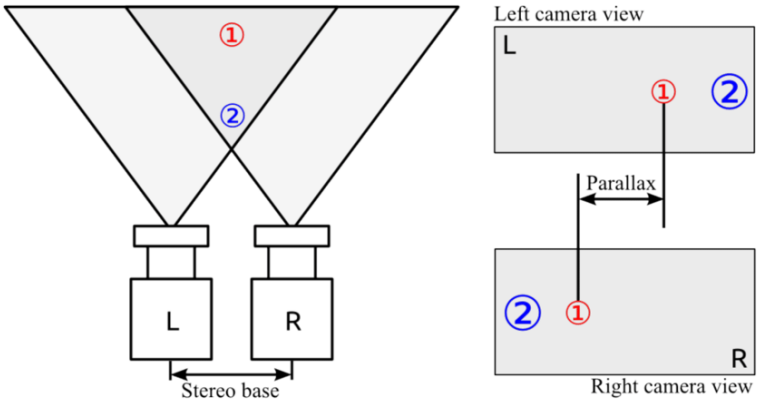

Fig. 19: Principle of operation of stereo cameras[28].

2.3.2 Libraries

To efficiently implement and utilize the hand detection model, a key step is to leverage appropriate libraries that enable advanced image processing and manipulation of numerical data. Within this project, the choice primarily fell on two libraries: OpenCV (cv2) and NumPy.

OpenCV, or Open Source Computer Vision Library, is an immensely powerful tool in the field of image processing and computer vision analysis. This library offers a rich collection of functions, allowing for effective manipulation of images, object detection, motion analysis, and many other tasks related to visual processing. In the case of the pre-trained model, this library was utilized to outline the detected objects in images using rectangles, a crucial step for subsequent operations.

NumPy is a Python library dedicated to handling operations on large arrays and matrices of data, making it indispensable for numerical programming. NumPy provides efficient methods for data processing, including various mathematical functions, matrix manipulation, and algorithms for working with numerical data. In this work, NumPy was used to effectively manage data related to images from the stereo camera. With the ability to operate on large sets of numerical data, this library significantly accelerated processes related to image analysis, enabling fast and efficient computations necessary for implementing algorithms related to hand detection and calculating the distance between the hand and the Spot robot.

2.3.2 Script

To implement the model, a script provided by Boston Dynamics[12] will be utilized. This script operates as follows: it captures images from the two front cameras of the Spot robot and, using our pre-trained model, performs detection for our newly created class. These operations are executed for a specified number of iterations, with a certain waiting time between each of them. Subsequently, the script determines the location of the object with the highest confidence for the specified class and directs the robot's movement towards that object.

The program is organized as three sets of Python processes that communicate with the Spot robot (Fig. 20). The main process communicates with the Spot robot using GRPC and continuously receives images. These images are placed in the RAW_IMAGES_QUEUE and read by the TensorFlow processes. These processes detect objects in the images and place their locations in the PROCESSED_BOXES_QUEUE. Then, the main thread determines the object's location and sends commands to the robot to move towards the object.

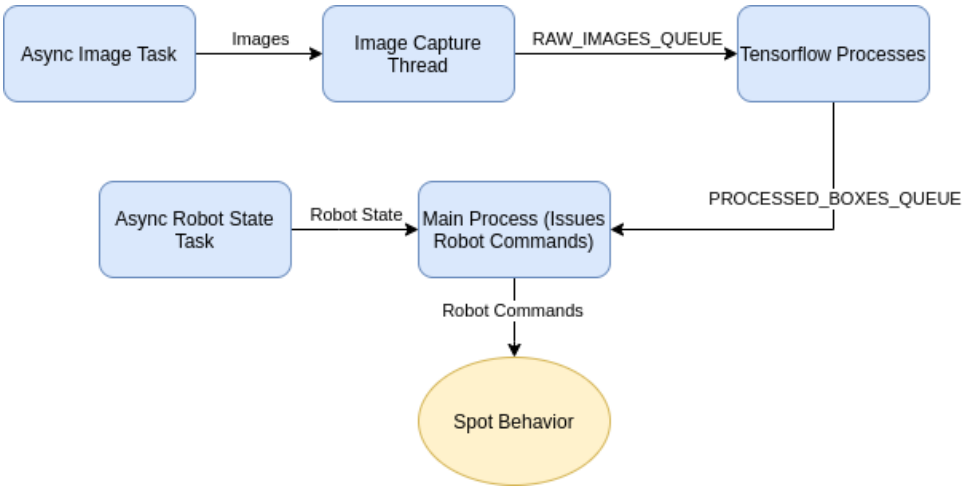

Fig. 20: System's flowchart [12].

