# MARI4YARD

MARI4ALLIANCE

**Grant agreement nº: 101006798**

**Call identifier: MG-3-7-2020**

**Deliverable D4.3**

Process perception module

**Work Package 4**

Exoskeletons assisting workers in outfitting & assembly Tasks

| | |
|---|---|
| **Document type** | : Deliverable |
| **Version** | : V1 |
| **Date of issue** | : 31/03/2023 |
| **Dissemination level** | : PUBLIC |
| **Lead beneficiary** | : LMS |

***This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement nº 101006798.***

The dissemination of results herein reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

## DOCUMENT HISTORY

| Version | Date | Changes | Stage | Distribution |
|---|---|---|---|---|
| V01 | 27/02/2023 | Initial document | draft | |
| V02 | 20/03/2023 | IUVO comments | draft | |
| V1 | 31/03/2023 | LMS addressed comments | draft | |

## EXECUTIVE SUMMARY

Text.

# 1 INTRODUCTION

The general purpose of the work package 4 is the design and development of wearable technologies that provide adaptive physical support to skilled workers operating in tasks of outfitting and assembly. Specifically, the aim is the development of two spring-loaded exoskeletons, respectively for the upper-limb and lumbar support. The hardware is supported with AI-based control that allows the devices to adapt their behavior online and autonomously, to assist the operators according to the estimated user's fatigue and based on the recognized task.
The deliverable D4.3, reports the development of a novel perception system that allows the detection/recognitions of the tasks being executed by human operators, facilitating the selection of the appropriate support settings for exoskeletons in an automated manner. The name of the perception module is Human Action Perception module (HAP).

The rest document is divided as follows; in section 2, the approach of HAP is presented, including information regarding the requirements, architecture, and the high-level descriptions of the included developments. Afterwards the implementation section follows, analyzing the developed algorithms and implementations in detail. Last but not least, the conclusions and the outlook are presented.

# 2 APPROACH

Section 2 presents the methodology of the HAP module, including an overview of its structure and a brief explanation of its key components.

## 2.1. Architecture

The main objective is the development of an AI methodology, utilizing computer vision techniques to recognize the activity of a worker while performing specific tasks. It focuses on a specific use case, to calculate an appropriate Level of Assistance (LoA) for an upper body, semi-active exoskeleton. To this end, a Human Action Perception (HAP) module was developed that combines several features that enable automatic assistance to the operator, in order to perform their tasks efficiently.

The main features are the following.

- Hand recognition
- Equipment recognition
- Mapping of hands to parts
- Activity recognition and intention prediction
- Focus detection.
- Interconnectivity

All these features are combined in order to determine a final LoA that the robot's operator is going to receive.

The HAP module consists of three layers. The first one, is the sensing layer and it includes all the sensory network that is being utilized to receive data such RGB images, IMU acquisitions, barometric pressure values etc. The sensor data are then shared to the processing layer, where their analysis is performed, including filtering, and reshaping. During this stage, the data is categorized based on their type to allow for input into the appropriate algorithm or deep learning network for decision-making. Afterwards, the results are inserted in a decision-making algorithm, to decide the required level of assistance of the exoskeleton for the specific process among the human's operations; this comprises the output layer of the HAP module.

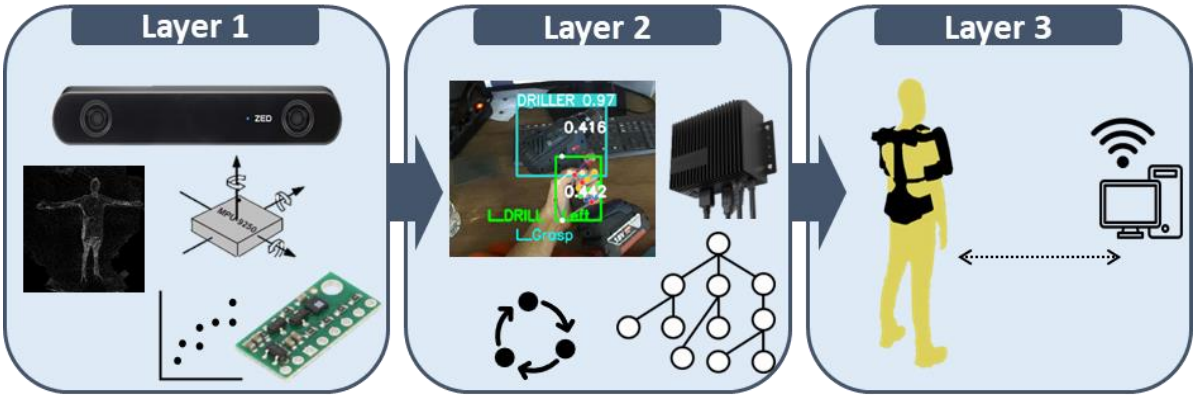


**Figure 1 HAP architecture layers**

## 2.2. Object detection pipeline

While executing different tasks several tools are used that correspond to and characterize them. For example, if the task is welding, a clamp and a welding torch must be used and so they are held by the operator.

In order recognize the tools, an object detection method has been deployed. It uses state of the art computer vision methods and machine learning models to achieve this. To train these models, both real-life and synthetic datasets were created, with annotated pictures containing the equipment the operator would be using.

Detecting these objects is also useful for determining how much weight the operator is lifting. Heavier objects will require higher LoAs. Furthermore, the weight of each tool must be considered when calculating the LoA, as the operator may need to hold them for extended periods and in challenging positions.

Determining the type of tool used can act as a trigger for certain tasks. While cleaning, a specific machine is held, and recognizing it can help decide on what algorithm will be used to check if it has been turned on and the cleaning task has started.

Enhancing the proposed approach, and pushing the limits of the detection capabilities of the systems for fast training even tough lack of the datasets occur, it an approach for synthetic datasets preparation and training has also been utilized.

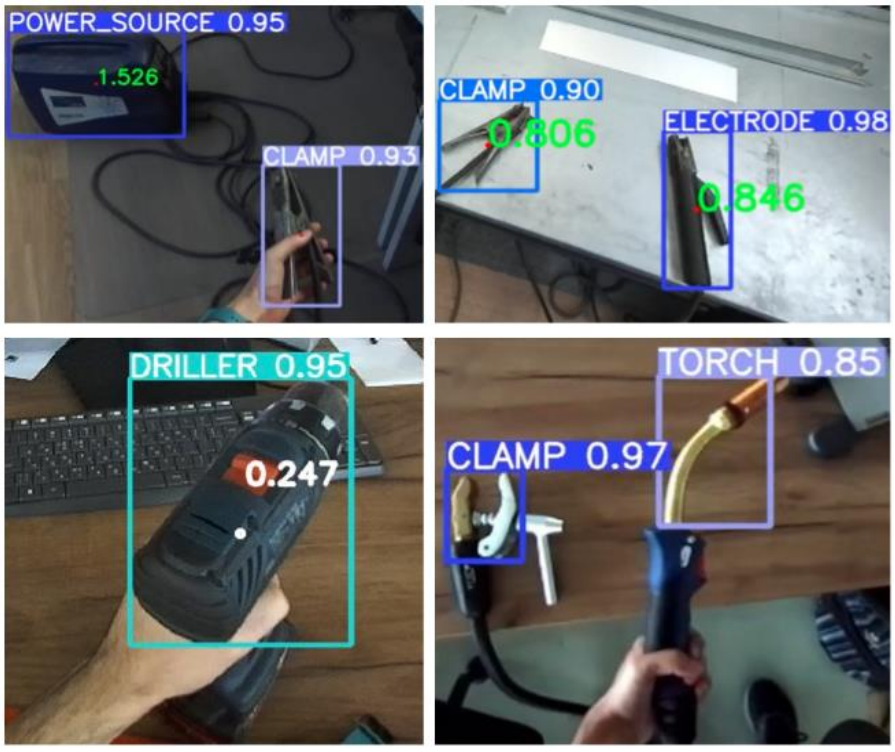


**Figure 2 Examples of tools being detected.**

## 2.3. Human action recognition pipeline

To recognize what action the operator is performing, a series of algorithms have been implemented. Action recognition is the main purpose of the HAP module and plays the biggest role in LoA calculation. To recognize a task, the following factors are considered:

- What tools are being detected
- Which of them the operator is holding
- If the task has been initialized

Detecting tools is crucial since they are associated with specific tasks and aid in identifying them. Simply detecting a tool in the environment is insufficient; HAP must also determine whether the operator is holding and using the tool.

