ACROBA

connect & produce through agile production

D4.4 Report on integration of the ACROBA platform to specific use case

WP4 (public version)

AITIIP

2023/06/02

Dissemination level: PU

Version V1

aitiip centro tecnológico

The ACROBA project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017284.

Approval Status

	Name and Surname	Role in the Project	Partner(s)
Author(s)	Iván Monzón	Task Leader	AITIIP
Reviewed by	Alejandro Muñoz Espiago Aly Magassouba	WP4 Leader WP3 Leader WP2 Leader	SteriPack IMR SIGMA
Approved by	Norman Baier	Project Coordinator	BFH

History of Changes

Version	Date	Description of Changes	By
0.1	15.04.2023	First version	Iván Monzón (AITIiP)
0.2	27.04.2023	Points distribution	Alberto Laguia (MOSES)
0.3	30.05.2023	SteriPack Contribution	Franck Petit-Renaud (SteriPack)
0.4	02.06.2023	CABKA Contribution	Sergio Picazo (CABKA)
0.5	08.06.2023	MOSES Contribution	Alberto Laguia (MOSES)
0.6	14.06.2023	AITIIP Refinement	Iván Monzón (AITIIP)
1	30.06.2023	Final Revision	Management Team

Disclaimer:

Table of Contents

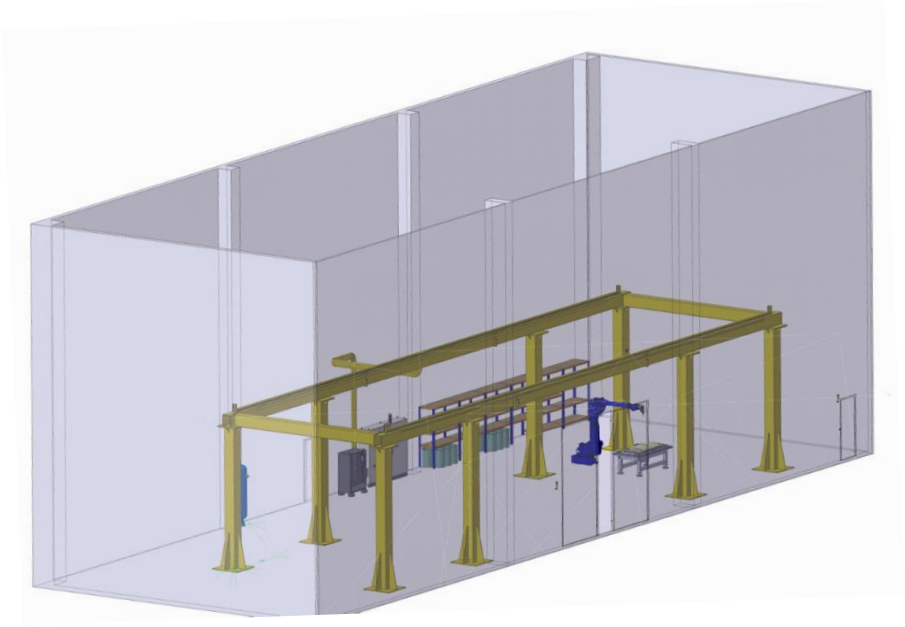
List of Tables

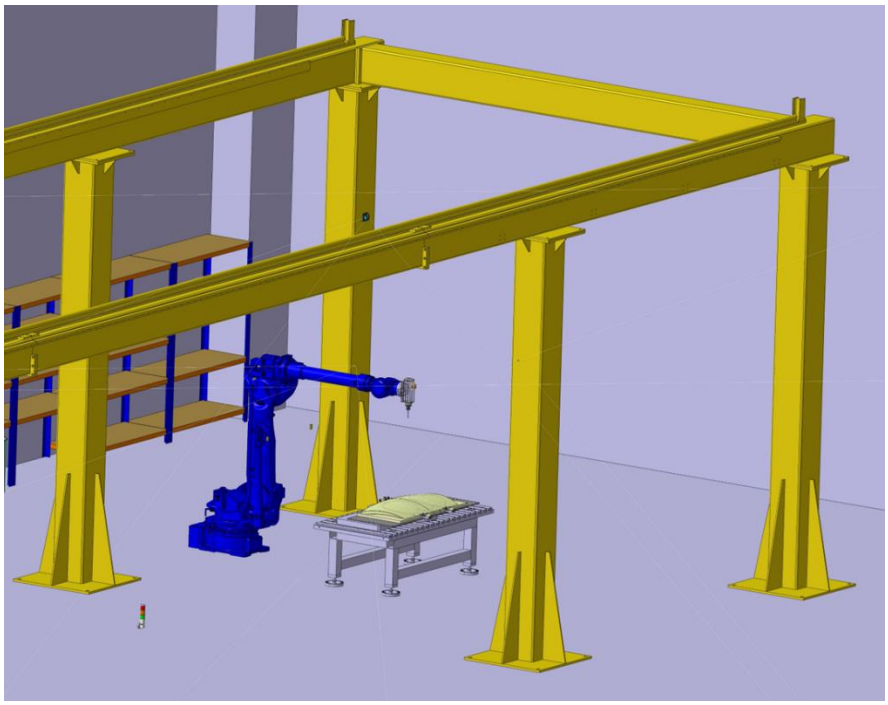
Executive Summary

Having defined the generic cell in WP3, the legacy equipment had been upgraded to integrate the ACROBA platform. A set of preparatory steps has been carried out prior to deployment, including mechanical, pneumatic, and electrical engineering, as well as the creation of drawing plans. Specific ROSE-Aps have been implemented within the factory machinery to support data acquisition and integration, while equipment providing perception and cognition modules had been put in place. The CAD models have been integrated into reinforcement learning modules, and a sensorized dummy tool specific to the use case at hand had been designed. A CPS set-up have been established, tailored to the specific use case. After the completion of the task, the generic cell has been adapted to the specific task at hand and has been operated autonomously for a period of time with different and previously unseen parts.

Design Cell

MOSES





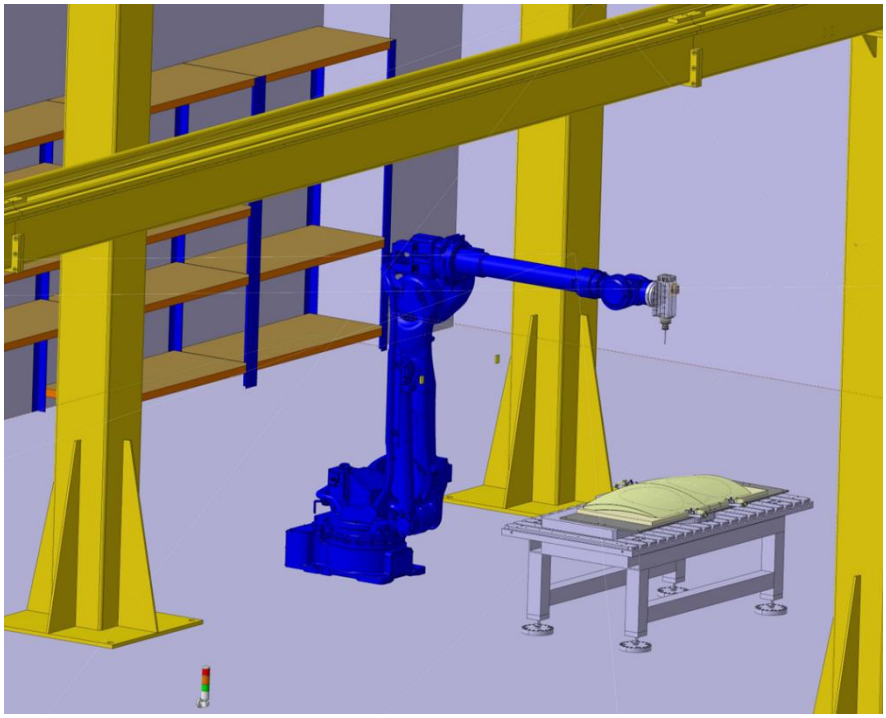


Figure 1 Moses Cell Upgrade

The main purpose of the MOSES cell is the cutting (by means of subtractive techniques) of container lids to provide them with the necessary pockets for the introduction of waste into the containers (see MOSES). Within the types of bin lids there is a great variety depending on the type of waste, as well as the size of the bin and its colour (the colour is referenced to the type of waste to be introduced). In this case, the robot executes previously programmed trajectories depending on the factors mentioned above. The purpose of the ACROBA project for MOSES is the creation of a flexible system that allows the trajectories to be adapted in a simple way to the container lid to be cut, trying to optimise the process and with the aim of allowing operators to change trajectories without the need to manipulate the robot's low-level programmes.

The MOSES cell is composed of a YASKAWA GP180 robot, which has an electro spindle anchored to its wrist, which allows it to perform each of the cuts necessary for the finishing of the container lids. In addition, this cell has a pneumatic system that allows the clamping of the lids to counteract each of the cutting forces generated during the operation.

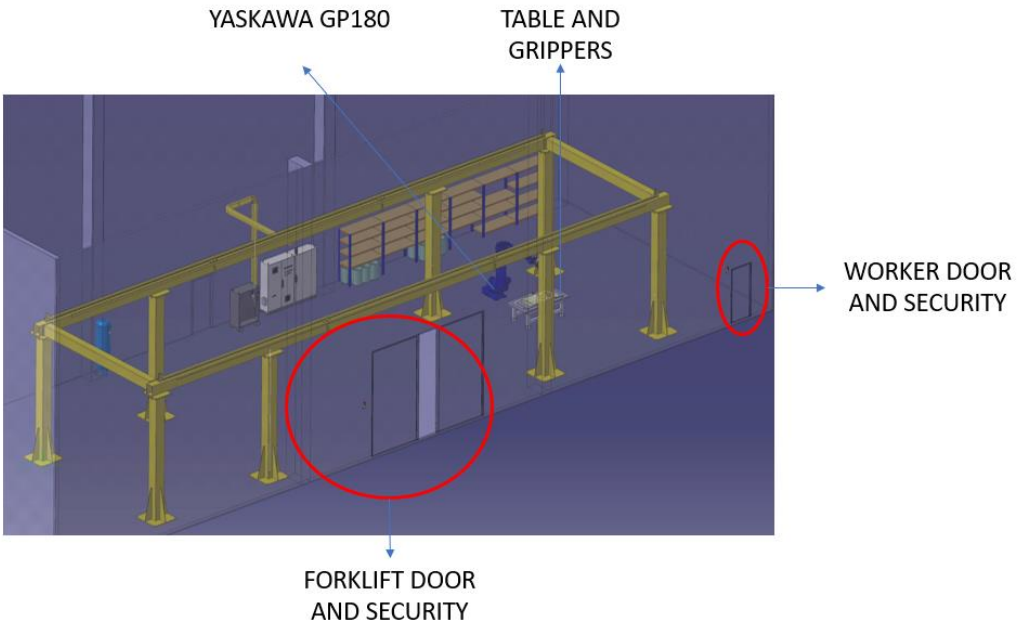


Figure 2 General overview of the MOSES cell.

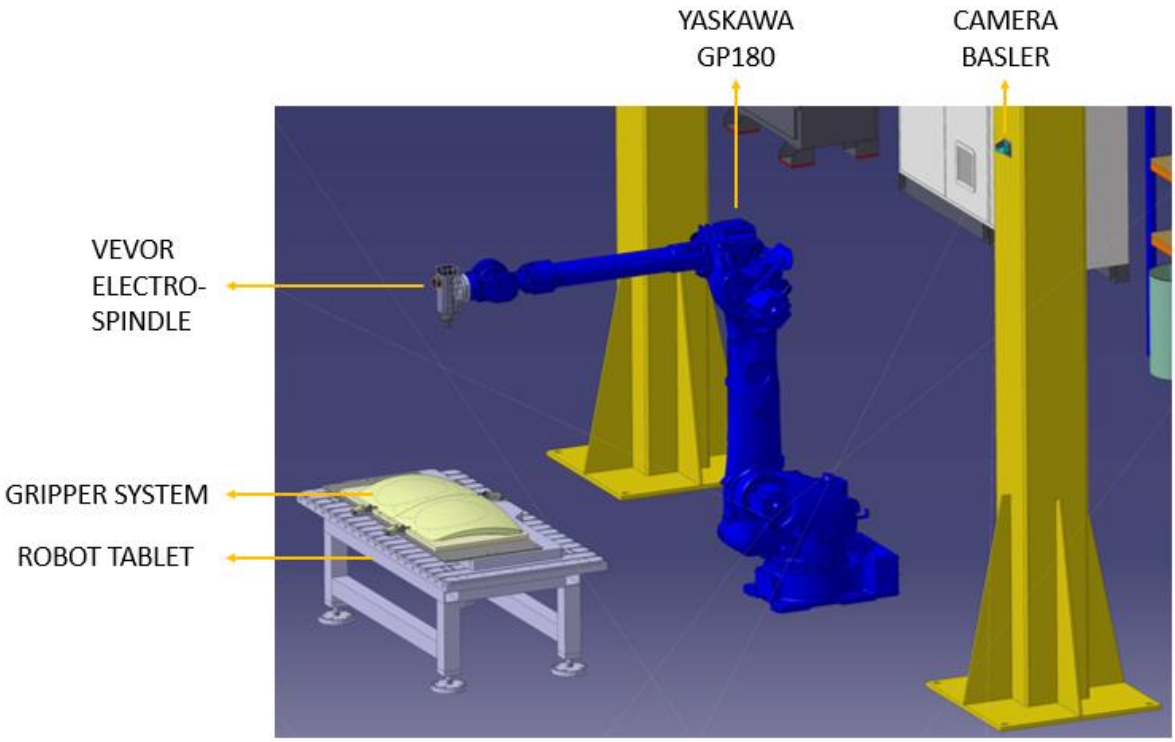


Figure 3 Detailed view of the MOSES Cell

Figure 2 and **Figure 3** show the main configuration of MOSES cell. **Figure 2** shows a general overview of the cell. In this figure you can see each of the entrances to the cell (entrance for workers and entrance for forklift), and both doors have the necessary safety devices installed to prevent possible personal injury. They have an audible-luminescent safety device to indicate the state of the robot (start-stop-error) and another inductive safety device that pauses the process (stops the robot and the spindle) if any of the doors are opened. In addition, the figure shows the location of the robot and the worktable inside the cell. **Figure 3** shows a more detailed view of the cell, showing the position of the robot with respect to the worktable and one of the monitoring cameras. Also, this figure shows the gripper system for anchoring the lids to be cut, generally this system works with a pneumatic system. Following table summarize all the devices installed on MOSES cell.