In the further part of this subsection, attention will be focused on discussing key aspects that are of significant importance for the functioning of the code. Analysing implementation details, the script structure as well as the functions of individual modules will be presented:

Code fragment 1: Rotation angles.

```
ROTATION_ANGLES = {
    'back_fisheye_image': 0,
    'frontleft_fisheye_image': -78,
    'frontright_fisheye_image': -102,
    'left_fisheye_image': 0,
    'right_fisheye_image': 180
}
```

Spot is able to detect a hand using a set of cameras. The key stage is adjusting the image acquired through them. In our case, it is necessary to use image rotation to achieve the correct orientation, as the cameras are mounted at an angle (with the exception of the left and back cameras).

Code fragment 2: Get_go_to function.

```
get_go_to(world_tform_object, robot_state, mobility_params,
dist_margin=0.5):
    vo_tform_robot =
get_vision_tform_body(robot_state.kinematic_state.transforms_snapshot)
    print(f'robot pos: {vo_tform_robot}')
    delta_ewrt_vo = np.array(
        [world_tform_object.x - vo_tform_robot.x, world_tform_object.y
- vo_tform_robot.y, 0])
    norm = np.linalg.norm(delta_ewrt_vo)
    if norm == 0:
```

```
        return None
    delta_ewrt_vo_norm = delta_ewrt_vo / norm
    heading = _get_heading(delta_ewrt_vo_norm)
    vo_tform_goal = np.array([
        world_tform_object.x - delta_ewrt_vo_norm[0] * dist_margin,
        world_tform_object.y - delta_ewrt_vo_norm[1] * dist_margin
    ])
    tag_cmd =
RobotCommandBuilder.trajectory_command(goal_x=vo_tform_goal[0],goal_y=
vo_tform_goal[1],goal_heading=heading,frame_name=VISION_FRAME_NAME,par
ams=mobility_params)

    return tag_cmd
```

Get_go_to function is determining a goal position for the robot based on the detected object's position, and then generating a trajectory command for the robot to move towards that goal. The goal position is calculated by considering the position of the detected object relative to the robot's current position. The destination is calculated within the distance margin set to 0.5m.

Code fragment 3: Depth_to_xyz function.

```
def depth_to_xyz(depth, pixel_x, pixel_y, focal_length,
principal_point):

    x = depth * (pixel_x - principal_point.x) / focal_length.x
    y = depth * (pixel_y - principal_point.y) / focal_length.y
    z = depth
    return x, y, z
```

Depth_to_xyz function, with the usage of depth value, pixel coordinates of the point in the image, focal length and principal points of the camera, calculates 3D coordinates in the camera's coordinate system. This function is crucial, because of its usage in the later part of the code. With SE3Pose math helper (encapsulate the translation and rotation used for transformations), it is determined 3D pose of the camera relative to the object.