To know what task the operator is performing, they need to be holding the appropriate tool. In some cases, such as painting, this information alone is adequate. For other tasks, such as welding, HAP predicts the initialization trigger by observing changes in image lighting when the tool is turned on. Additionally, for tasks like ladder climbing, specialized methods have been developed that use unique markers or the IMU data of the ZED camera to recognize their action.


**Figure 3 Welding task recognition**

Equipment recognition is performed utilizing the method described above. Mapping of hands to parts is achieved by combining the results of hand recognition and equipment recognition while taking into account their relative distance and the distance from the RGB-D sensor. The IMU data of the ZED camera is used to determine whether the operator is focused thus helping deduce their intention.

## 2.4. Level of assistance adjustment

The exoskeleton utilizes a combination of algorithms to determine the appropriate level of assistance (LoA) required for each limb. The LoA is calculated based on various factors, such as the weight of held objects, the operator's head orientation, and the amount of stress on each limb. The system provides more assistance to the limb under stress and less to the others, allowing for greater freedom of movement. The appropriate LoA is continuously calculated, but adjustments take some time and consume power, so the system averages the LoA over a predetermined time period before communicating it to the operator. In cases where immediate action is required, such as when maximum or minimum LoA is needed, the system responds accordingly, taking ergonomic concerns into account. In this context the aim of HAP is to provide the control system with the appropriate LoA adjustment commands wisely; more in-depth information may be found in the D4.1.

## 2.5. Interconnection

There are two modules that calculate a LoA for the exoskeleton’s upper limbs. The one is of course the HAP module that uses AI and computer vision to analyze the data provided by the ZED camera and recognize the tasks that the operator performs and the tools they use. The other module calculates the LoA based on biometrics and the operator’s stress levels at any given time. Both modules communicate with the exoskeleton’s control module and request a LoA to be provided. The HAP module needs to be aware of the current LoA as well as provide the LoA it calculates. To provide this functionality, a IoT messaging protocol was utilized call MQTT. The communication method is described in more detail in the implementation section.

# 3 IMPLEMENTATION

## 3.1. System Structure

## 3.2. Hardware components

The hardware components that were used for training the Machine Learning models as well as running the overall HAP module are shown in Table 1. The software that was used is also described in Table 2.

| Device | Description | Reference |
|---|---|---|
| Desktop PC | CPU: 11th Gen Intel® Core™ i7-11700F @ 2.50GHz<br>GPU: Nvidia RTX 3070 ti<br>RAM: 32GB DDR4 @ 3200MHz<br>Power supply: Corsair CX750 (750W) | |
| System On a Module (Portable PC) | ZED Box with Nvidia Jetson Xavier GPU: 384-Core Nvidia Volta with 48 Tensor Cores<br>CPU: 6-Core Nvidia Carmel ARM v8.2 64 bit<br>DL accelerator: 2xNVDLA<br>Memory: 8GB LPDDR4x 59.7 GB/s<br>Power supply: 100W power bank @ 20V - 5A | https://store.stereolabs.com/products/zed-box-xavier-nx<br>https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/ |
| Camera | ZED 2 AI Stereo Camera<br>The ZED 2 is a stereo camera that provides high-definition 3D video and neural depth perception of the environment. The camera also offers a Built-in IMU, Barometer & Magnetometer | https://www.stereolabs.com/zed-2/ |
| Microsoft HoloLens 2 | The Microsoft HoloLens 2 is a mixed reality headset that allows users to interact with holograms and virtual objects in the real world. It has an enhanced field of view, eye, and hand tracking technology. It was used as a portable solution for testing and validation purposes. | https://www.microsoft.com/en-us/hololens |

**Table 1: Hardware components**

| Software | Description | Reference |
|---|---|---|
| Ubuntu 20.04 | Operating System | https://releases.ubuntu.com/20.04/ |
| Nvidia CUDA toolkit | Develop GPU-accelerated applications. Requirement for training Machine Learning models. Also used by ZED Camera | https://developer.nvidia.com/cuda-toolkit |
| ZED SDK | A set of tools for developing applications that use the ZED Camera | https://www.stereolabs.com/developers/release/ |
| Python 3 | High level, interpreted, object-oriented programming language | https://www.python.org/ |
| OpenCV | Library of programming functions aimed at computer vision | https://opencv.org/ |
| Pytorch | An open-source machine learning framework. Used for training YOLO object detection models | https://pytorch.org/ |
| Paho MQTT | Library for implementing the MQTT protocol. MQTT protocol is a machine-to-machine Internet of Things (IOT) connectivity protocol. | https://pypi.org/project/paho-mqtt/ |

**Table 2: Software components**

## 3.3. Communication

For the HAP module to communicate with the exoskeleton control module and other modules that change the LoA, a communication protocol using MQTT was chosen. The MQTT is a messaging protocol for machine-to-machine Internet of Things (IOT) connectivity. It uses a subscriber – publisher messaging transport. In this case, the exoskeleton's control module subscribes to several topics and the HAP module publishes the LoA as an integer to two of them. The topics are described in Figure 4.

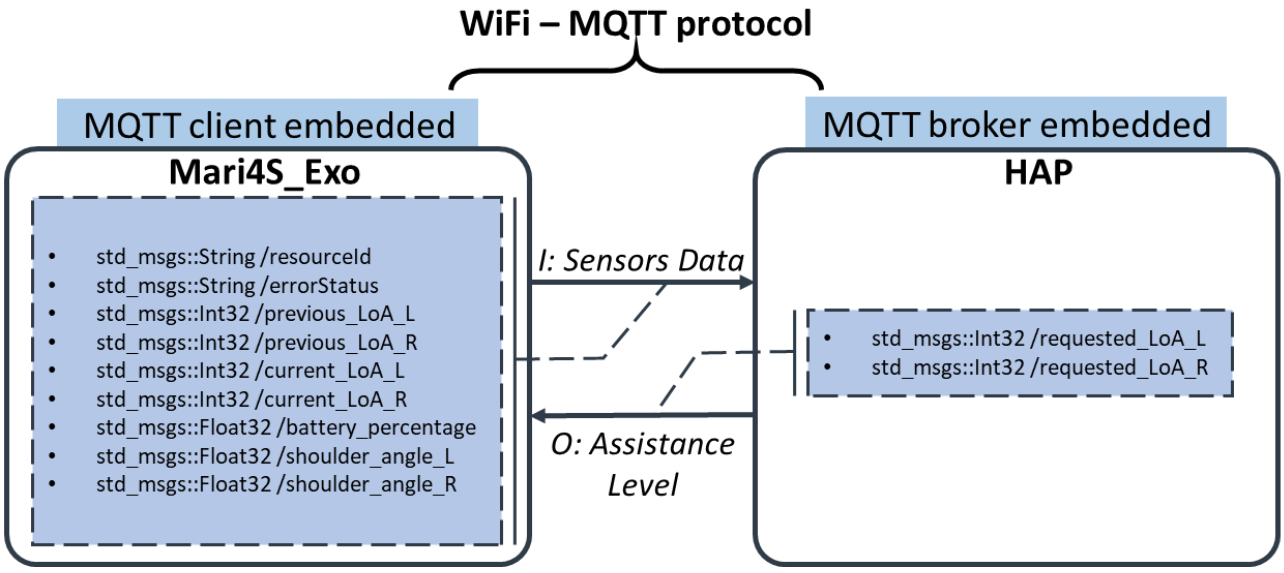


**Figure 4 MQTT communication topics**

While HAP is running, it constantly calculates the current LoA based what it perceives. The control module can adjust the torque of the exoskeleton from one LoA to another, however, this consumes power. To minimize power consumption the following points are applied:

- The LoA calculation is averaged and transmitted at fixed intervals, such as every three minutes.
- LoA might change immediately, for its minimum (1) or the maximum (5) value.

More in depth information will be provided in D4.1.