Table 1 MOSES Cell main components.

Device	Quantit	Name	Specifications	Type	Description
Robot	1	Yaskawa GP180	GP180+YRC1000 Controller +AI card +DOI cards	Purchased	120 kg payload to support the cutting head and to limit vibrations and inertias produced during cutting
Spindle	1	VEVOR JST-JGF-4KW-ER25	220V, 4Kw, Aircooled, 18000rpm	Purchased	up to a maximum of 18000 rpm it allows the use of tools from 6mm to 12mm in diameter.
Flange	1	Steel flange for electro-spindle anchoring	Steel F114	Milled	Robotic flange made ad hoc in high-performance steel to withstand the vibrations caused during the cutting process
Table	1	Steel table for tooling placement	Steel F114	Milled	Multipurpose work table, T-slots for anchoring any type of tooling.

					Manufactured with high performance steel.
Pnaumatic grippers	3	DESTACO 802-U-LC	OUT 490N	Purchased	Pneumatic Gripper to assure the support of the parts in the tooling. 490N load capacity when the inlet pressure is 5bar
Camera	3	BASLER ACE acA5472-5gm	20MPx + BASLER lens C11-1620 -12M-P 90º horizontal field	Purchased	Cameras for tracking processes in which dummy tools are used.
Light towers	2	Schneider Electric Harmony XVC6	24V Ac/dc, three LED red/ green/orange	Purchased	Visual and audible signalling of machine status. Machine status information at all cell access doors.
Inductive safety sensors	2	SICK RE13-SA05	24V DC	Purchased	Active safety of open/closed door control. Access detection and process pause.
Monitorin cameras	3	HIKVISION 4MPX AcuSense Fixed Dome DS-2CD2146G2-IF2.8	4MPX, omnidirectional camera	Purchased	Security camera for cell control, as well as process monitoring.
Comunication board	2	HHC-N-8I8O	9-24V IN, 8 relay OUT, 8 DIN	Purchased	security cards for door control, security light towers...

Mechanical aspects

Among the mechanical aspects carried out in the integration of the MOSES cell, two systems are worth highlighting. The first is the mechanical anchoring to the floor of both the robot and the table, by means of screws that are valid for fastening metal elements to the industrial floor.

The installation is basic, first the systems are placed in their definitive location and then a hole is drilled to introduce the special screws for industrial floors. All these actions must ensure that both the robot and the table are perfectly anchored to the floor, transmitting directly to the floor all the stresses generated during daily cutting operations.

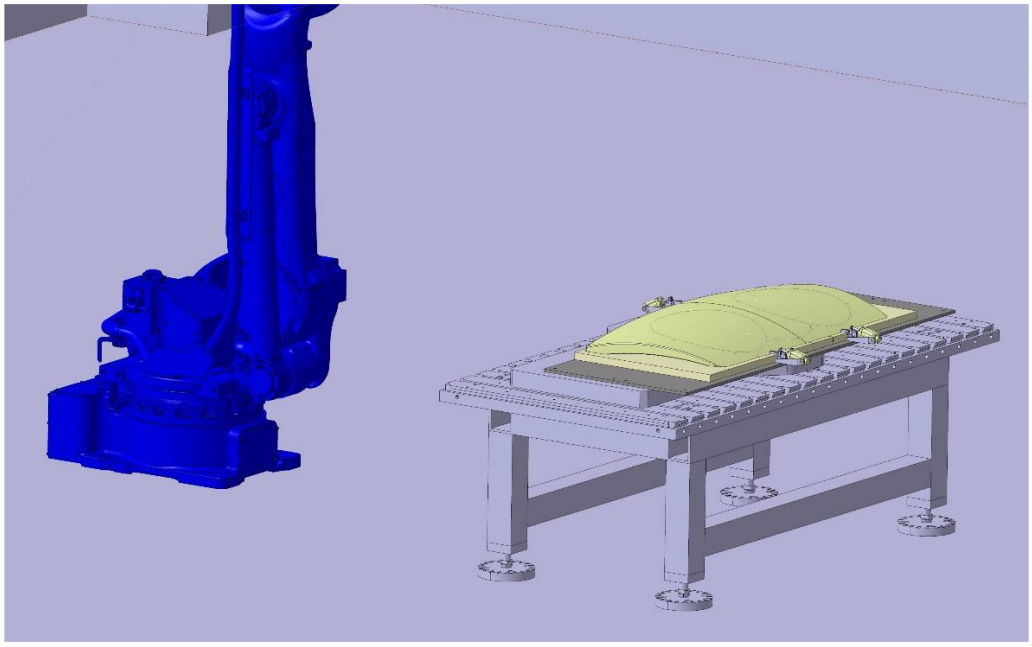


Figure 4 Mechanical anchoring for the robot and worktable to the industrial floor.

Figure 4 shows a detailed view of each of the anchorages made to the industrial floor, both for the table and the robot.

The work table is made up of three structures (hard steel structures), the support system for the tools required for cutting each of the covers (flat plate with T-shaped slots to give versatility to the system by having a large number of anchoring areas), the support system (reinforced metal structure that gives rigidity to the entire table) and the support system (system based on spherical supports that allows tables to be interchanged without the need to remove the screws fixing them to the industrial floor). This workbench has been entirely designed and manufactured to facilitate the flexibility that the system will be provided with after the installation of the ACROBA software. **Figure 5** shows a detailed view of the worktable.

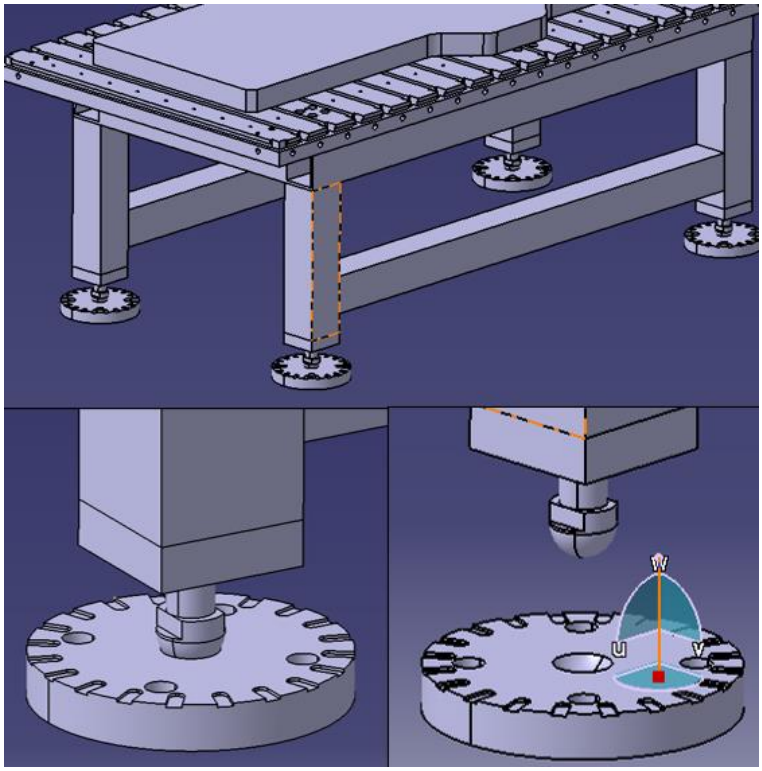


Figure 5 Worktable details.

The other mechanical system included in this development is the flexible coupling for the cutting head. This system is made up of 2 steel metal sheets, coordinated with each other by self-locking turning systems, which allow the cutting head to always be positioned in the same position, thus improving the versatility of the robot, being able to exchange this head for other heads with different purposes (manipulators, meters, welders...). **Figure 6** shows both pieces of the system, as well as each of the couplings between the robot and the cutting head. **Figure 7** shows a detailed plan for the manufacture of each one of the parts, in which all the operations to be carried out to obtain the parts are specified. This tooling is made of high-quality steel and is responsible for absorbing all the vibrations generated during the normalization operation of the robot.

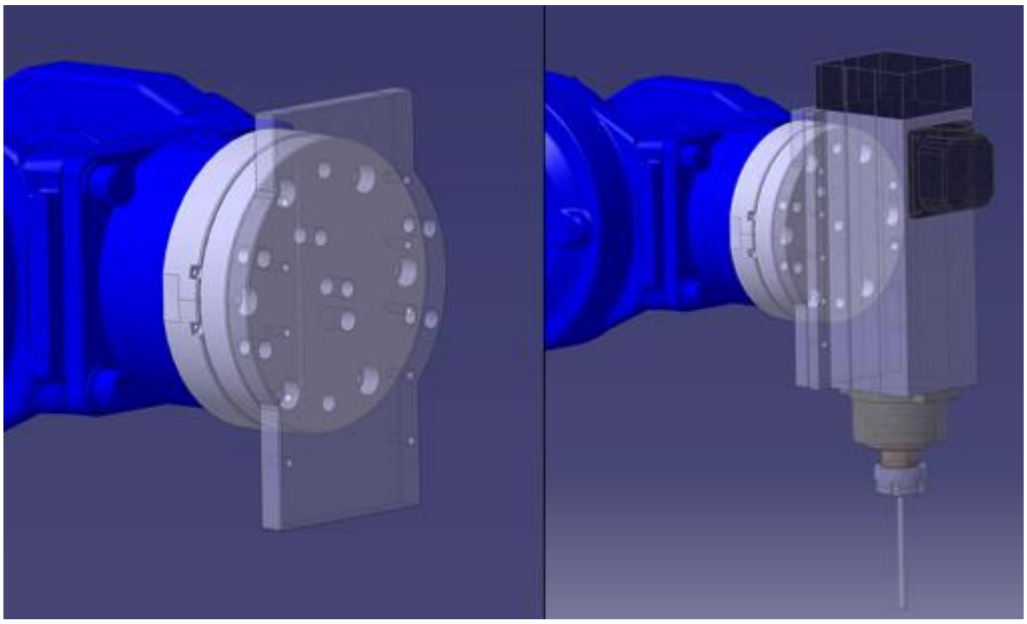


Figure 6 Anchoring system for the electro-spindle head.