Code fragment 4: Get_object_position function.

```
def get_object_position(world_tform_cam, world_tform_gpe, visual_dims,
depth_image, bounding_box,
                        rotation_angle):

    if visual_dims is None or depth_image is None:
        return

    points = [(bounding_box[1], bounding_box[0]), (bounding_box[3],
```

```
bounding_box[0]),
              (bounding_box[3], bounding_box[2]), (bounding_box[1],
bounding_box[2])]

    origin = (visual_dims[0] / 2, visual_dims[1] / 2)

    points_rot = [rotate_about_origin_degrees(origin, point,
rotation_angle) for point in points]

    y_min = max(0, min([point[1] for point in points_rot]))
    x_min = max(0, min([point[0] for point in points_rot]))
    y_max = min(visual_dims[1], max([point[1] for point in
points_rot]))
    x_max = min(visual_dims[0], max([point[0] for point in
points_rot]))

    if (x_min < 0 or y_min < 0 or x_max > visual_dims[0] or y_max >
visual_dims[1]):
        print(f'Bounding box is invalid: ({x_min}, {y_min}) |
({x_max}, {y_max})')
        print(f'Bounds: ({visual_dims[0]}, {visual_dims[1]})')
        return

    try:
        if depth_image.shot.image.pixel_format ==
image_pb2.Image.PIXEL_FORMAT_DEPTH_U16:
            dtype = np.uint16
        else:
            dtype = np.uint8
        img = np.fromstring(depth_image.shot.image.data, dtype=dtype)
        if depth_image.shot.image.format ==
image_pb2.Image.FORMAT_RAW:
            img = img.reshape(depth_image.shot.image.rows,
depth_image.shot.image.cols)
        else:
            img = cv2.imdecode(img, -1)
        depth_image_pixels = img
        depth_image_pixels = remove_ground_from_depth_image(
            depth_image_pixels,
depth_image.source.pinhole.intrinsics.focal_length,
            depth_image.source.pinhole.intrinsics.principal_point,
world_tform_cam, world_tform_gpe)

        max_distance = 8.0
        depth = get_distance_to_closest_object_depth(x_min, x_max,
y_min, y_max,

depth_image.source.depth_scale,

depth_image_pixels, max_distance=max_distance)

        if depth >= max_distance:

            print('Not enough depth data.')
            return False
        else:
            print(f'distance to object: {depth}')

        center_x = round((x_max - x_min) / 2.0 + x_min)
        center_y = round((y_max - y_min) / 2.0 + y_min)
```

```
        tform_x, tform_y, tform_z = depth_to_xyz(
            depth, center_x, center_y,
depth_image.source.pinhole.intrinsics.focal_length,
            depth_image.source.pinhole.intrinsics.principal_point)
        camera_tform_obj = SE3Pose(tform_x, tform_y, tform_z, Quat())

        return world_tform_cam * camera_tform_obj
    except Exception as exc:
        print(f'Error getting object position: {exc}')
        return
```

Get_object_position function uses previously explained *Get_object_position* function to finally perform calculation : *world_tform_cam * camera_tform_obj* , resulting in representation the position of the object in the world frame. Additionally, it has implemented exception handling, if any issues would occur during this process. This code has build in limitation, when it comes to allowed distance to distance an object. If the depth is greater than or equal to 8.0, an error message is printed, and the function terminates.

2.4 Safety measures

The Spot robot is a multifunctional device designed to operate in diverse conditions. It has multiple safety features in case of incorrect robot operation or errors caused by human actions. Efficient and, above all, safe work with the robot requires understanding its limitations and learning procedures for handling undesired events. During the period when the research was conducted, the following precautions were taken:

a) E-stop functionality.

The E-stop function was a fundamental measure taken during work with the robot. This function can be activated using the built-in button located on the rear part of Spot or through a tablet with the installed application. During the testing process of the trained model using the robot, the work was carried out with the involvement of two persons. The first person served as a tester responsible for presenting their hand in front of Spot to verify if the robot could properly detect the open hand in real conditions. The

second person supervised the process and, in case of an undesired situation potentially threatening safety, was ready to use the E-stop function with the help of the tablet.

b) Maintaining a safe distance.

Responsible work with Spot the robot involves a limitation in the form of a minimum distance between a person and the robot, ensuring that this distance is not less than 2 meters [27]. This minimizes the risk of undesired situations. When testing programs responsible for Spot's movement, users are required to adhere to certain principles. Never touch the robot when its motors are active, and avoid standing at the base of an incline where Spot is positioned.

a) Appropriate testing room.

To ensure the safety of both the robot and the individuals conducting the study, a room with dimensions of 5x5 meters was chosen. Special emphasis was placed on preventing Spot from sustaining significant mechanical damage in the event of a collision with the surroundings. While Spot is equipped with features allowing it to navigate stairs, it is advisable to avoid this type of movement for the specific function being investigated (Fig. 21).