## 3.4. Object Detection

### 3.4.1. 2D image object detection

In order to decide what tools the operator is using at any given time, a Machine Learning (ML) object detection model was trained and then utilized. This model needs to accomplish four things. First, to be able to accurately detect the objects in the environment. Second, to have high confidence scores and avoid false positives. Third, to be able to detect the tools even on cases when the operator is holding them and the hand occludes them. Fourth, to be able to detect the objects in different poses that may not be represented in the training dataset as the material required for this is not always abundant. This is because the tools that are used are numerous and most of the time not available in the testbench in order to create the datasets that are required for training. Training time and the number of epochs that the model needs to be trained for also needs to be taken into consideration as the more tools that will be added, the larger the dataset needs to be.

Different object detection models were considered: Faster Region-based Convolutional Neural Network (Faster RCNN) [3], Single Shot Detection (SSD) [2], and You Only Look Once (YOLO) [1]. These are some of the most popular detection models in Machine Vision (MV) and differ between one another on their detection accuracy and speed. Thus, these three models were compared by training them on the same in-house dataset and tested on the same evaluation videos. These videos were representative of the deployment environment as much as possible, containing three tools that would be used in a welding use case. Their metrics are shown in Table 3.

| Model | Accuracy | Recall |
|---|---|---|
| Faster RCNN | 63.11% | 66.88% |
| SSD | 51.82% | 56.34% |
| YOLO5 | **83.52%** | **91.53%** |

**Table 3: Object detection models comparison**

The evaluation videos also confirmed that the YOLOv5 model was more accurate than the other two models, with higher confidence scores and lower false positives without compromising in runtime speed. Therefore, YOLOv5 was the model that was chosen to for the needs of object detection.

As far as datasets are concerned, in most cases they were created in the lab for testing the different features that were being developed. Even if videos of the tools were provided, the annotation of the image was done manually. In order to reduce dataset creation time and increase dataset size in cases where the tools were not available, the creation of synthetic datasets was explored as per [4] using blender [5]. From research and tests that were performed, a pipeline for creating synthetic datasets was developed. The steps of creating a dataset of a tool are shown in Figure 5. A YOLOv5 model was trained on three different datasets: A synthetic, consisting

only of synthetic images; a mixed dataset, consisting of both synthetic and real-life images; and a real-life dataset, consisting only of real-life images. The metrics of the detection model trained on the three different datasets are shown in Table 4.

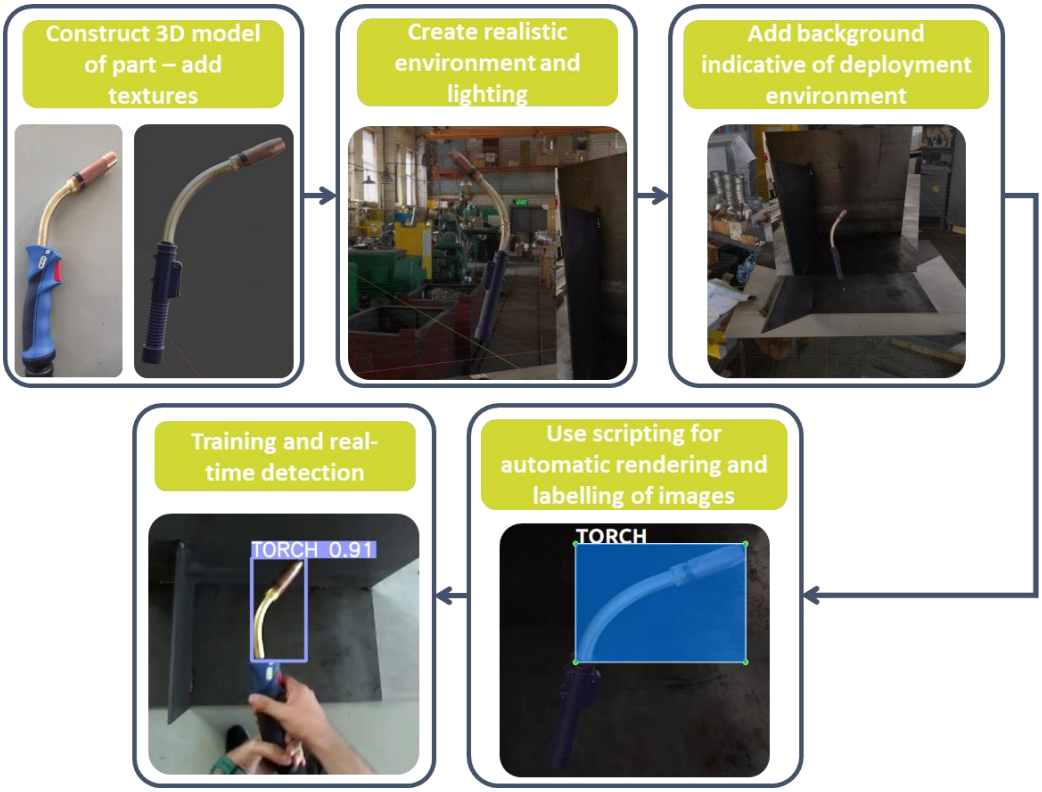


**Figure 5 Synthetic dataset creation pipeline**

| Dataset | mAP_05:0.95 | Recall |
|---|---|---|
| Synthetic | 28.29% | 37.30% |
| Mixed | 69.12% | 78.36% |
| Real-Life | **76.68%** | **80.02%** |

**Table 4: Synthetic - Real-Life datasets metrics**

The model trained only on the real-life dataset managed to outperform the model trained only on the synthetic dataset. The model that was trained on a select few 50 real-life images and the 200 synthetic images performed more closely to the real-life dataset. The best performing model was the one trained on 200 real-life images. The evaluation videos also confirmed the metrics. Consequently, unless the ability to create real-life datasets is impossible, synthetic data is not a replacement.

The use of public datasets was also explored. Public datasets would help train the models when the tools are not readily available while also reducing the manpower required to train the models as the inhouse datasets require annotating the images captured though to the best of the authors knowledge no such datasets were found. The tools are not static and not always viewed in the same angle or pose from the camera as the operator is

manipulating them in order to use them. The operator is also moving in the environment, and this creates a blur in the detection image. They are also grabbed by the operator's hands. Therefore, the datasets that are found online do not meet these criteria.

To summarize, our tests showed that when given enough data, our detection model is able to accurately detect the tools that were available in the lab and this allowed us to test all the features that were developed so far.

### 3.4.2. Position detection of detected parts

Detecting the position of the objects in the environment is important for deciding whether they are used by the operator. A tool is in use when it is held. If this is the case, it allows for LoA calculation in two different ways. First the LoA is calculated according to the weight of the tool. A lighter tool will require a lower LoA and vice versa. Second, if the tool is used by the operator it may act as a trigger for the algorithm to check for a tasks' initialization. Thus, the distance of an object from the camera and consequently its distance from the operator needs to be calculated.

This is achieved by utilizing the ZED camera's RGB-D sensor. The ZED camera can calculate the distance from any point in the image by matching the pixel coordinates to its point cloud. The object detection algorithm returns a bounding box that contains the object that was detected. Then the object is expected to be found somewhere inside this bounding box. A number of points along the middle if the bounding box are matched to the point cloud and then the point with the minimum distance value is selected. This point is chosen because some bounding boxes have a lot of "empty" space even though the object is contained within. In an egocentric point of view, the point with the minimum distance is where the object is found as the background is further away.

However, in some cases the operator's hands occlude this area as they are manipulating other tools in the environment or holding the tool itself. Therefore, a second check is performed. If the hand's calculated bounding box, as described in the sections below, overlaps with the objects' bounding box then a different point is chosen outside the overlapping area. This is demonstrated in Figure 6


**Figure 6 Distance calculation while hand obscures object**

### 3.4.3. Welding and waterjet detection features

The algorithm can so far detect the initialization of two tasks: The welding task and the waterjet cleaning task. This is accomplished by detecting when the operator is holding certain tools. For the welding task that is the welding torch and for the cleaning task is the tool that ejects water. For each task a specific trigger is chosen, based on its characteristics.

When welding, the activation of the machine produces a bright light that illuminates the image near the location where the torch tool is detected, resulting in a sudden increase in the pixel values of this region. If the operator is holding the welding torch and the mean average value of this region rises significantly, it can be inferred that welding is being performed. Figure 7 depicts this process.