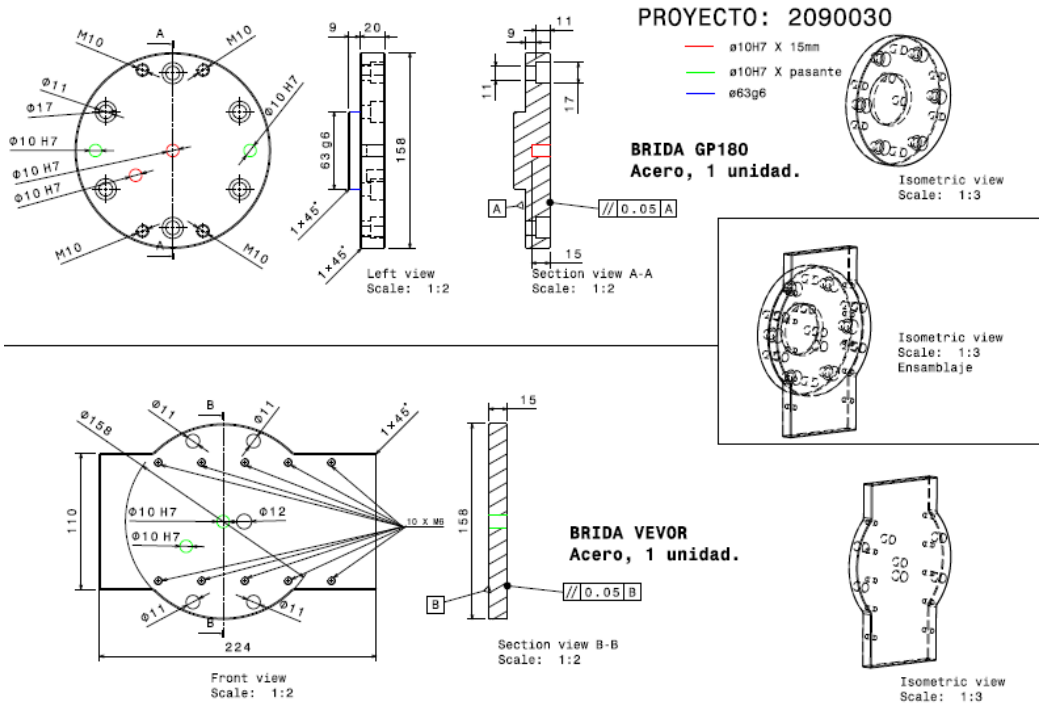


Figure 7 Drawing for the electro-spindle anchoring system.

Electric distribution

In addition to all the common electrical connections required by a robotic cell (robot power supply, sensor power supply...), for the ACROBA-MOSES system, several additional systems have been considered to make the cell as safe and flexible as possible.

On the one hand, and as will be discussed below, the installation of 12 sockets for ethernet connections has been carried out, in this case, each of these sockets has the POE system that allows to feed each of the devices through the RJ45 connections. These sockets are intended to establish TCP/IP communications between all the cell systems and the ROS central computer, as well as providing power to the systems that require it (monitoring cameras, dummy tool cameras...).

The following figure shows the distribution of each of the sockets (RJ45) available in the cell, as well as all the implementations made through fibre optic communications, in order to increase the data traffic and make all the cameras work at maximum speed.

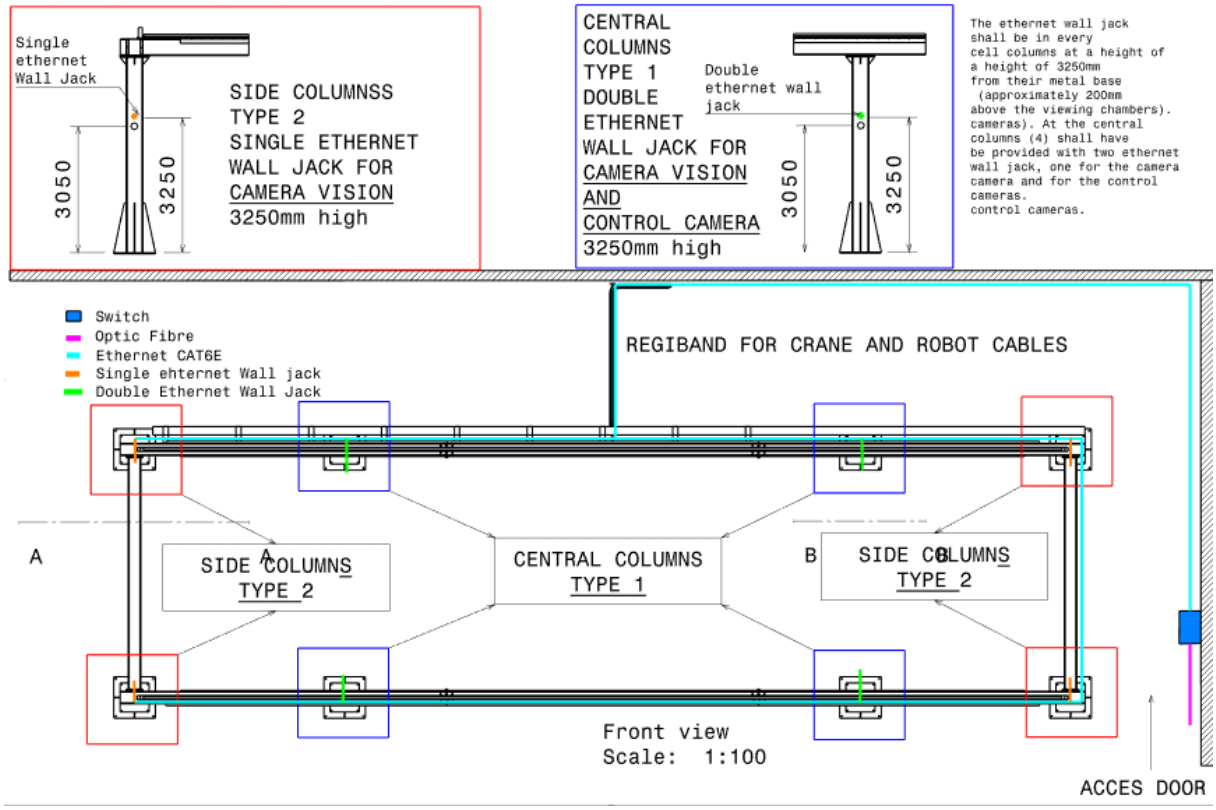


Figure 8 Drawing for the ethernet wall jack distribution.

So far, 3 BASLER cameras for the generation of trajectories provided by the Dummy Tool have been installed and connected to the sockets mentioned above. At the same time, 3 monitoring cameras (security and control systems) have been installed to always control the movements of the robot. In addition, and with the purpose of improving the data captured by the BASLER cameras, LED lights have been installed above the cameras, improving the sharpness of the images, as well as avoiding the generation of dark areas.

Figure 9 Tree Basler cameras with three LED lights to stablish the control system.

An external input/output board is used to manage the switching on and off of each LED spotlight, as well as to manage the safety and signals provided by the Dummy Tool. Basically, this board is responsible for managing the operation of the security light towers, inductive door opening sensors, enabling the recording of the BASLER cameras, as well as managing switching on and off each of the LED lights.

Figure 10 TCP/IP board.

Finally, another electrical aspect to consider is the Electro-spindle, an electrical device composed of a spindle (rotating tool-holding head) and a driver to control its rotation. In this case, the communication between the driver and the robot has been made by direct communication between driver and robot, through digital signals.

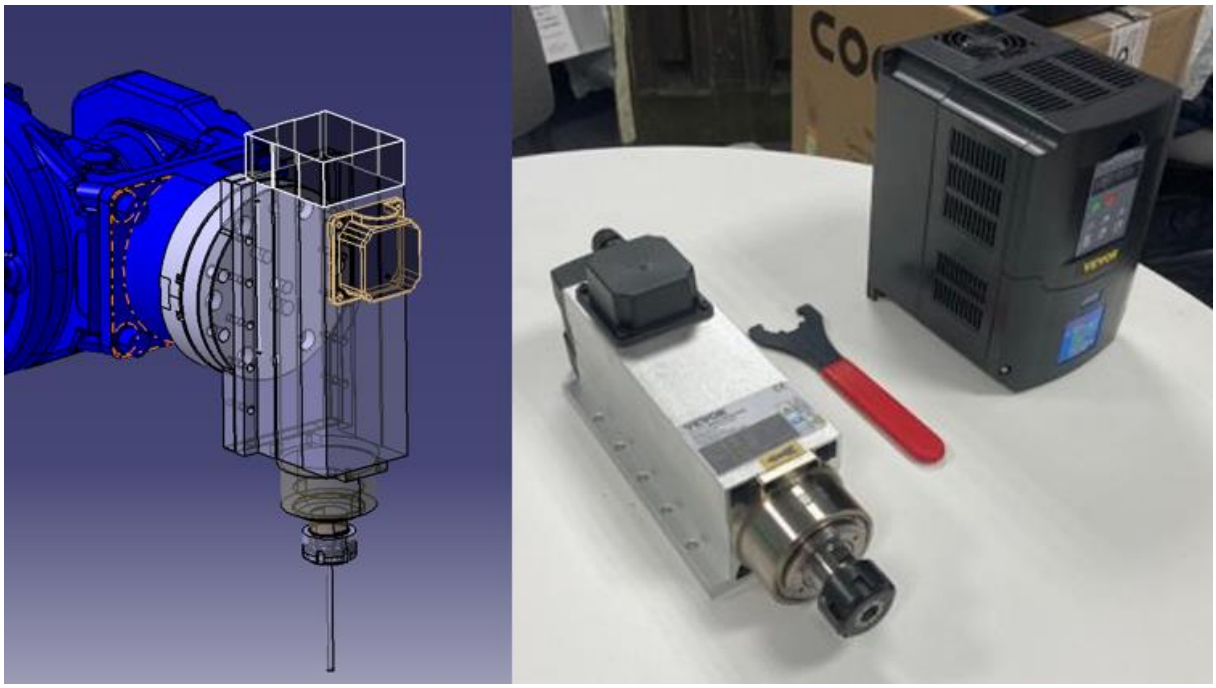


Figure 11 VEVOR electro-spindle.

Pneumatic assembly

In this case, the MOSES cell has a very simple pneumatic system, using a 24V relay, connected to a DOUT of the robot, the system enables or disables the grippers for securing the container lids. These grippers are made up of small pneumatic cylinders that ensure the correct position of the cover in the cutting tool, they also have sensors that indicate to the robot the position in which they are (closed or open) and serve as a starting point for the initialization of the process. If the sensors do not detect the closed position (this means that the container lid is not in the correct position) the robot will not execute the trajectory until the operator places the lid in the correct position.

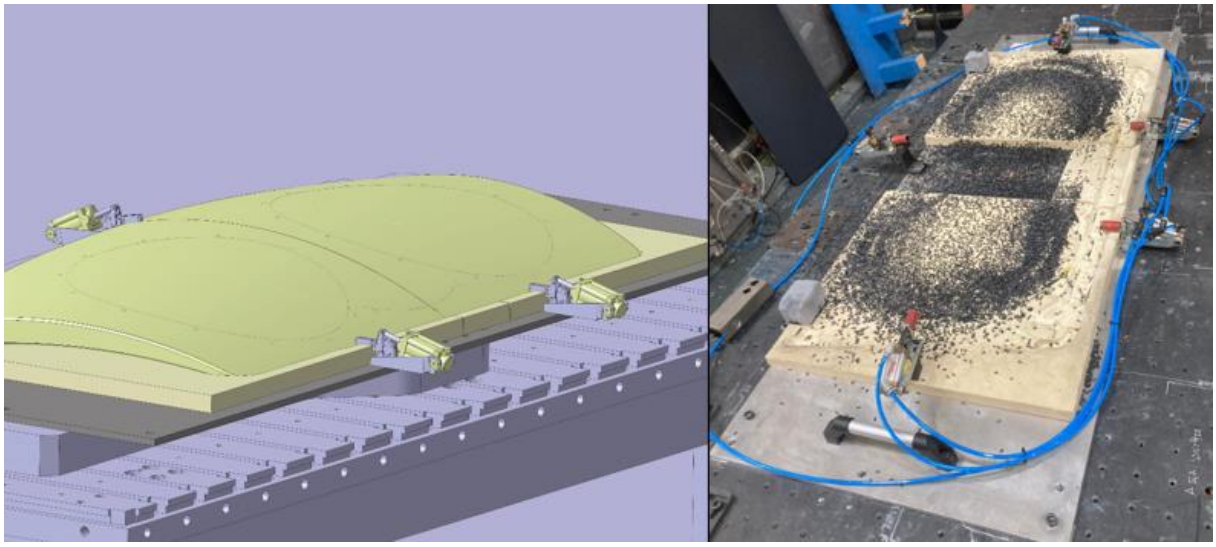


Figure 12 Grippers for ensuring the position of the container lid on the tooling.

Safety aspects

To contribute to human-machine and machine-machine safety, several systems have been implemented in this cell capable of detecting any unforeseen event that could harm humans and cause material damage. Firstly, the cell has a complete enclosure (see **Figure 2**) that prevents the entry of any user unrelated to the work to be carried out. This enclosure has two entrances, one for the entrance of the operators and another, larger, for the entrance of the forklift to manage the container lids. Both entrances have inductive security systems that if any of the doors are opened (without the user's request) the system pauses, stopping the movements of the robot and stopping the spindle rotation.