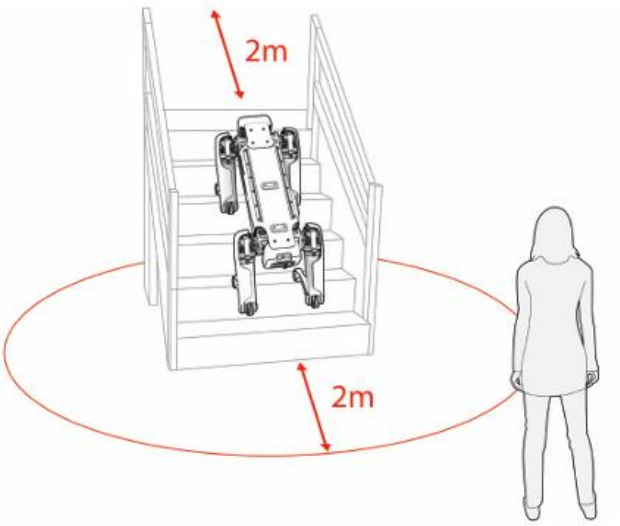

Fig. 21: Illustration showing the correct positioning in regards to the position of the robot [27].

3. Results

Fundament of the correct functioning is properly trained neural network model to detect a hand. To evaluate effectiveness of the detector will be used the concept of mean Average Position (mAP). It is basing on IoU (Intersection over Union) (Fig. 22), the term expressed as the ratio of the overlap of the areas determined by the model and area determined by the sum of these areas. Object area of the model is verified to check if in the assigned place exists the desired object.

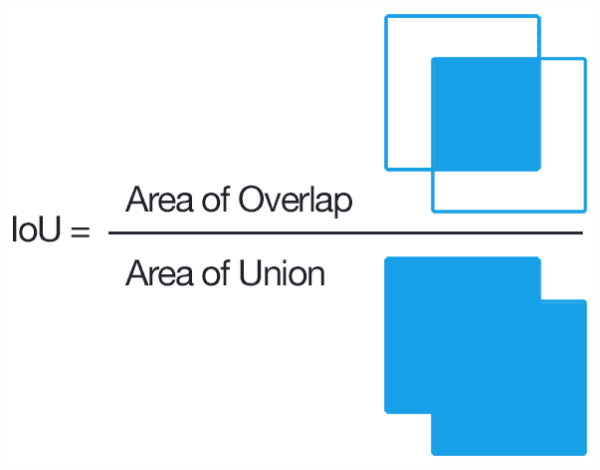

Fig. 22 Illustration representing correct positioning [6].

To calculate mAP there was a specially designed calculator . Its work reveals in a way that it sorts the results of detection with decreasing confidence and assigns it to the reference objects on the base of IoU coefficient and calculates Average Precision (AP) using numerical integration of the curve Precision Recall For the examined model the following result was obtained: 55.79%. The accuracy of the model of this rate allowed the robot to swiftly detect a hand (Fig. 23), as this influences the speed which Spot is able to approach the hand.

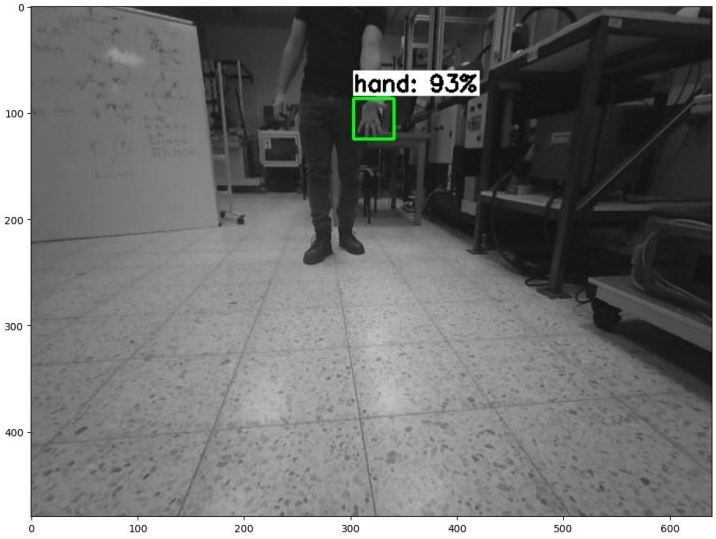

Fig. 23: Example of hand detection function.