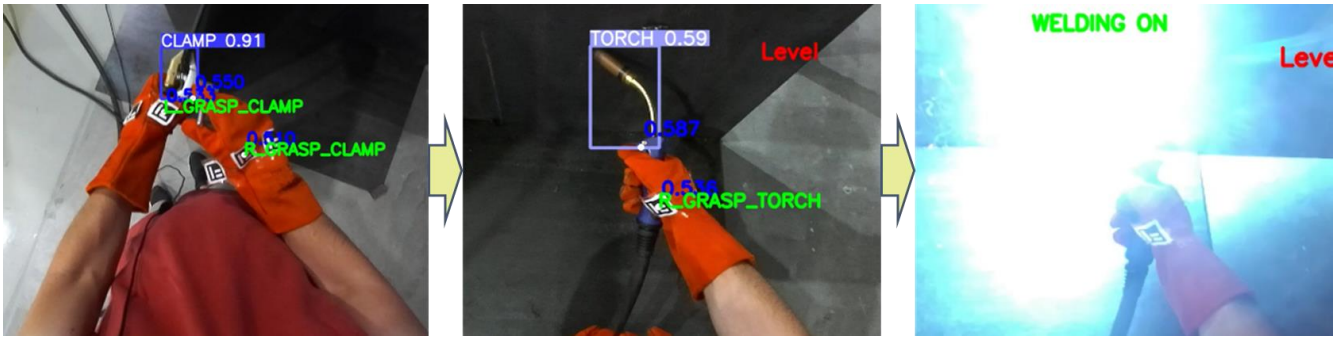


**Figure 7 Welding task initilization**

Regarding the cleaning task, the trigger is different. When the tool is turned on it produces a white-water jet that can be detected using two ways. The first way is training the object detection model to detect this jet. If its calculated bounding box is next to the tool's bounding box, then the machine is on. The second way is similar to the welding's initialization. As the water jet is white it increases the values of the pixels in the image. By using an OpenCV library function the brightest spot in the image can be detected. That spot is where the water jet is located. If this spot is also close to the bounding box of the object, again, this means that the machine is on. The different methods described are shown in Figure 8.


**Figure 8 Cleaning task initialization**

## 3.5. Human action recognition

### 3.5.1. Hand tracking

The presented approach demonstrates how to track the position of tools within the environment. To identify whether an operator is holding and using an object, a hand recognition method was developed to recognize their hands and also track their position. In this regard, there are two ways that can be achieved.

The first way is using the Mediapipe [6] hands library in python, where a hand finger and tracking solution is implemented. It employs machine learning to infer 21 landmarks, each corresponding to a different part of the hand (wrist, knuckles, fingers). The ML pipeline consists of multiple models working together: A palm detector that locates palms in the input image and a hand landmark model that operates on the cropped image region defined by the palm detector to return 3D hand keypoints.

The availability of hand landmarks proves to be extremely beneficial in determining the hand's location and its distance from the camera. This is primarily achieved by examining the wrist of the hands. First, the pixel coordinates of the wrist are compared to the bounding box of the detected tools. Second, the wrist's coordinates are matched to the ZED camera's point cloud and their distance from the camera is calculated and then compared to the distance of the tool. If the distances of both depth and pixel proximity are lower than a certain threshold, then the hand is close to the object.

A final check is then performed to determine if the hand is grasping the tool and not merely next to it. As it has been mentioned there are 21 hand landmarks and most of them correspond to a certain point in the operator's fingers. When the operator is making a grasping gesture, these landmarks are close to each other and it can be assumed that the hand is closed. In contrast, when they are further away the hand is open. Consequently, when it is established that the hand is in close proximity to the tool and has assumed a grasping gesture (closed hand), it can be inferred that the tool is being held.

This is showcased in Figure 9 and Figure 10.

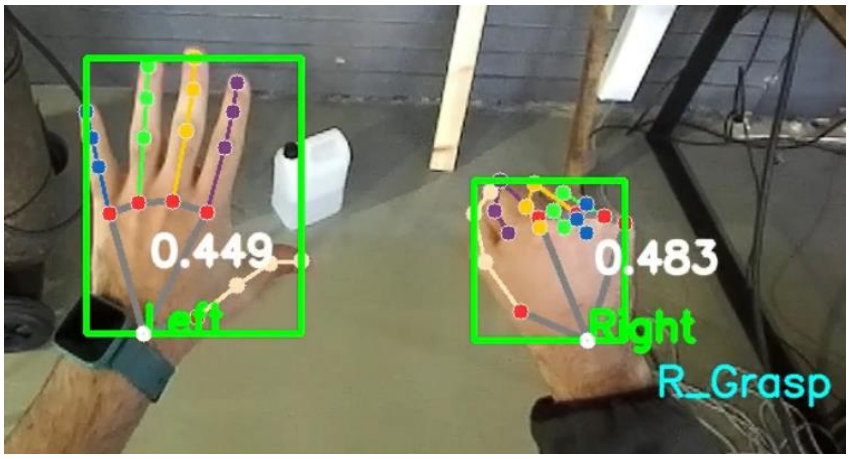


**Figure 9 Left hand - Open, Right Hand - Closed**

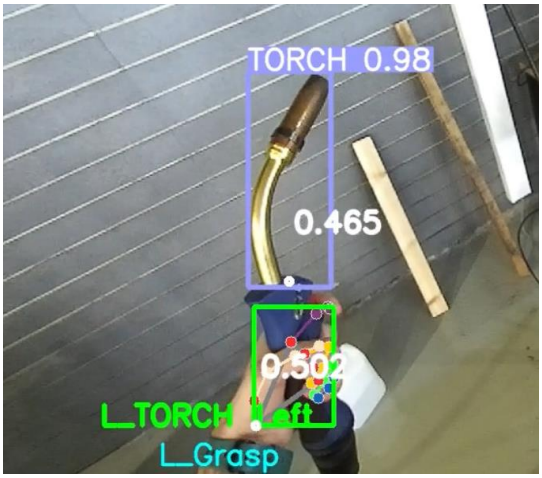


**Figure 10 Left hand holding a tool (Torch)**

The availability of the pixel coordinates of the hand landmarks facilitates the determination of a bounding box for the hand. This involves using the minimum x and y values of all the landmarks as the starting point for the box, and the maximum x and y values as the end point. This is particularly valuable as it enables the calculation of the distance from the camera to the tools, even in cases where they are obstructed by the hands. Therefore, as it has been described above, when the bounding box of the tool and the hand intercept, a different point inside the bounding box of the tool is chosen to be matched to the point cloud.

The second way of hand detection is using ArUco markers to determine the hands' coordinates. ArUco markers are a specific type of identifiers that can be detected and their coordinates calculated by the OpenCV Python library. Other types of markers could be used, such as QR codes. ArUco markers were chosen because their detection is fast, robust and simple.

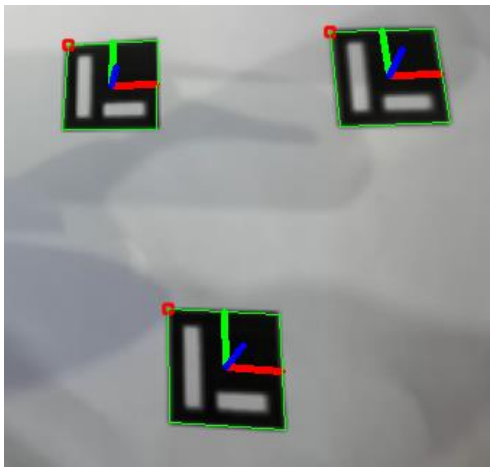

**Figure 11 ArUco marker OpenCV detection example**

The reason that a second type of hand detection is needed is because, in most cases, the operators are wearing gloves while executing their day-to-day tasks and this is mandated due to safety reasons. Unfortunately, gloves cannot be detected by the Mediapipe library, especially bulky ones where the fingers are not clearly discerned such as welding gloves. This problem is solved with the use of ArUco markers. The markers can be printed and glued directly on the gloves, or a band can be created with the markers on it and then be worn on top of the

gloves. Each marker has a unique identifier, and this is used for detecting whether the hand wearing the glove is left or right. For example, a marker with the id of 1 corresponds to the left hand and with the id of 17 to the right hand.

Once the pixel coordinates of the marker's center are calculated they can then be used to compare to the pixel coordinates of a tool's bounding box. The marker's coordinates can also be fed to the ZED camera's point cloud to calculate its distance from the camera. As with the Mediapipe method, once the distance thresholds are small enough then the hand is in close proximity to the tool. This time, however, there is no check for a grasping gesture as this method cannot calculate the 21 hand landmarks of the Mediapipe method. Once the hands are close to the tools then they are immediately considered to be grasping the tool. Bounding boxes also cannot be calculated. A gloved hand grasping a tool is shown in Figure 12.