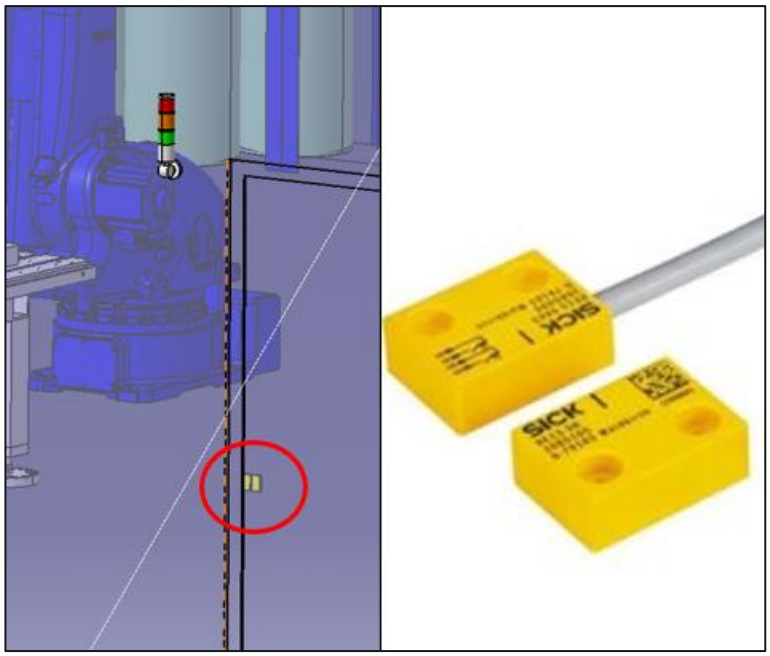


Figure 13 Inductive safety sensors for the doors of the MOSES cell.

In addition, the system has security light towers (see **Figure 14**) that indicate the status of the machine (green: stopped, red: running, orange: pause). This cell also has several monitoring cameras (see **Figure 15**), which allow the process to be monitored from the control centre (external to the cell)

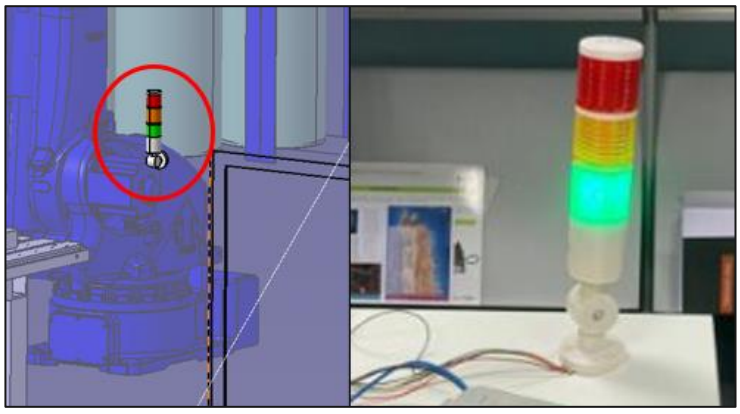


Figure 14 Security light towers.

Figure 15 Monitoring cameras for MOSES cell control.

On the other hand, and to protect the machine from possible collisions, the robot itself has a force measurement system, which in the event of a collision, the robot stops automatically, blocking the motors and stopping the spindle from turning.

All these systems are monitored by connexion boards (see **Figure 16**) based on a TCP/IP connection, which allows switching between the different colours of the light security tower. This system is constantly monitoring the inductive sensors arranged in each of the doors, acting with minimum response times.

Figure 16 Security cards for securities control.

All components included in this cell, with the exception of the worktable and the flange to hold the cutting head, have the corresponding CE marking, in compliance with current regulations. Furthermore, the design of the cell complies with the relevant national and European standards. The machine directive 2066/42/EC has been followed, which at regional level is expressed in the Royal Decree 1215. In addition, harmonized standards such as ISO 12100 and ISO 13849 have been considered, in terms of risk assessment and risk level scale respectively. On the other hand, ISO 13855 (positioning of protective devices according to the approach speed of parts of the human body), ISO 13857 (safety distances to prevent reaching dangerous areas with the upper and lower limbs) and ISO 14119 (interlocking devices associated with movable guards) have been considered as standards for the application of safety measures. **Figure 17** shows the required safety aspects used for MOSES cell design.

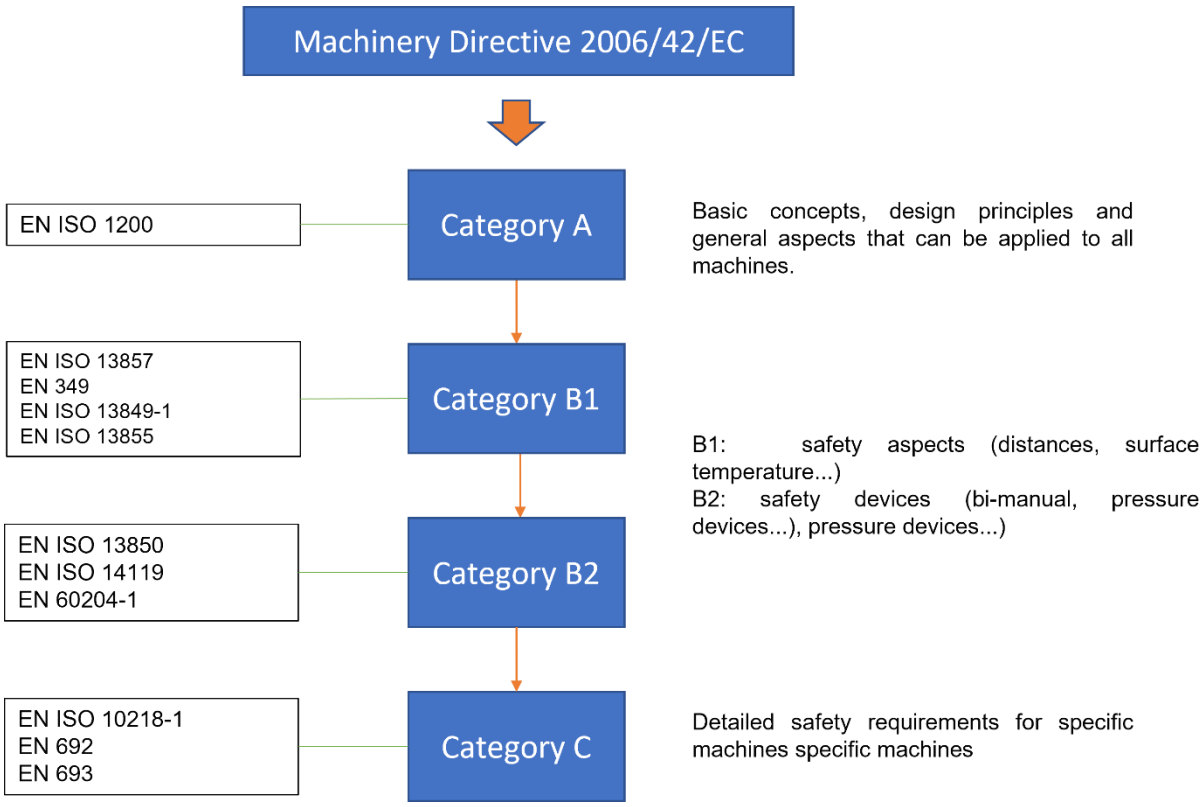


Figure 17 Safety aspects on MOSES cell.

CABKA

The main proposal for the CABKA use case is to automate the manual process, where an operator check the quality inspection on the pallets and remove the burrs with a cutter.
The inspection and deburring process are split in two stations due the dirty produced by the deburring. Then the robot 1 (Kuka KR 16 R1610-2) is the attendant of doing the inspection task and the robot 2 (Kuka KR 16 R1610-2) is the responsible of the deburring task. The third robot on the cell is a Kuka KR 50 R2500, is the pallets manager. The robot 3 do the pick and place in the line.

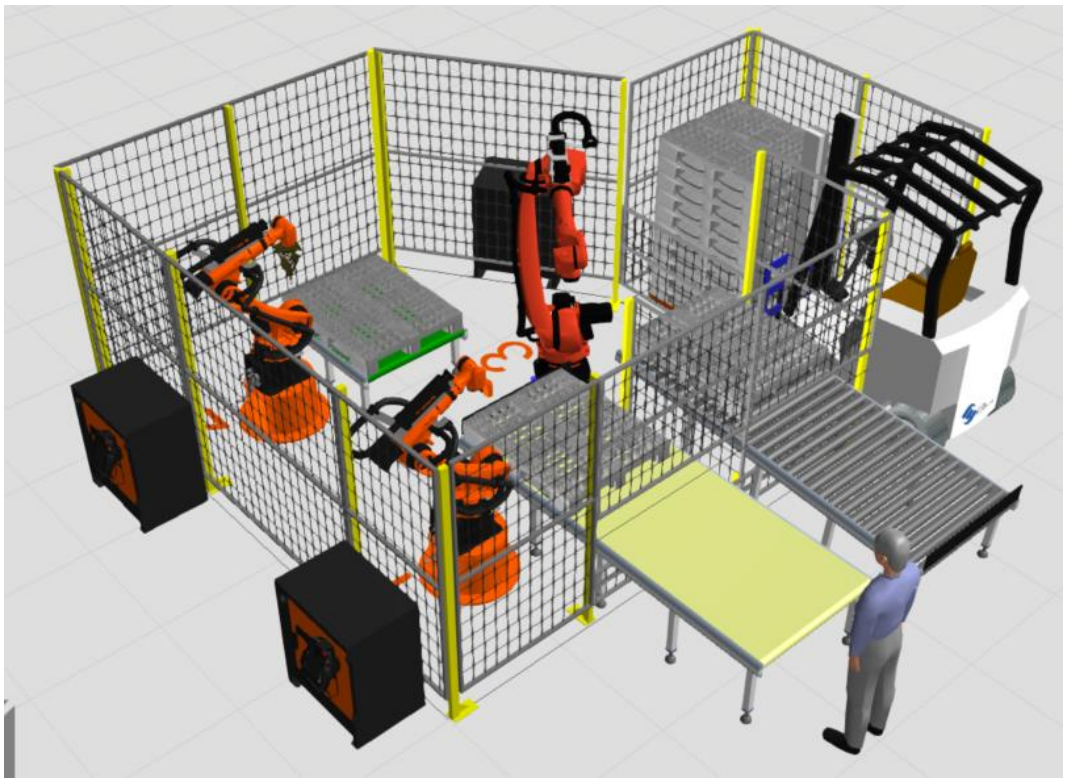
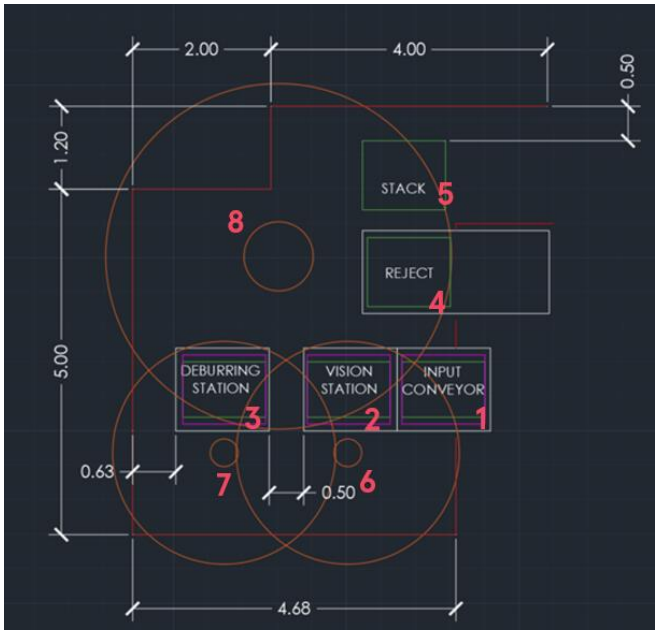


Figure 18 Cabka Cell Upgrade

Mechanical aspects

The line is composed of the following elements:



1. Input conveyor. This is a belt conveyor with a tunnel for the safety and laterals guides for the entry parts in the automation line.
2. Vision station. The same belt conveyor to finish the entry of the parts. This station has different clamps to fix the parts in an indexed position.
3. Deburring station. The station has fixations to index the part. The station is designed to collect the burrs outside the line.
4. The reject zone consists of inclined idler rollers to extract the bad parts out.
5. The stacker zone has a base structure to drop the part and pick them with the forklift.
6. This is the robot 1, a KUKA KR 16. Is installed in top a base, the hight of this base is designed to have the maximum reach area possible with the robot in relation the workstation. In addition, it is needing a support for the Zivid camera.
7. This is the Robot 2, another KUKA KR 16, with the base and gripper for the milling tool and RealSense camera.
8. This is the Robot 3, a KUKA KR 50. It has a base and gripper to pick and drop the parts on the line.

Electric distribution

The PLC and the electrical cabinet support all the electrical system in the line. The inputs and outputs are managed by the PLC. The ACROBA platform use the TCP/IP protocol to communicate with the PLC and the OPC UA communication for the data transmissions.

Pneumatic assembly

The line has a main pneumatic supply which is distributed to three different zones: vision, deburring, and robots. In the vision and deburring stations, pneumatics clamps are filled with air to fix the pallets on the table. The three robots are connected to the pneumatic for the lens cleaning system (Robot 1), pneumatic milling tool (Robot 2) and the end of arm tool for the Robot 3 (the gripper).