While using this feature we have to be aware of the limitations and potential issues, which need to be resolved. Research and tests were conducted in the laboratory where external factors have no access. With appropriate lighting the system had no problems with hand detection. Ensuring similarly effective conditions in much more environments would require many more samples of marked images and correctly optimized model training process. Despite the fact that using this feature is possible in theory without configurating the e-stop, this approach is endangered to appearing of many potentially harmful situations so it is not advised to consider.

4. Conclusions

The aim of preceding engineering thesis was to develop a unique functionality of the Spot robot, which is to follow an open hand. This feature was obtained with the use of already existing tools shared as the elements of SDK toolbox from Boston Dynamics as well as the methods related to machine learning and neural networks. Under special treatment was put the use of earlier trained models, that were able to be used as a foundation on which the functionality was examined.

The first step of learning a new model was to gather varied sets of data samples, using loads of robot's built-in cameras. Next with the use of appropriate libraries Tensorflow and COCO database trained model EfficientDet-d0 a key model for the examined model in the given functionality.

After training the model, the process of script's implementation was initialized. There were introduced the ways that robot processes the distances between robot and a detected object. The emphasis was placed also on the safety measures in the applied methods as well as explaining optimal testing environment.

Engineering thesis was realized with the rapid development in popularity and market growth due to mobile robots' possibilities. This thesis shows the features of Spot robot as a highly resourceful robot in industry. Although the thesis does not fully pierce the subject, the fundamental function which is following a human has a huge potential to develop in many directions. With the most important being creating the new functionality that use following the human function.

Bibliography

[1] A. Koval, S. Karlsson, G. Nikolakopoulos. Experimental evaluation of autonomous map-based Spot navigation in confined environments. Luleå University of Technology 2022.

[2] Arm specification Dostępny: https://dev.bostondynamics.com/docs/concepts/arm/arm_specification (Odwiedzony: 15.11.23)

[3]Atienza R. Advanced Deep Learning with TensorFlow 2 and Keras: Apply DL, GANs, VAEs, deep RL, unsupervised learning, object detection and segmentation, and more, 2nd Edition, 2020.

[4] Base services. Dostępny: https://dev.bostondynamics.com/docs/concepts/base_services (Odwiedzony: 8.9.23)

[5] C. Zhou, D. Ren, X. Zhang, C. Yu, L. Ju. Human Position Detection Based on Depth Camera Image Information in Mechanical Safety. Nanjing University of Science and Technology 2022.

[6] Calculator mAP. Dostępny: https://github.com/Cartucho/mAP (Odwiedzony: 8.9.23)

[7] Capture images. Dostępny: https://dev.bostondynamics.com/docs/python/fetch_tutorial/files/capture_images.py (Odwiedzony: 18.9.23)

[8] COCO. Dostępny: https://cocodataset.org/#home (Odwiedzony: 8.9.23)

[9] Conda. Dostępny: https://docs.conda.io/projects/conda/en/stable/ (Odwiedzony: 8.9.23)

[10] Core I/O Dostępny: https://bostondynamics.com/wp-content/uploads/2020/10/spot-core-io.pdf (Odwiedzony: 15.11.23)

[11] Convolutional neural network. Dostępny: https://paperswithcode.com/methods/category/convolutional-neural-networks (Odwiedzony: 8.9.23)

[12] Detect and follow. Dostępny: https://github.com/boston-dynamics/spot-sdk/tree/master/python/examples/spot_detect_and_follow (Odwiedzony: 8.9.23)

[13] Generate tfrecord. Dostępny: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/_downloads/da4babe668a8afb093cc7776d7e630f3/generate_tfrecord.py (Odwiedzony: 18.10.23)

[14] LabelImg. Dostępny: https://github.com/HumanSignal/labelImg (Odwiedzony: 18.9.23)

[15] M. Tan, R. Pang, Q. Lee. EfficientDet: Scalable and Efficient Object Detection. Google Research, Brain Team 2020.