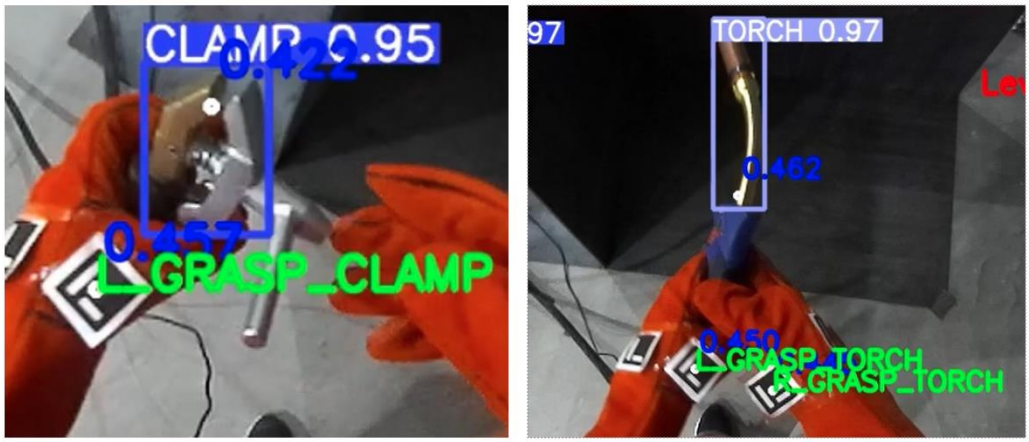


**Figure 12 Gloved hand holding tools (Clamp - Torch)**

Another advantage of the ArUco hand detection method over the Mediapipe method is that it is possible to detect the hands even when they are completely occluded by the object. Figure 13 shows an example of the left hand not being able to be detected with the Mediapipe method from this viewpoint while it is holding a bigger tool. In contrast, it can be detected with the ArUco method while the operator wears a band with the marker printed on it over the glove.

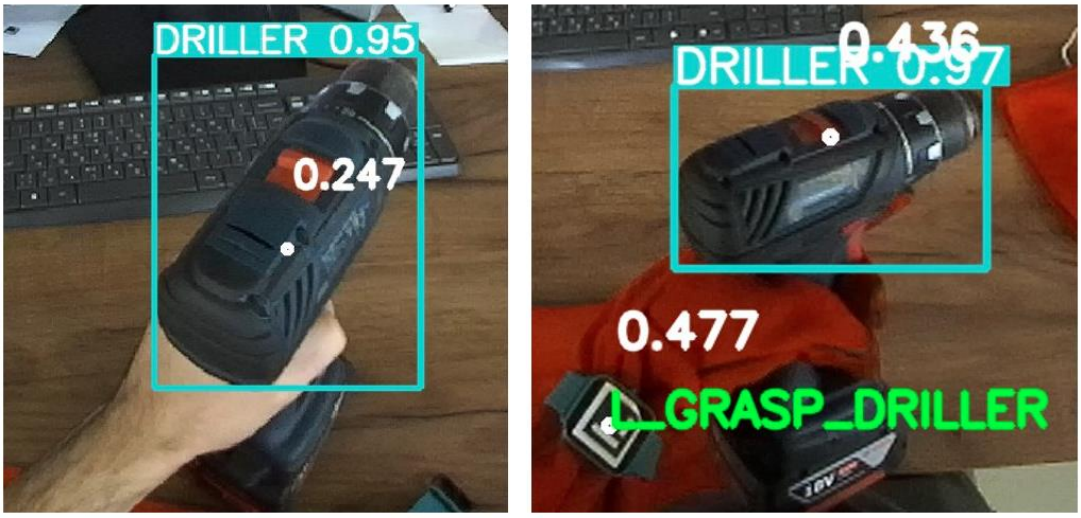


**Figure 13 Hand not being detected by Mediapipe (Left), Hand detected using ArUco markers (Right)**

In short, both methods offer distinct advantages and disadvantages with each one solving different problems of hand detection and tracking. In general, the Mediapipe method is preferred due to its ability to detect the grasping gesture, making sure that the operator is holding the object and not merely close to it, but the marker method is the one that is applicable in most cases where the operator would be wearing gloves.

### 3.5.2. Head orientation detection

There is another factor to consider when calculating the final LoA that HAP calculates. That is in what position the operator is holding the tool. Overhead tasks may require more effort as they apply more strain to the operator's hand and back. Tasks such as welding may be performed on the ceiling by holding the tool above their head or on a workbench in a normal position. When the task is performed by holding the tool overhead the LoA must increase for the position to be comfortable. The reason that the LoA cannot be increased to the maximum possible value that could be required is that higher values come at the cost of mobility. Operators generally prefer a lower LoA while executing their tasks when they are not strained, for example when performing welding on a workbench. Thus, they must increase or decrease accordingly.

To address this, a method was developed to detect when the operator is holding the tool in an overhead position or not. This is accomplished by detecting when the operator tilts their head upwards or downwards. By using the ZED camera's IMU sensor, changes to the camera's orientation can be calculated. Since the camera is mounted on the helmet this means that the head's orientation is tied to that of the camera.

By calling functions provided by the ZED SDK a quaternion is returned that represents the orientation of the camera. The roll, pitch and yaw can then be calculated. Specifically, the pitch factor is used to determine when the camera is tilted upwards or downwards. It is expected that the operator is looking at the tool while holding it, so if they are holding it upwards the camera is also tilted thus. After the calculation of the LoA based on other factors, a final check is done to determine whether the operator is looking up, down or have their head leveled (neutral position). Then a value is subtracted or added to the final LoA correspondingly.


**Figure 14 Operator looking Down (Left), Level (Middle), Up (Right)**

The left or right tilt of the camera can also be calculated. This is currently not considered for LoA calculation, as there is no use case for it yet, but this may prove a useful feature later down the line.


**Figure 15 Operator tilting their head**

### 3.5.3. Ladder climbing detection.

There is a specific subtask that can arise when executing other main tasks and that is ladder climbing. This is considered as a separate subtask because it forces changes when calculating the LoA.

As it has been mentioned, higher LoA values result in less freedom of movement and vice versa. While climbing a ladder the operator should have the maximum mobility allowed while wearing the exoskeleton. So, when the operator intends to climb one the LoA must immediately switch to the lowest value possible to allow them to do so. After the operator has climbed, they may want to execute a main task such as welding or cleaning. Then the LoA must switch to the appropriate value calculated by the algorithm. Once they are done executing their tasks the LoA must again switch to the lowest value to allow the operator to climb down.

To address this important subtask and its complexity, two different methods have been developed to detect ladder climbing. The first method is using QR markers mounted on the ladder. Two different QR makers are used

with specific identifiers, for example “Ladder Up” and “Ladder Down”. An OpenCV function is used to detect those markers. To avoid unintended activations from the operator just passing by a ladder and its QR marker the operator must be focused on them and this can be recognized with the focus detection algorithm described in the section below. Once the operator is focused on the QR marker at the bottom of the ladder, the LoA switches to the lowest value. The operator is then able to freely climb up the ladder. When they reach the top of the ladder, they must again focus to the QR marker at the top of the ladder; then the LoA is again able to freely change according to the factors mentioned so far. When they are done with their task, the reverse of this process is performed. The operator focuses on the upper QR marker, the LoA switches to the lowest value and they climb down, they focus on the lower QR marker and the subtask is considered finished.