Safety aspects

The line has a door to access the line. In addition, a scanner area and a safety light curtain for the stack area are available. There are also different Emergency stop buttons around the line, as for example, in the box for the HMI, in the teach pendant from the robots, in the door access and electrical cabinet.

A 24 volts power supply, who enables the outputs and the modules, are managed by a safety relay.

STERIPACK

The Steripack 3D printing line layout is shown in the following figures (figures 17,18 and 19). Each element is clearly identified, and their individual functions detailed in a separate table.

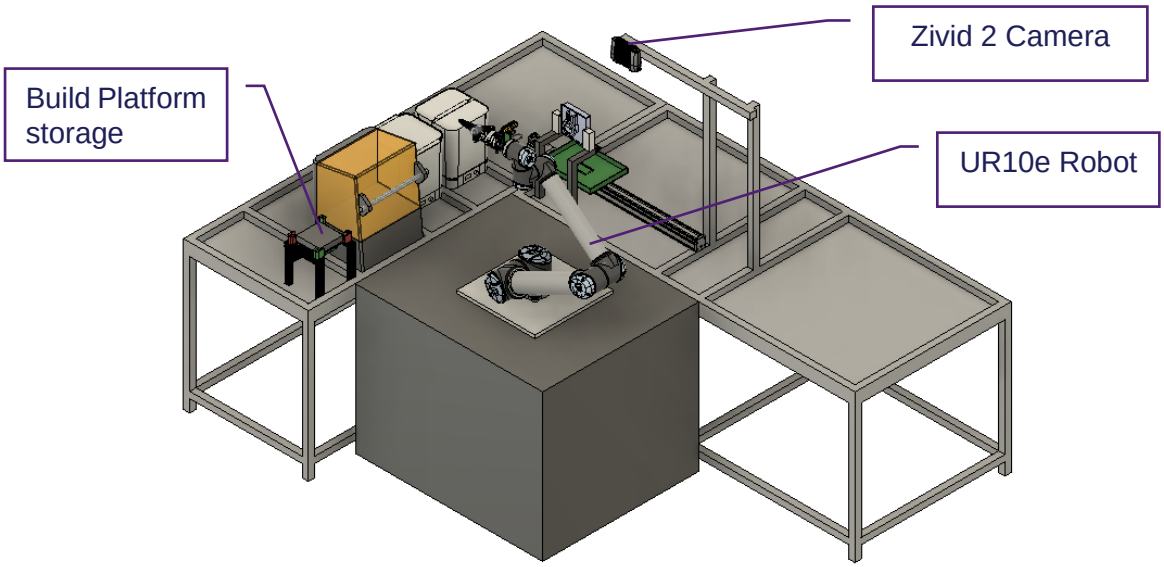


Figure 19 SteriPack Cell Upgrade – View 1

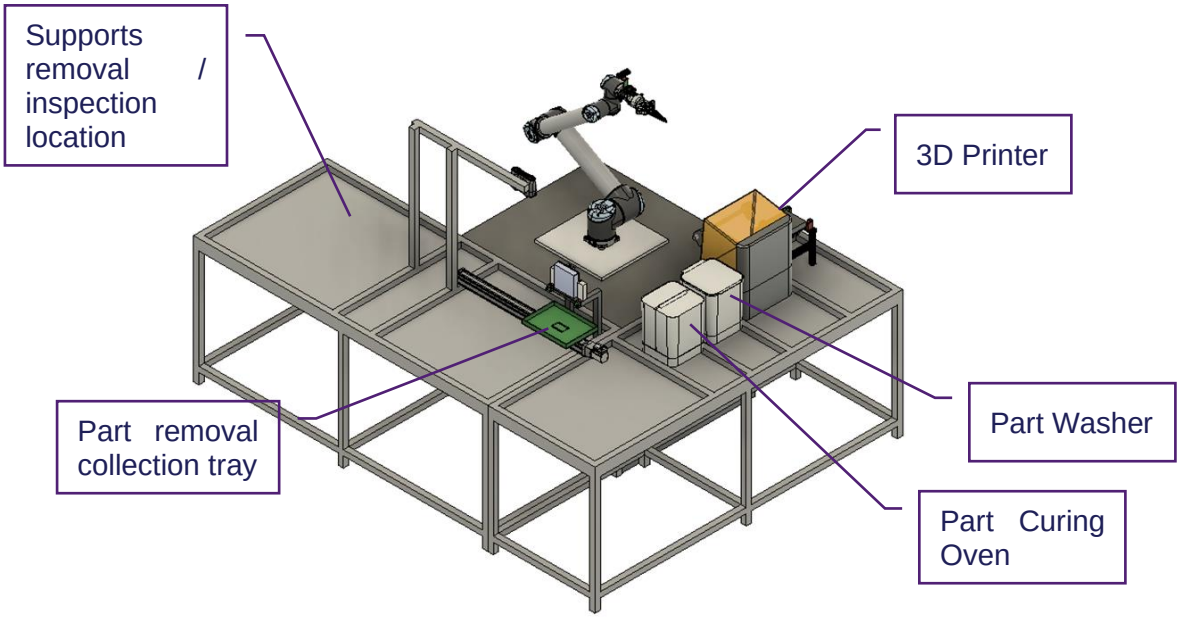


Figure 20 SteriPack Cell Upgrade – View 2

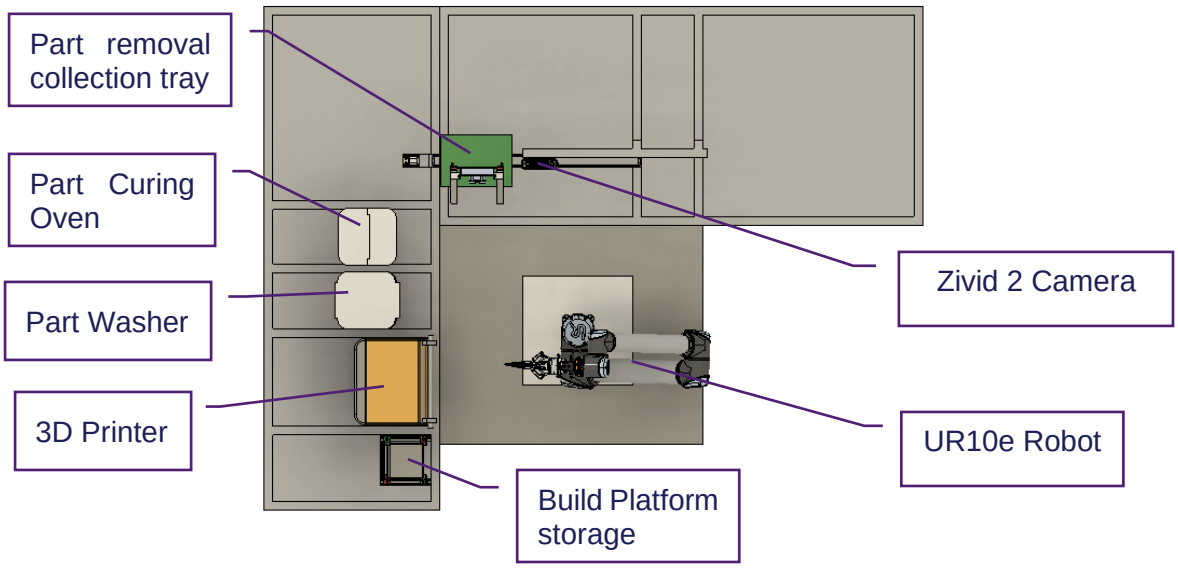


Figure 21 SteriPack Cell Upgrade – Top View

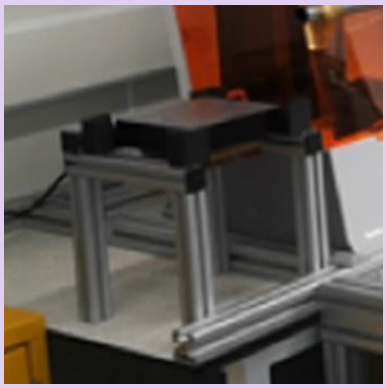
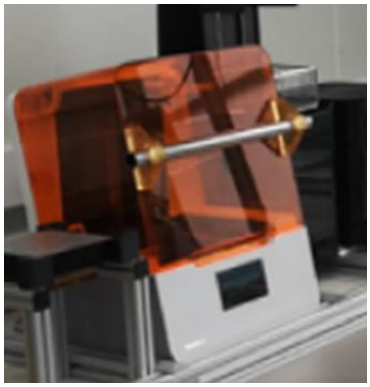
Figure 22 SteriPack Cell Upgrade – General View

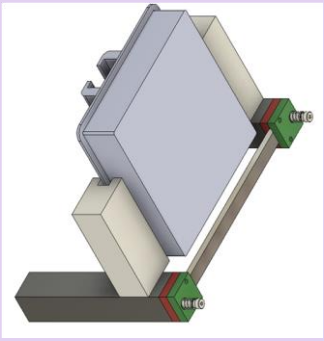
Mechanical aspects

As outlined in the previous section, the 3D printing cell is comprised of the following elements:

- Build platform holder / storage
- 3D printer
- Part washer
- Part UV curing oven
- Part removal fixture
- Support removal fixture

The following table lists each section / element of the line and their function:

Equipment Name	Function Description	Device
Build platform holder / storage	Fixture used to hold / store build platform ready to be picked up for queued print project	
3D Printer	Formlab 3B+ Printer used to process 3D printing projects	

Part UV curing oven	Formlab “Form Cure” UV oven to cure 3D printed parts	
Part Washer	Formlab “Form Wash” cleaning station to automatically clean 3D printed parts after the printing process is complete. This washer currently used IPA as the cleaning agent.	
Part removal fixture	Station used to hold the build plate with the printed parts and carry out the part removal step 	

Support removal fixture (In design)	Station used to carry out the support removal process and final inspection of the 3D printed parts	At concept and development phase.

In addition, a bin picking station is set up with a frame mounted camera, located at the part removal fixture and an Intel RealSense D435 camera is mounted on the robot, as shown in the following figure:

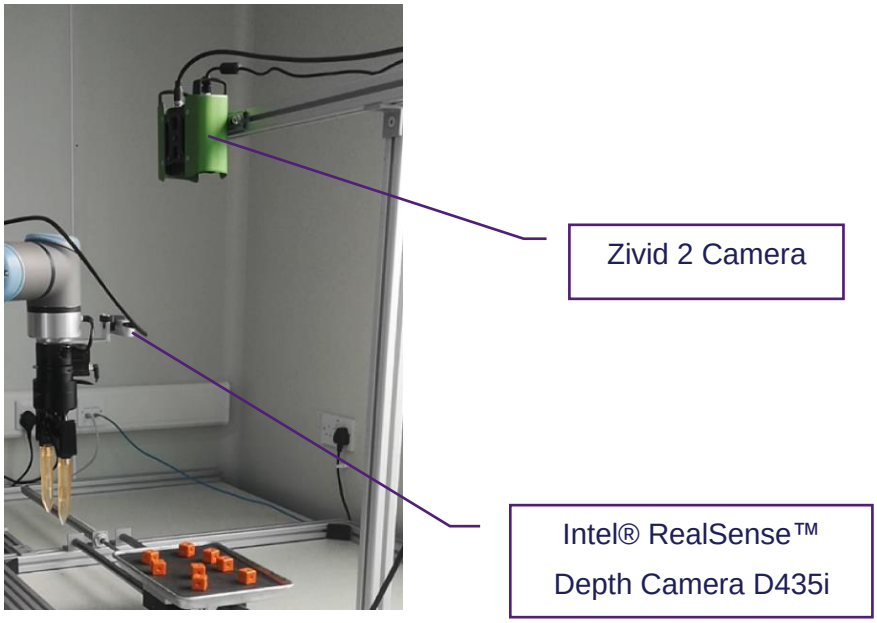


Figure 23 Bin picking station with Zivid 2 camera

The different elements of the cell are connected and / or communicate as per the high-level descriptive figure below, with the following table identifying each device:

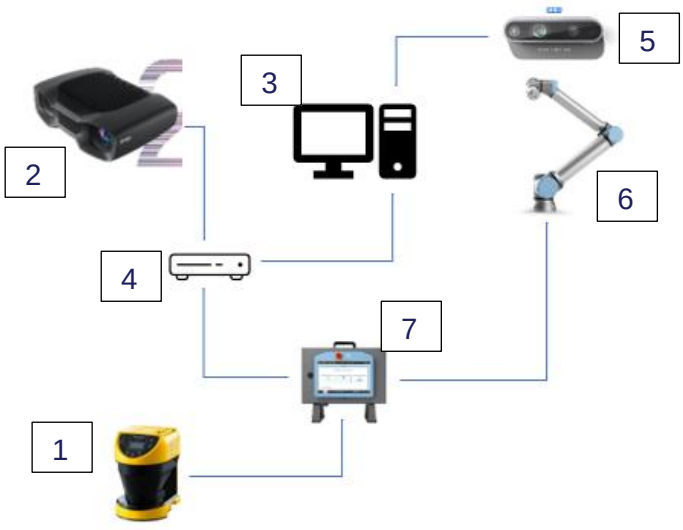


Figure 24 SteriPack Cell Upgrade – Connection / Communication architecture Overview