[16] Operating Spot. Dostępny: https://support.bostondynamics.com/s/article/Operating-Spot (Odwiedzony: 15.11.23)

[17] Przemysł 5.0?. Dostępny: https://przemyslprzyszlosci.gov.pl/przemysl-5-0/ (Odwiedzony: 15.11.23)

[18] Quick start. Dostępny: https://github.com/boston-dynamics/spot-sdk/blob/master/docs/python/quickstart.md (Odwiedzony: 8.9.23)

[19] Robot services. Dostępny: https://dev.bostondynamics.com/docs/concepts/robot_services (Odwiedzony: 15.11.23)

[20] Rocznik Demograficzny 2023. Dostępny: https://stat.gov.pl/obszary-tematyczne/roczniki-statystyczne/roczniki-statystyczne/rocznik-statystyki-miedzynarodowej-2023,10,11.html (Odwiedzony: 10.11.23)

[21] Sample Return Rover. Dostępny: https://www.researchgate.net/figure/NASAs-Sample-Return-Rover-SRR_fig1_2470364 (Odwiedzony: 15.11.23)

[22] Soryu-C. Dostępny: https://www.hibot.co.jp/products/soryu-c/ (Odwiedzony: 15.11.23)

[23] Spot battery and charger. Dostępny: https://support.bostondynamics.com/s/article/Introduction-to-the-Spot-battery-and-charger (Odwiedzony: 15.11.23)

[24] Spot Boston Dynamics. Dostępny: https://bostondynamics.com/wp-content/uploads/2023/05/spot-explorer-web-2-2048x1152.jpg (Odwiedzony: 15.11.23)

[25] Spot Classic. Dostępny: https://www.bostondynamics.com/wp-content/uploads/2023/07/spot-classic-2-1.jpg (Odwiedzony: 15.11.23)

[26] Spot self-charging station Dostępny: https://support.bostondynamics.com/s/article/About-the-Spot-Dock-self-charging-station (Odwiedzony: 15.11.23)

[27] Spot user guide. Dostępny: https://www.generationrobots.com/media/spot-boston-dynamics/spot-user-guide-r2.0-va.pdf (Odwiedzony: 8.9.23)

[28] Stereo camera. Dostępny: https://www.researchgate.net/figure/Principle-drawing-of-a-stereo-camera-setup-Objects-1-2-in-various-depth-ranges-are_fig5_303307354 (Odwiedzony: 8.9.23)

[29] Tensorflow ZOO. Dostępny: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (Odwiedzony: 8.9.23)

[30] Transfer learning. Dostępny: https://www.geeksforgeeks.org/ml-introduction-to-transfer-learning/ (Odwiedzony: 18.11.23)

[31] U. Almog. Object Detection With Deep Learning: RCNN, Anchors, Non-Maximum-Suppression. Dostępny: https://medium.com/swlh/object-detection-with-deep-learning-rcnn-anchors-non-maximum-suppression-ce5a83c7c62b (Odwiedzony: 8.9.23)

[32] V. Vanniyakulasingam. Autonomous mission configuration on Spot from Boston Dynamics, Politecnico di Torino, 2023.

[33] W. Garmacka. Wykorzystanie wstępnie wytrenowanych konwolucyjnych sieci neuronowych w zagadnieniu transferu stylu. Wojskowa Akademia Techniczna 2023.

[34] World Robotics 2023 Report: Asia ahead of Europe and the Americas. Dostępny: https://ifr.org/ifr-press-releases/news/world-robotics-2023-report-asia-ahead-of-europe-and-the-americas (Odwiedzony: 10.11.23)

[35] Y. Guo, X. Lu. ST-CenterNet: Small Target Detection Algorithm with Adaptive Data Enhancement, Entropy 2023.

[36] Zielińska T. Maszyny kroczące , Warszawa 2014.