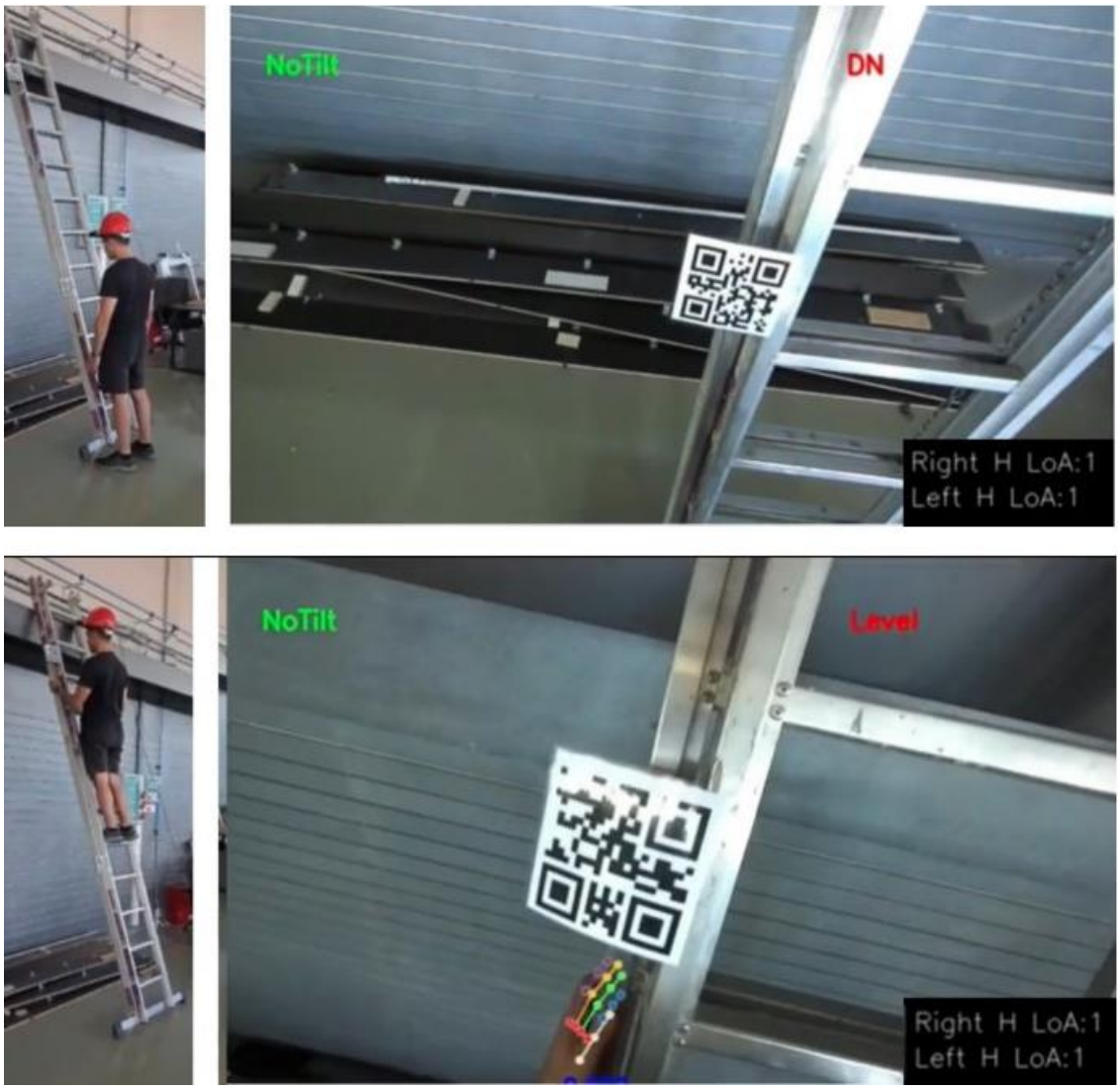


**Figure 16 Operator looking at QR markers.**

Along with this, a second, more novel method was developed to detect ladder climbing, again by using the ZED Camera’s IMU sensor. By using functions provided by the ZED SDK, the values of the altimeter can be obtained. This method does not require QR markers, instead it uses the same object detection method described in the above sections to detect the ladder in the environment. The task is initialized by the operator standing in front

of the ladder and holding it for a few seconds. The hands can be mapped to the detected ladder using the same hand mapping method developed for the tools.

Once the task has been initialized, the current altimeter value is saved. The LoA changes to the minimum value and the operator can start climbing the ladder. While climbing it the altimeter value increases and when they reach the top it stops. The operator is considered to have climbed and the LoA is free to change again to allow assistance for other tasks. Once they are done and they start to climb down the ladder, the altimeter value starts to decrease and this signals the algorithm to change the LoA to the minimum value. When the operator climbs down, the altimeter value is the same as the initial one and this means that the operator has finished climbing.


**Figure 17 Detecting operator climbing using the IMU sensor**

As with the two hand detection methods here there are also advantages and disadvantages to each one. For the QR marker method the advantage is that it does not require training the object detection model for the different types of ladders the operator may encounter. The disadvantage is that it requires for the markers to be placed on every ladder that could be used.

As far as the altimeter method is concerned, the advantage is that it does not require the placement of all those markers and it is more elegant. The disadvantage is more complex and requires further explaining.

When the HAP calculates a LoA value and communicates it to the exoskeleton, it will take some time for the torque to adjust. This is not an issue for the QR marker method as the operator can be asked to wait for a few moments before climbing after the QR has been recognized. This is also not an issue for the altimeter method when the operator is climbing up for the same reason, they could wait after initialization. The problem arises when the operator intends to climb down. Once up the ladder, for the altimeter value to decrease, the operator must already be climbing down, and this does not allow enough time for the LoA to change. In the meantime, the operator may have already completed the subtask. Consequently, until this issue is resolved the QR marker method should be used moving forward.

### 3.5.4. Human focus detection

A method for detecting human focus was developed to distinguish between when the operator is performing a task and when they are simply moving around the environment. It is assumed that during a task, the operator will maintain focus and stop moving their head, allowing for their intention to be inferred.

Focus detection can also act as a signal to wait for a specific trigger. In the welding use case example, it was mentioned that the trigger to consider welding to be taking place is to calculate the mean average values of the image. This must not be done constantly for two reasons. First, because it consumes computational resources and may slow down the system. Second, because there may be other reasons for sudden change in lighting in the image such as being in a dark environment and turning on a flashlight. Thus, the process of initialization takes also into account whether the operator is focused or not. The same logic can be also applied on other tasks such as ladder climbing to help recognize when the operator intends to climb and not just passing by a ladder.

As mentioned in the head orientation section, the camera's pose can be calculated. Then, if the quaternion values or roll, pitch, yaw values do not change beyond a certain threshold for several seconds, it is assumed that the operator is focused on a task, that is, the camera and their head remain relatively still.

## 3.6. Level of assistance adjustment

### 3.6.1. Level of assistance adjustment algorithm

Up to this point, the individual technologies used to decide upon what the operator is doing or intends to do have been described. These features need to work in tandem to calculate the appropriate LoA. To ease the explanation of such a complex algorithm, Figure 18 shows how the LoA is calculated in different scenarios.