Device ID #	Equipment Name	Function Description
1	Keyence safety laser scanner, SZ-01S	Safety sensor with different zones set up to ensure safety of the cell during processing
2	Zivid 2 Camera	Camera for the bin picking operations
3	PC with Ubuntu Operating System	Used to run the ACROBA platform

4	Ethernet switch	Used to enable and manage communications between each device
5	Intel® RealSense™ Depth Camera D435i	Planned for inspection and location
6	UR10e	Robot configured to run the 3D printing process
7	UR10e Robot controller	Robot controller

Electric distribution

All the equipment used in the SteriPack cell setup is by a single-phase 240VAC voltage. More details for each device power requirements are outlined in the next table:

Equipment Name	Function Description	Power / Electrical Requirements
3D Printer	Formlab 3B+ Printer used to process 3D printing projects	100–240VAC, 2.5A, 50/60Hz, 220W
Part UV curing oven	Formlab “Form Cure” UV oven to cure 3D printed parts	90–240V, 6.0A, 50/60Hz, 144W
Part Washer	Formlab “Form Wash” cleaning station to automatically clean 3D printed parts after the printing process is complete. This	90–240V, 2.0A, 50/60Hz, 50W

	washer currently used IPA as the cleaning agent.	
Keyence safety laser scanner, SZ-01S	Safety sensor with different zones set up to ensure safety of the cell during processing	Power through I/O board of UR10e Robot controller, 24 VDC ±10 %, Ripple (P-P) 10 % or less
Zivid 2 Camera	Camera for the bin picking operations	240VAC, with 24 V 5A power adapter
PC with Ubuntu Operating System	Used to run the ACROBA platform	240VAC
Ethernet switch	Used to enable and manage communications between each device	240VAC
Intel® RealSense™ Depth Camera D435i	Planned for inspection and location	Power through Ubuntu PC via compatible USB 2.0 / USB 3.1 port
UR10e Robot + controller	Robot configured to run the 3D printing process	100-240 VAC, 47-440 Hz

Pneumatic assembly

Currently, there are no requirements for compressed air for the cell. However, if the need arises due to a design change for some fixtures, tooling or otherwise, a compressed air supply is readily available to provide clean, dry, air up to 6bar at the cell location.

Safety aspects

Safety features are implemented throughout the cell and include the following:

- E-Stops (1 located on robot pendant and 1 located at edge of cell)
    - These E-stops will interrupt all actions immediately on being activated.
- The UR10e collaborative safety features.
- A Keyence safety laser scanner (SZ-01S)
    - The scanner is set to detect safety
        - Safe zone: Out of reach and harm from robot movement – No action, robot moves at set speeds and cell runs in optimal conditions.
        - Intermediate zone: Robot will continue to operate but at a reduced speed.
        - Unsafe zone: Robot stops immediately.

Figure 25 SteriPack Cell Upgrade – Keyence safety laser scanner, SZ-01S

All components and devices withing the cell carry a CE mark and each individual element meet the following standards:

Equipment Name	Function Description	Safey
3D Printer	Formlab 3B+ Printer used to process 3D printing projects	CE mark

Part UV curing oven	Formlab “Form Cure” UV oven to cure 3D printed parts	CE Mark
Part Washer	Formlab “Form Wash” cleaning station to automatically clean 3D printed parts after the printing process is complete. This washer currently used IPA as the cleaning agent.	CE Mark
Keyence safety laser scanner, SZ-01S	Safety sensor with different zones set up to ensure safety of the cell during processing	IEC61496-1, EN61496-1, UL 61496-1 (Type 3 ESPE), IEC61496-3, EN61496-3 (Type 3 AOPDDR), IEC61508, EN61508, IEC62061, EN62061 (SIL2), EN ISO13849-1:2015 (PL d, Category 3), UL508, UL1998
UR10e Robot + controller	Robot configured to run the 3D printing process	EN ISO 13849-1, PLd category 3, EN ISO 10218-1

Integration of Devices

ROBOTS

In today's rapidly evolving field of robotics, researchers often face the task of selecting the most suitable robotic systems for their projects. One intriguing approach is to utilize multiple brands of robots in a research endeavour. This allows for the exploration of diverse features, capabilities, and interoperability among different brands.

When considering the benefits of using multiple robot brands, several key points emerge. Firstly, the range of features and capabilities offered by each brand provides researchers with a rich set of options. Yaskawa, KUKA, and ABB, for example, offer distinct models with varying payload capacities, reaches, speeds, and precision. By incorporating robots from different brands, researchers gain access to a wider spectrum of capabilities, tailoring their choices to meet the specific requirements of their research tasks.

Flexibility and availability are additional factors to consider. The utilization of multiple robot brands ensures researchers have the freedom to select the most suitable robot from each brand for a particular task or experiment. The market availability of different brands also facilitates the acquisition process, as researchers can tap into a broader pool of resources to obtain the necessary robots for their projects.

Furthermore, the interoperability and compatibility among robot brands deserve attention. While each brand may possess its own controller and software, ACROBA offers integration and communication options that transcend brand boundaries. This cross-brand compatibility empowers researchers to seamlessly integrate robots from different brands into a cohesive system, enabling collaborative research and investigating advanced techniques for coordination and collaboration among diverse robotic entities.

Yaskawa Motoman GP50

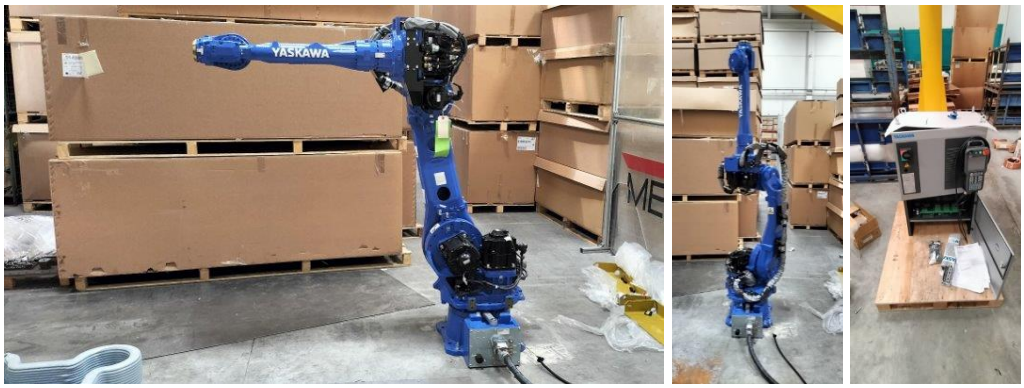


Figure 26 Reception of the Motoman GP50

The creation of the pilot cell located in the AITIIP facilities for the testing of the skills of the plastic use cases has started. In this way, a Yaskawa Motoman GP50 industrial robot and a series of peripherals, such as cameras and probes, have been set up and their connections have been configured with a ROS environment in ACROBA's workspace.

This robot has been fixed to the floor and has been manually set up to ensure its correct operation. It has also been installed the necessary Jobs for communication with ROS through the file INIT_ROS.JBI and prepared for automatic execution.

Installing MotoPlus Code

Installation using Binaries

This method of installing MotoROS is recommended for all users, unless you intend to develop MotoROS itself (ie: are looking to change the behaviour of MotoROS, not just use it).

Downloading a binary

The binary files are linked below. Different binaries are required for different robot controller models.

Locate the correct binary for your robot controller and download it to a location on your PC (fi the `Downloads` folder).

Verifying integrity of the download

Check the integrity of the downloaded binary to avoid loading corrupted binaries onto the controller.

The example command below uses `md5sum` on Linux, but any utility which can calculate MD5 hashes could be used, on any OS.

To calculate the MD5 hash on Debian/Ubuntu for the FS100 binary linked above, run the following commands (on other OS, use the appropriate commands or tools instead):

```
cd /path/to/where/the/binary/was/saved
md5sum -b MotoRosYRC1_1910.out
```

For `v1.9.10` of MotoROS for YRC1000 controllers, this command will return `A66C43CCA899BD3127297DC1EAC42D99`.

If using this command for other versions of MotoROS and/or with binaries for other controller models, replace the filename with the correct one.

Compare the output of `md5sum` when run against the binary downloaded in the previous section (*Downloading a binary*) with the values listed in the following table. The values must match *exactly*.

Controller	File	Version	MD5 hash
YRC1000	MotoRosYRC1_1910.out	v1.9.10	A66C43CCA899BD3127297DC1EAC42D99

Loading the Binary File

In all cases (ie: whether using pre-built binaries or a version compiled from source), the `.out` file should be placed on an external memory card; Compact Flash (CF) or **USB**. Now insert the memory card into the robot's programming pendant and refer to the following sections.

YRC1000

Turn on the robot controller while holding the `{Main Menu}` key on the keypad to enter **Maintenance Mode**. You may release the key when you see the logo appear on the screen.

Now in Maintenance Mode:

1. upgrade to MANAGEMENT security level by touching `[System Info]→[Security]` (default password is all 9's)
2. touch `[System Info]→[Setup]` and select `OPTION FUNCTION`
3. move to `MotoPlus FUNC.`, set this to `USED`
4. move cursor down to `MOTOMAN DRIVER` and set to `USED`
5. touch `[MotoPlus APL]→[Device]` to select either CF or USB memory type
6. touch `[MotoPlus APL]→[Load (User App)]` to select and load `MotoRosXXXXX.out`
7. touch `[MotoPlus APL]→[File List]` and verify that MotoROS was properly installed and no other MotoPlus applications are currently loaded on the controller
8. rotate the pendant key-switch (upper left of pendant) fully counter-clockwise into `TEACH` mode
9. reboot the robot controller into regular mode

Within 30 seconds of fully booting up, you should get alarm `8001[10] Speed FB enabled, reboot now`. Reboot again and there should be no alarms.

Installing INFORM Code

The motoman driver specifies a required handshake between the native INFORM robot code and the MotoROS layer. The MotoROS code will automatically run an INFORM job called `INIT_ROS` at the start of a motion command, which will provide the required handshaking.

Load the correct `INIT_ROS.JBI` file from the appropriate directory in motoman/motoman_driver/Inform onto the robot controller. Be sure to load the file for the correct configuration of your controller (ie: single arm, dual arm, with base axis, etc). Copy the `IONAME.DAT` file found in the same directory as well.

Place this job file on an external memory card; Compact Flash (CF), SD, or USB. Insert the memory card into the robot's programming pendant. Turn on the robot controller and boot into normal operation mode.

Using the Smart Pendant directly (without PC Software Pendant):

1. Touch `[MENU]→[Utility]→[File Transfer]`.
2. Touch `[To Controller]`.
3. If the file is not in the USB root, then touch `[Change Folder]` and navigate to the appropriate folder.

4. In the Job list, check the box in front of the INIT_ROS.JBI job and then touch the [COPY FILES TO CONTROLLER].
5. Answer [YES] to the File Transfer to Controller – Confirmation message.

Controller Configuration

Verify that the controller is enabled to receive remote commands in Remote mode:

Using standard pendant:

1. Rotate the pendant key-switch (upper left of pendant) fully counter-clockwise into TEACH mode.
2. Upgrade your Security Level to MANAGEMENT
    a. Touching [System Info]→[Security].
    b. Default password is all 9's.
3. Using the pendant, select [In/OUT]→[PSEUDO INPUT SIG]
4. For DX100, DX200, and FS100, set input #82015 (CMD REMOTE SEL) to ON. The signal number is #87015 on YRC.
    a. Move the cursor to the circle beside the specified input
    b. Press [INTERLOCK]+[SELECT] to turn on the input
5. Rotate the pendant key-switch (upper left of pendant) fully clockwise into REMOTE mode.
6. Using Smart Pendant:
7. Rotate the pendant key-switch (upper left of pendant) fully clockwise into REMOTE mode.

KUKA

Figure 4 KUKA KR 16 R1610-2 installation

The robot has been fixed to the base and the base into the floor. The robot has been manually set up to ensure its correct operation. Is installed the software package KUKA.RobotSensorInterface 3.3 to connect with ROS. This node can manipulate a KUKA robot arm via RSI 3.

This node advertises services for moving a KUKA robot arm into a desired position—specifying the X, Y, Z, A, B, C values for the robot's end effector—and for obtaining its current position.