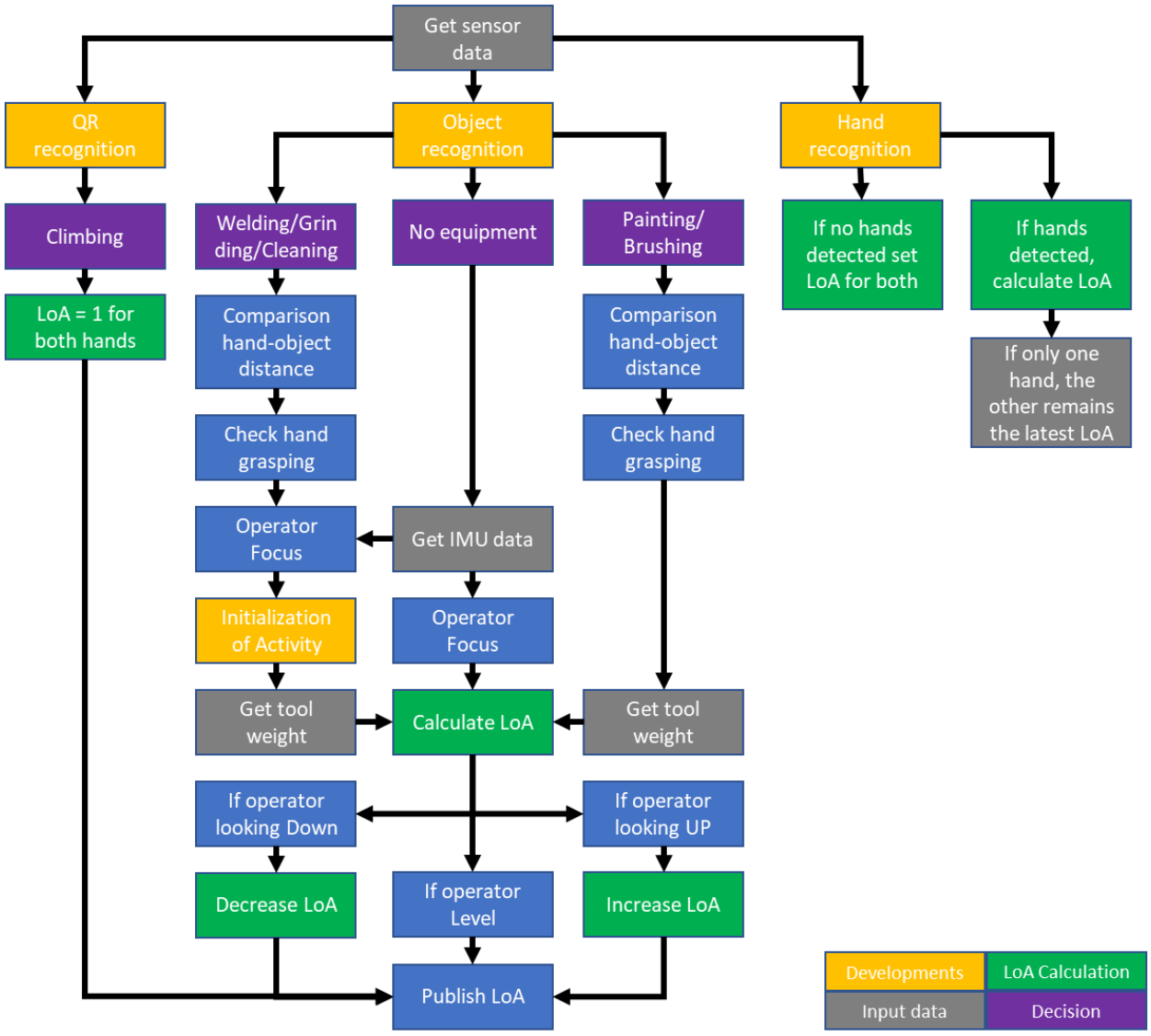


**Figure 18 LoA calculation graph**

In most cases, the LoA is calculated with the following logic. First, detect the tools in the environment and determine what task the tool is used for. Second, get the tools weight to increase the LoA accordingly. Third, detect the operator's hands and determine if they are holding the tool. If the operator is focused, wait for a case specific activity trigger. Finally, after the LoA has been calculated by all other factors, determine if the operator is looking down or up while performing the task. If they are looking up, then this means that they are holding the tool in an overhead position so the LoA must be increased. If they are looking down, then the execution is more comfortable and the LoA decreases to allow for more freedom of movement. For other tasks such as painting or brushing the tool can be detected but no specific trigger for initialization is needed. Even though, LoA calculation follows a similar logic.

Even if no equipment is detected, the operator may be performing a task without a tool, for example cabling. When this is the case, the operator is still expected to be focused and their hands detectable. Then an appropriate LoA is calculated.

The diagram depicting the logic applied for the ladder climbing sub task concerns the QR code method that is currently being used. When no tools are detected, the LoA for each hand is set to a pre-determined value. The stress based LoA provided by a different module is expected to influence its changes. All the diagrams assume that the Mediapipe method is utilized. However, if the ArUco marker method is preferred, the same process will be applied, except for the grasping gesture detection.

### 3.6.2. Power consumption feature

The exoskeleton is semi active which means that it has adjustable torque that results in different LoA provided. However, it is not wise to constantly change the LoA as each time it is adjusted it consumes power. Since this is a portable device, it is of outmost importance to conserve energy as much as possible.

During the operation of the HAP algorithm, it continuously determines the necessary LoA based on the latest data. Whenever an operator picks up a tool, even briefly, a new LoA is calculated. However, updating the exoskeleton with each change in LoA would cause frequent fluctuations in torque, which is not ideal for energy conservation. Instead, the LoA should be calculated as usual, but changes should only be requested after a certain agreed-upon time interval. The algorithm accomplishes this by averaging out the LoA values calculated over this period and then transmitting them to the exoskeleton. This method reduces the number of changes made and conserves power.

## 3.7. Hardware integration

### 3.7.1. Zed

The developments described in previous sections were based on portable solution that will be mounted onto the operator using a jetson based embedded system. The zed camera will be connected to the controller, were all the data processing will occur. The portable solution is powered by a power bank while the communication with the exoskeleton will take place using its wifi adapter for wireless communication, without any medium such as router.

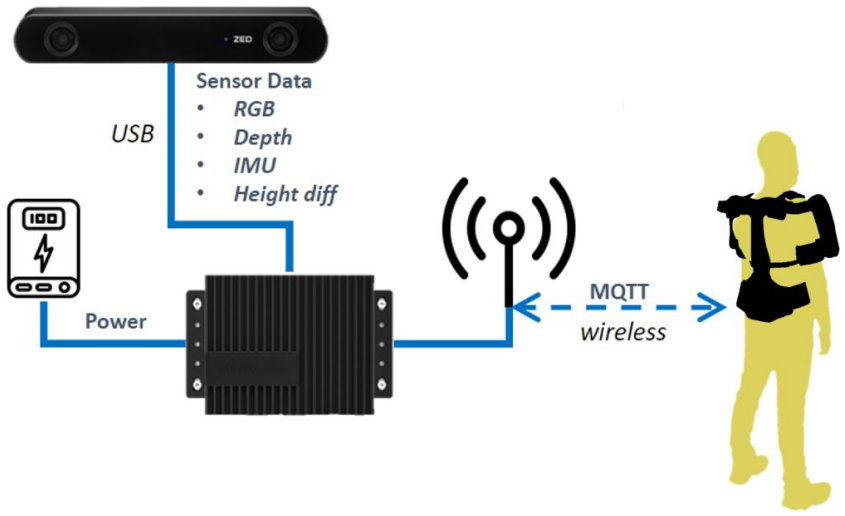


**Figure 19 Zed based approach h/w interconnection.**

### 3.7.2. HoloLens 2

Besides the ZED camera, the HoloLens 2 augmented reality headset was tested.

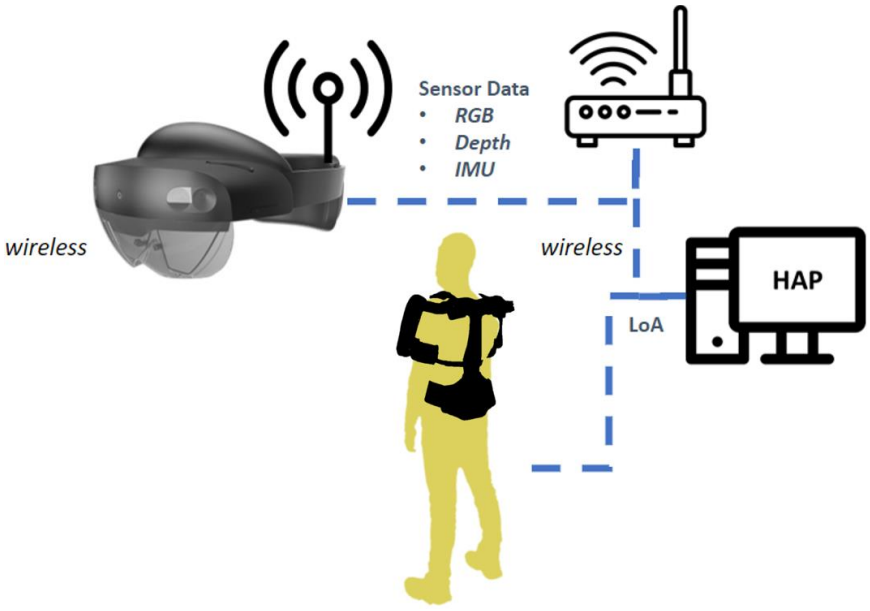


**Figure 20 Hololens 2 based approach h/w interconnection.**

As it has been established above, there are three sensors that are required of a camera/headset in order to implement the functionalities of HAP. These sensors are the following:

- An RGB sensor, preferably with a wide FoV
- A Depth sensor
- An IMU sensor with a Gyroscope
- Preferably a barometer/ altimeter to implement the ladder climbing subtask detection without the use of markers.