ABB 1200 and UR10e Robots

Although it was originally planned to use a UR5, STERIPACK have a long-term goal to use a single robot provider, in this case ABB. The preference for ABB 1200 robot was due to three main reasons:

- Staff familiarity with ABB: STERIPACK have been recently expanding the robotics team. The incoming staff have had extensive experience with ABB robots in industrial settings.
- Local supplier availability: An approved supplier of ABB robots and experienced integrator is located less than 15 minutes from the production facility. Thus, STERIPACK believe that by using an ABB robot the chances of ACROBA been developed, implemented more widely and certified is increased.
- Industrial robot: STERIPACK aims to fulfil the lights-out manufacturing paradigm i.e., with no human operators in the loop. As such collaborative robots are in principle necessary. An industrial robot will enable a successful automated solution to be run at high speeds with a longer lifespan.

Based on the above, the ABB robot was an attractive option. However, following further testing and evaluation, it was decided to switch to a UR10e robot as it offered the collaborative option as opposed to the ABB robot, which did not.

The added benefits to switching to the UR10e also included the following:

- Better reach.
- Increased payload.
- Collaborative operation.
- Easier ROS integration.
- Simpler EOAT (End Of Arm Tool) integration.

The overall characteristics of the ABB and UR10e robots can be compared in the following table:

Title	ABB 1200 Details	UR10e Details
Type	ABB IRB 1200-7/0.7	UR10e
Payload	7kg	12.5kg
Reach	700mm	1300mm

Repeatability	0.02	0.02
Number of axes	6	6
Controller dimensions	320 x 449x 422mm	460 x 449 x 254mm
Controller weight	25.5kg	12kg
I/Os	External module	External module

Figure 27: ABB IRB 1200 and UR10e specifications overview

Comparing both robots, the UR10e has great benefits in terms of payload, reach and ease of integration compared to the ABB 1200. As a result, the UR10e was selected to be integrated as part of the 3D printing cell.

CAMERAS

Alternatives Discussion

Two vision cameras have been analysed for fitting the requirement of ACROBA project.

Specifications

Camera model	acA5472-17uc Basler ace	acA5472-5gm Basler ace
Sensor Size	13.1 mm x 8.8 mm	
Resolution (HxV)	5472 px x 3648 px	
Resolution	20 MP	

Pixel Size (H x V)	2.4 µm x 2.4 µm	
Frame Rate	17 fps	5 fps
Mono/Color	Color	Mono
Resolution (H x V Pixels)	5496 x 3672 (full resolution)	
	5472 x 3648 (default resolution)	
Resolution	20 MP	
Sensor Type	Sony IMX183CQJ-J	Sony IMX183CLK-J
	Progressive scan CMOS	
	Rolling shutter	
Sensor Format	1"	
Effective Sensor Diagonal	15.86 mm	
Pixel Size (H x V)	2.40 x 2.40 µm	
Image Data Interface	USB 3.0, nominal max. 5 Gbit/s	Fast Ethernet (100 Mbit/s)
		Gigabit Ethernet (1000 Mbit/s)

Pixel Formats	BGR 8	Mono 12
	Bayer RG 12	Mono 12 Packed
	Bayer RG 12 Packed	Mono 8
	Bayer RG 8	
	Mono 8	
	RGB 8	
	YCbCr422_8 (YUV422_8)	

The acA5472-17uc Basler ace and acA5472-5gm Basler ace cameras exhibit significant differences in various aspects. While both have a sensor size of 13.1 mm x 8.8 mm and a resolution of 20 MP, the former offers a frame rate of 17 fps, whereas the latter is limited to 5 fps. Additionally, the acA5472-17uc is a colour camera, while the acA5472-5gm is monochrome. They also differ in the type of sensor used, with the former utilizing the Sony IMX183CQJ-J and the latter employing the Sony IMX183CLK-J. Another distinctive factor is the data interface: the acA5472-17uc features USB 3.0 with a nominal maximum speed of 5 Gbit/s, while the acA5472-5gm offers faster options such as Fast Ethernet and Gigabit Ethernet. Lastly, the cameras have variations in the supported pixel formats, providing different choices for image capture.

In the case of the acA5472-17uc we don’t consider viable to get a full image transfer throw a USB interface. Therefore, in order to work with the maximum amount of information, it is considered preferable to work with the acA5472-5gm.

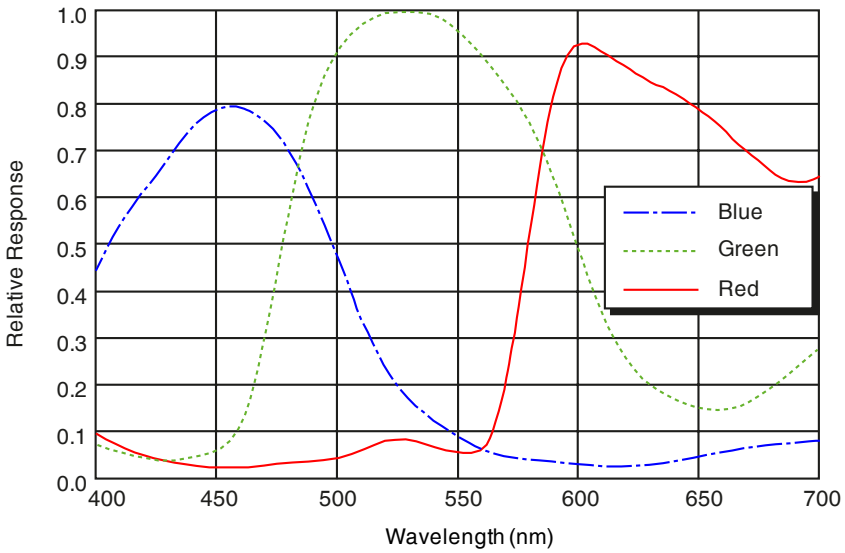


Figure 28. acA5472-17uc Basler ace Spectral response

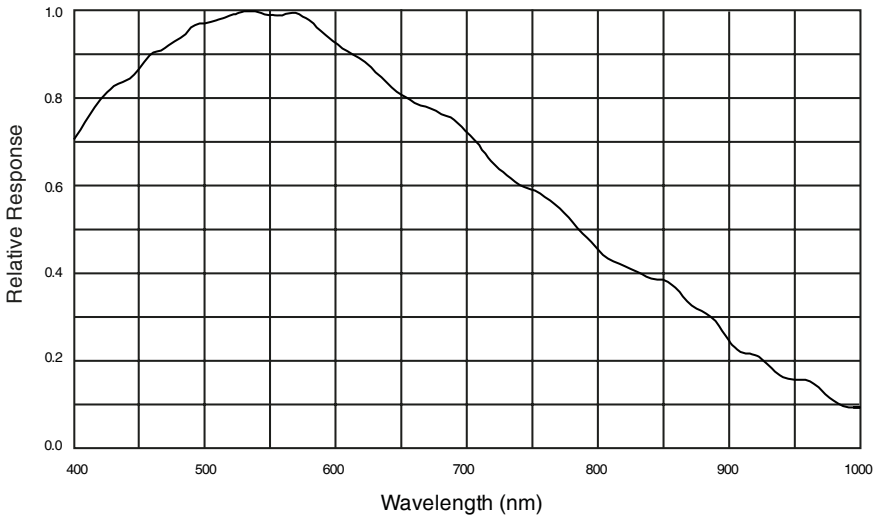


Figure 29. acA5472-5gm Basler ace Spectral response

In addition, a monochrome sensor, compared to an RGB sensor, has several advantages in terms of the wavelength it can capture. These advantages stem from how monochrome and RGB sensors capture and process color information.

1. Higher sensitivity: Monochrome sensors are more light-sensitive than RGB sensors. Without color filters, monochrome sensors capture light across the entire wavelength range, allowing them to gather more information and details in low-light conditions. This makes them ideal for applications such as night vision or in environments with limited lighting.
2. Higher effective resolution: Since monochrome sensors lack color filters, each pixel captures full luminance information. This means that the effective resolution of a monochrome sensor is higher than that of an RGB sensor, where each pixel only captures one color component (red, green, or blue). In applications that require fine details and high resolution, such as scientific photography or microscopy, monochrome sensors offer significant advantages.
3. Larger dynamic range: Monochrome sensors often have a wider dynamic range than RGB sensors. Dynamic range refers to a sensor's ability to capture details in both high and low-light areas of a scene. Without color filters that reduce the amount of light reaching each pixel, monochrome sensors can record a greater amount of detail in shadow areas and reveal more information in bright areas.

However, it is important to note that RGB sensors also have their advantages in applications where color is crucial, such as landscape or portrait photography. Additionally, the ability to capture color information enables specific image analysis and processing, such as color-based object recognition. Therefore, the choice between a monochrome sensor and an RGB sensor will depend on the specific needs and requirements of each application.

Supported ROS Image encoding

The correct format compatibility has been also checked. In this way, the list of supported ROS message sensor_msgs/Image encodings are as shown in the following *Abstract*[1]:

```
ABSTRACT_ENCODING_PREFIXES []
BAYER_BGGR16 ="bayer_bggr16"
BAYER_BGGR8 ="bayer_bggr8"
BAYER_GBRG16 ="bayer_gbrg16"
BAYER_GBRG8 ="bayer_gbrg8"
BAYER_GRBG16 ="bayer_grbg16"
BAYER_GRBG8 ="bayer_grbg8"
BAYER_RGGB16 ="bayer_rggb16"
BAYER_RGGB8 ="bayer_rggb8"
BGR16 = "bgr16"
BGR8 = "bgr8"
BGRA16 = "bgra16"
BGRA8 = "bgra8"
MONO16 ="mono16"
MONO8 ="mono8"
RGB16 = "rgb16"
RGB8 = "rgb8"
RGBA16 = "rgba16"
RGBA8 = "rgba8"
```

ArUco ROS image detection

For the implementation of the detection code, the aruco_ros library is used. In the Aruco Ros 3.0.3, line 562, the library always converts color images into grayscale, so it does not take advantage of the full color information:

[1] Source: sensor_msgs::image_encodings Namespace Reference. Available at: https://docs.ros.org/en/api/sensor_msgs/html/namespacesensor__msgs_1_1image__encodings.html
Also available at: http://docs.ros.org/en/noetic/api/sensor_msgs/html/image__encodings_8h_source.html

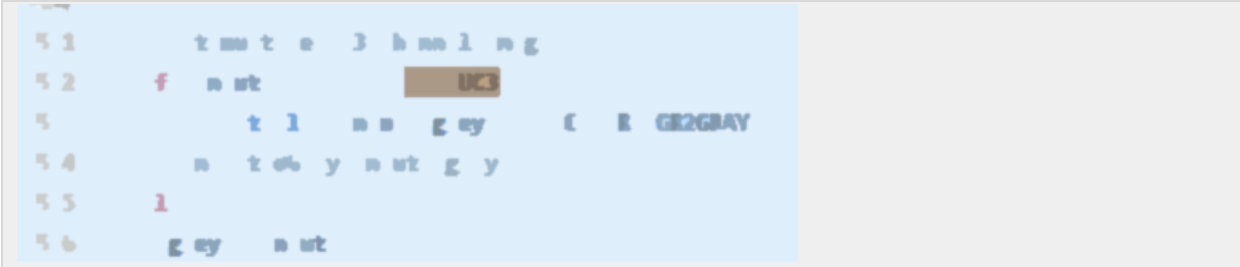


Figure 30. aruco_ros/markerdetector.cpp at noetic-devel · pal-robotics/aruco_ros (github.com).

Remarks

The acA5472-5gm camera, in mono8 format, representing grayscale images with 8 bits per pixel, offers a more efficient use of bandwidth compared to other formats like RGB. This optimization results in faster data transmission, reduced storage requirements, and more efficient use of available bandwidth.

Mono8 cameras, capturing images using a monochrome sensor without color filters, offer improved spectral response curves. These cameras can capture a broader range of wavelengths and exhibit enhanced sensitivity to different spectral components. This characteristic makes Mono8 cameras particularly suitable for applications that require accurate and detailed spectral analysis. In scientific research, spectroscopy, or industrial quality control, where precise spectral data is crucial, Mono8 cameras excel. The absence of color filters also eliminates any potential loss of light transmission associated with these filters, ensuring a more accurate and reliable measurement of spectral data.

In summary, selecting a Mono8 camera provides benefits in terms of bandwidth utilization and spectral response. The optimized use of bandwidth allows for faster data transmission and reduced storage requirements. Additionally, the improved spectral response curves enable more accurate and detailed spectral analysis, making Mono8 cameras ideal for applications where efficient data transfer and precise spectral measurement are essential.