HoloLens 2 is a wearable that includes a RGB camera, depth sensors and an IMU sensor. It also includes its own hand detection method that outputs a point for each joint (finger tip, finger middle, etc.). To test its suitability as an alternative to the ZED camera, the HAP functionalities were implemented. These include object detection/ tool recognition, hand detection and left/right hand distinction, hand mapping to tools, Up/Down look and focus detection.

The high-resolution RGB sensor that is utilized for object detection and marker recognition, provides a clear image with better colour accuracy, exposure and less grain than the ZED camera. This can aid in object recognition by providing more information for inference. However, the RGB sensor's Field of View (FoV) is

significantly smaller than the ZED camera's, resulting in some objects or the hands holding them not being detected in situations where they would be detected by the ZED camera.

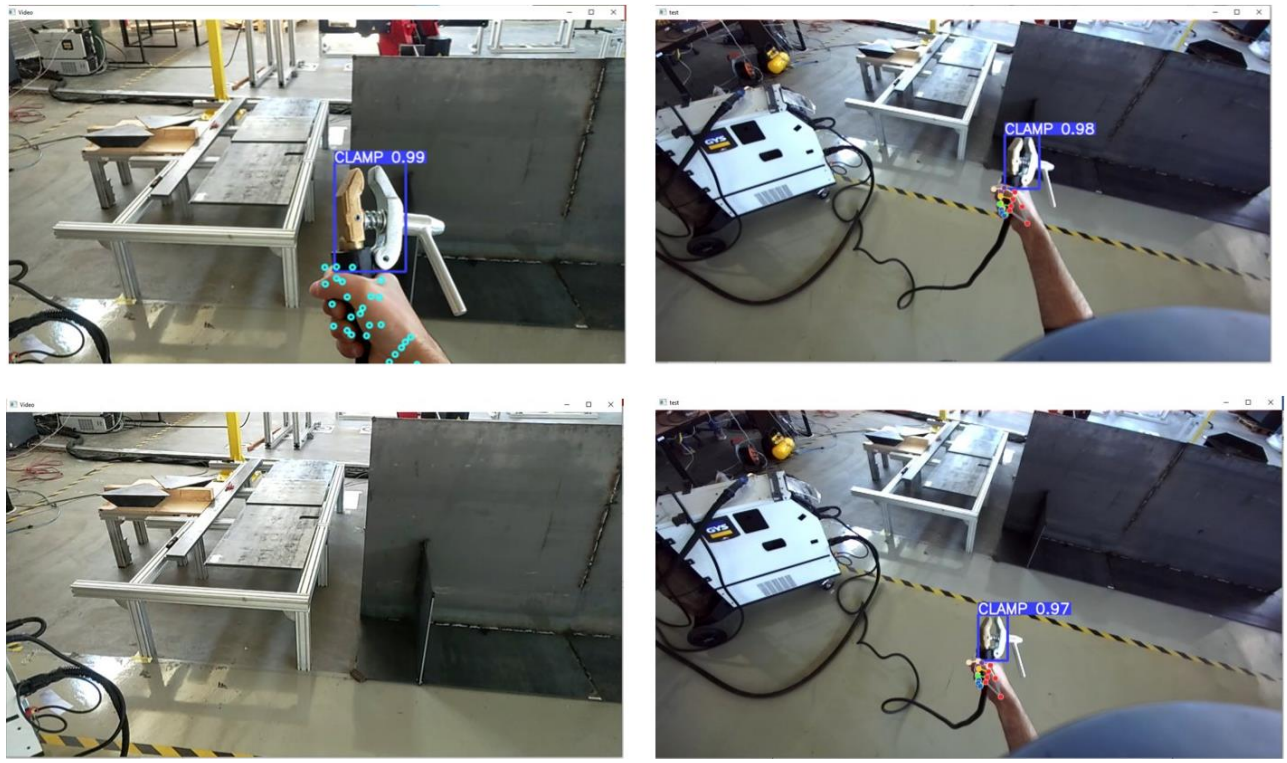


**Figure 21 Left - HoloLens view / Right - ZED view**

The depth sensors can be used for obtaining the distance from a specific point in the image with the same methodology described above. Nevertheless, the senor's resolution and FoV is much smaller than the ZED's camera. Aligning the depth and RGB images, a process required for translating the pixel coordinates of the objects detected in the RGB image, reduces the resolution and FoV of the RGB sensor. Therefore, obtaining the distance of an object is not feasible using the HoloLens 2 headset.

The IMU sensor works as expected when detecting the headset's and thus the wearer's head pose. It can accurately detect when the wearer is looking Up or Down and if the pose does not change for a certain period (e.g., three seconds) it can be assumed that the operator is focused.

HoloLens 2 does not include an altimeter sensor so the ladder climbing subtask detection must use the QR marker method. This is not necessarily a disadvantage over the ZED camera as the marker method works better in some ways as described in the corresponding section.

Additionally, HoloLens 2 offers its own hand detection method which allows for gesture detection (grasping) and calculating the hand's distance from the tool (in pixels as there is no distance from camera detection). This method works as well as the one offered by Mediapipe library, and this reduces the complexity and the requirements of the algorithm.

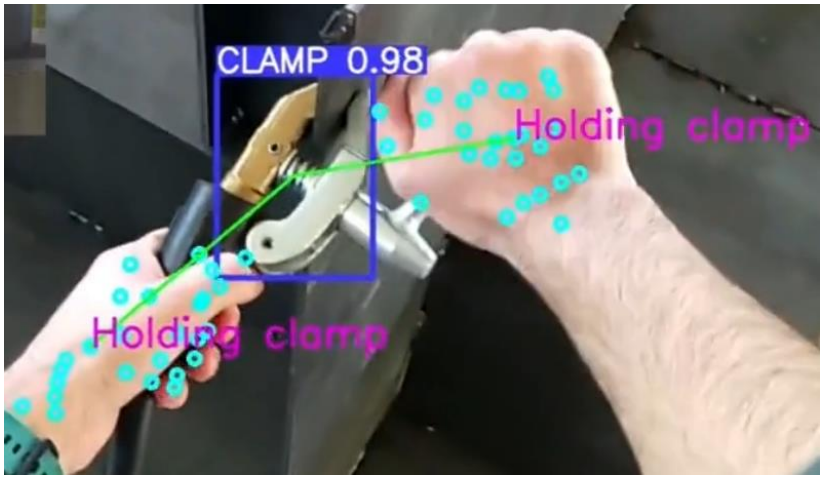


**Figure 22 Example of hand to object mapping using HoloLens**

A clear advantage of the HoloLens over the ZED camera is that is a computing machine on its own. This means that features such as depth perception and hand detection do not require an SDK and they are run by this system. This could reduce the load on the computer that runs HAP. Moreover, it can be powered by its own battery, and it does not require a cable for connecting it to a PC as its data are transmitted wirelessly.

Overall, while the HoloLens 2 headset seemed promising, its disadvantage on FoV, distance detection and lack of barometer do not make it a suitable alternative to the ZED camera. Its build poses some challenges as it is not meant to be mounted on a helmet or worn on top of a welding mask. Its price could also be considered as the ZED camera costs considerably less.

# 4. CONCLUSIONS AND FUTURE OUTLOOK

In this document the description of the initial protype of the HAP module was presented. The three main features of the solution consist of the object detection pipeline to locate parts in 3d space, a human action recognition pipeline to evaluate human actions and last but not least a LoA adjustment algorithm.

As for the next period, the plan is to finalize the integration and adaptation activities with the exoskeleton, the training of the hap module with material provided by the end users as well as deploy and test the integrated solution at end user premises, evaluating the effectiveness of the HAP solution.

# 5. REFERENCES

[1] G. Jocher et al., “ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference.” Zenodo, Feb. 2022. doi: 10.5281/zenodo.6222936.

[2] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14 (pp. 21-37). Springer International Publishing.

[3] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

[4] Manettas, C., Nikolakis, N., & Alexopoulos, K. (2021). Synthetic datasets for Deep Learning in computer-vision assisted tasks in manufacturing. Procedia CIRP, 103, 237-242.

[5] Community, B. O. (2018). Blender - a 3D modelling and rendering package. Stichting Blender Foundation, Amsterdam. Retrieved from http://www.blender.org

[6] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Lee, J. Mediapipe: A framework for building perception pipelines. arXiv 2019. *arXiv preprint arXiv:1906.08172*.

[7] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.