20 MPX BASLER CAMERAS

Mechanical Installation

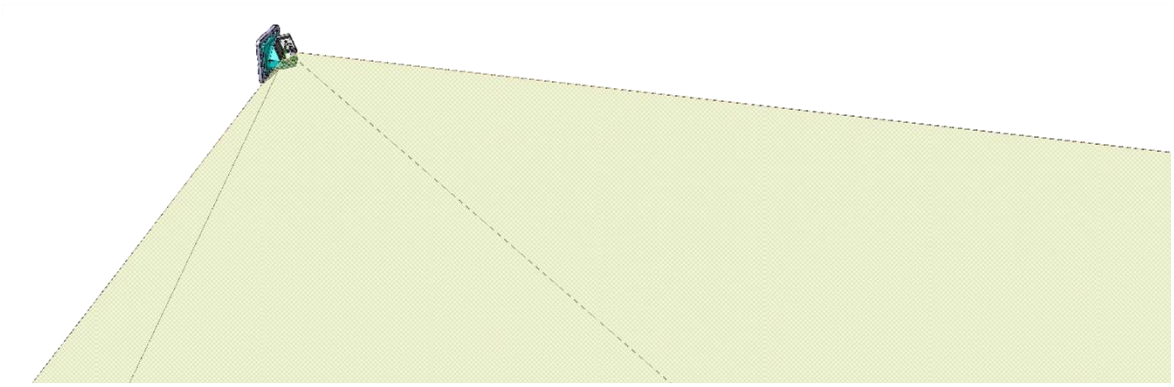


Figure 31 Field of view for the camera

Among the aforementioned sensors, we find the vision cameras that capture the environment. These cameras are Basler ace acA5472-5gm of 20Mpx in black and white, with Basler Lens C11-1620-12M-P f16mm lenses that give us a horizontal field of view of 90º. This field of view allows us, positioning the cameras in the 4 lateral columns, to cover 100% of the cell. Over the cameras, 150W spotlights have been installed to eliminate shadows due to perspective and make the identification of QRs and AruCos more accurate.

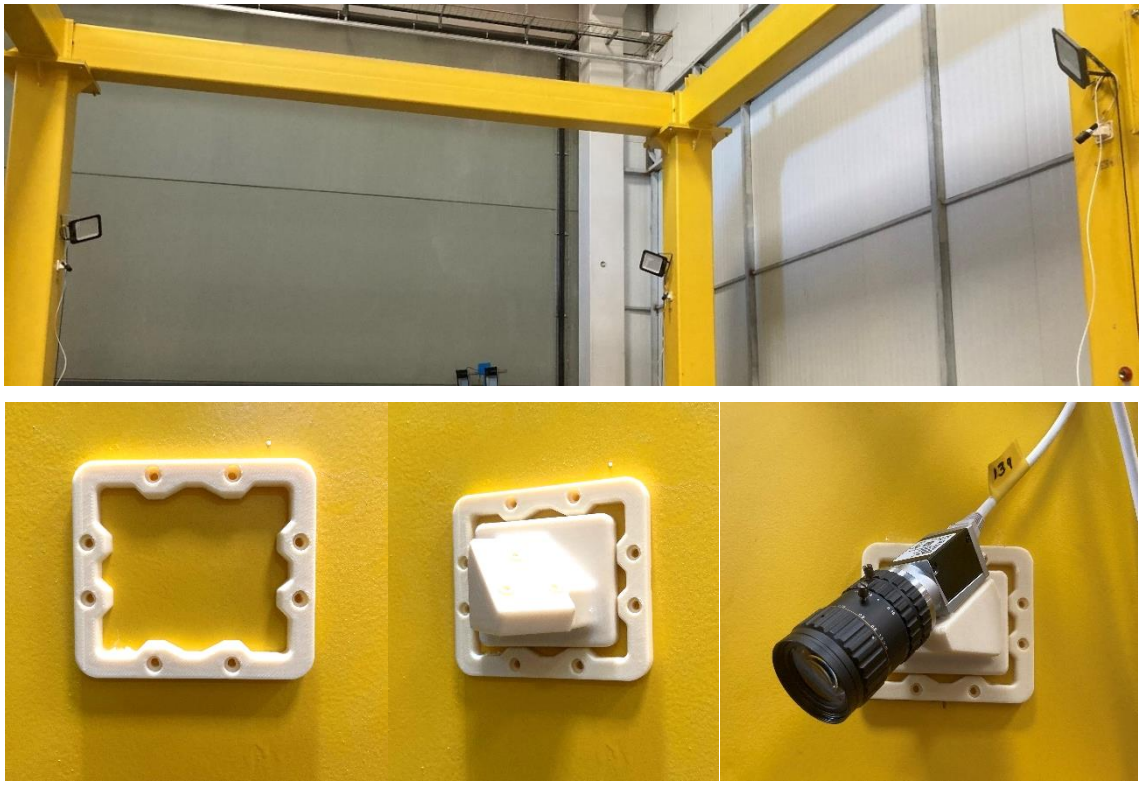


Figure 32 Camera installation

These cameras have been installed in such a way as to ensure their position in case they are disassembled, since the quality of the surveys is closely linked to the accuracy of the extrinsic matrix (characteristics of the camera with respect to its environment). For this purpose, a guiding element has been installed which will remain fixed on the column, even if the camera is disassembled. Together with this guiding element will be the camera support itself, which will also ensure the angle to coincide with its theoretical positions and orientations.

Figure 33 Lighting

These cameras have been calibrated by means of a chest board (intrinsic matrices) to define the deformation and aberration behavior of the lenses. This board has been printed in the largest size that would not have deformations and could be transported. In this way a 1000 mm x 1250 mm board was printed on feather board and moved along the field of view.

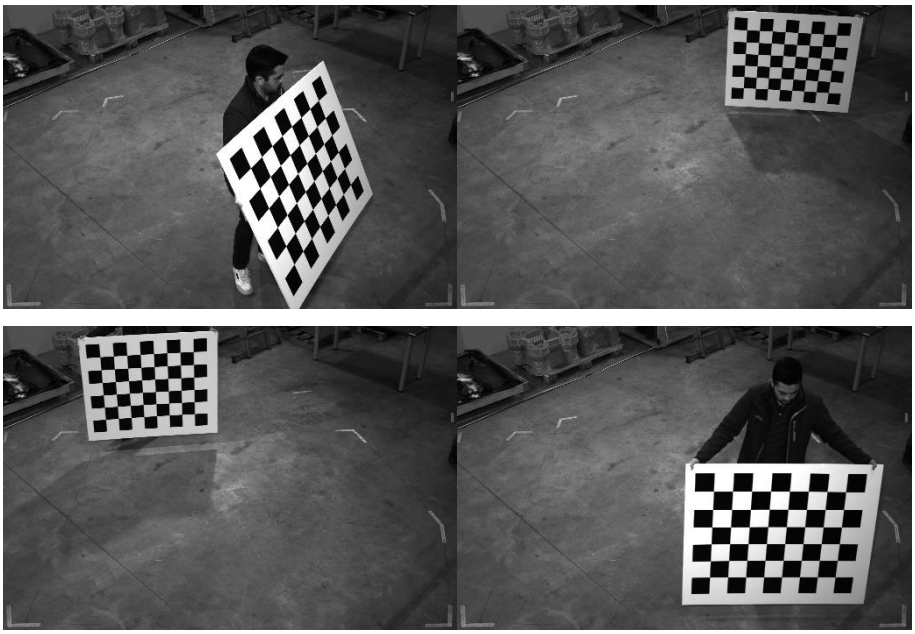


Figure 34 Camera Intrinsic matrix calibration

This calibration makes use of the Brown-Conrady distortion model to correct for radial aberrations according to the following formula:

$$r = \sqrt{x^2 + y^2}$$

$$dr = k_1 r^3 + k_2 r^5 + \ k_3 r^7 + \cdots + k_n r^{n+2}$$

As well as tangential aberration as the solution to the following problem:

$$dt = P(r)\cos\left(y\ -\ y_0\right)$$

where ψ is our off-center distortion profile function.

Thus, the resulting coefficients are as follows:

$[k_1, k_2, p_1, p_2, k_3]_1 = [-0.154890635142534,\ 0.332718275854098,\ 0.00866435113120582,\ -0.00333851631004805, 0]$

$[k_1, k_2, p_1, p_2, k_3]_2 = [-0.145238010730331,\ 0.152052050349106,\ 0.00993961816617193,\ -0.00035077414796319, 0]$

$[k_1, k_2, p_1, p_2, k_3]_3 = [-0.144903969161072\ , 0.190023335120791,\ 0.00595240460325730,\ -0.00192167296933974, 0]$

And the intrinsic matrix of the camera according to the following matrix,

$$\begin{bmatrix} u \\ v \\ e \end{bmatrix}_1 = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

it would result in:

$$\begin{bmatrix} u \\ v \\ e \end{bmatrix}_1 = \begin{bmatrix} 7215.050055563702 & 0 & 2430.253591810172 \\ 0 & 7263.067748612555 & 1890.866476353492 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ e \end{bmatrix}_1 = \begin{bmatrix} 7215.050055563702 & 0 & 2430.253591810172 \\ 0 & 7263.067748612555 & 1890.866476353492 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ e \end{bmatrix}_1 = \begin{bmatrix} 7327.571742210247 & 0 & 2626.687938432756 \\ 0 & 7273.029714813312 & 2529.170395602811 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ e \end{bmatrix}_1 = \begin{bmatrix} 6921.100312902593 & 0 & 2597.037315122553 \\ 0 & 6913.989104516228 & 2080.443637280554 \\ 0 & 0 & 1 \end{bmatrix}$$

The extrinsic matrices have been found by inverting a common point in the scene at three different positions to reduce the measurement noise. A combination of ArUcos and Chestboard called ChArUco has been used for this measurement.

Figure 35 Camera Extrinsic Matrix Calibration

The matrices, according to the following formulation would be:

$$\begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_1 = \begin{bmatrix} 0.7083823 & -0.437928 & 0.553547 & -2.963761 \\ -0.705358 & -0.410576 & 0.577838 & -2.877297 \\ -0.025778 & -0.799779 & -0.599741 & 3.223020 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_2 = \begin{bmatrix} 0.704611 & 0.477706 & 0.524709 & 2.788571 \\ 0.709589 & 0.476983 & 0.518623 & -2.909515 \\ -0.002528 & -0.737755 & -0.675063 & 3.335804 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_2 = \begin{bmatrix} -0.707010 & -0.437718 & 0.555464 & -2.808907 \\ -0.707191 & 0.442269 & -0.551615 & 2.514186 \\ -0.004212 & -0.782816 & -0.622238 & 3.129106 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

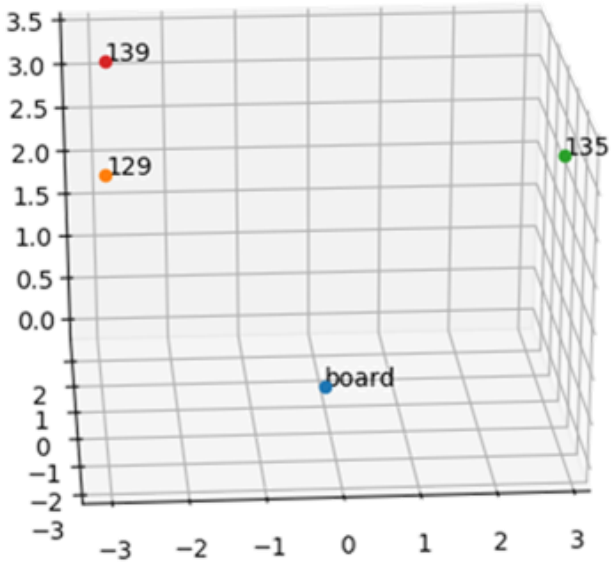


Figure 36 Cell Camera distribution

RGB-D Zivid CAMERA

As part of the bin picking operation as well as the inspection, the SteriPack cell uses the following 2 cameras:

- Zivid Two M70
- Intel® RealSense™ Depth Camera D435i

The RealSense camera is mounted on the UR10e robot arm and the Zivid camera is mounted on a frame, in a fixed location, to be used for the bin picking operations of the process.
The installation and execution of the relevant ROS drivers was carried out as follows:

- For the Intel® RealSense™ Depth Camera D435i:
  - The relevant drivers can be found at the following location:
    - https://github.com/IntelRealSense/realsense-ros
    - - Open a new window or terminal
    - - Source the terminal: **source devel/setup.bash**
    - - Launch the realsense driver by executing**: roslaunch realsense2_camera rs_camera.launch align_depth:=true enable_pointcloud:=true publish_tf:=false**

- For the Zivid Two M70:
  - The relevant drivers can be found at the following location:
    - https://github.com/zivid/zivid-ros
    - -Open a new window or terminal
    - Source the terminal: **source devel/setup.bash**
    - Launch the Zivid driver by executing: **ROS_NAMESPACE=zivid_camera rosrun zivid_camera zivid_camera_node**
  - To capture data from the Zivid Two camera:
    - Open a new window or terminal
    - Source the terminal: **source devel/setup.bash**
    - Launch Zivid by executing: **rosrun zivid_samples sample_capture_cpp**

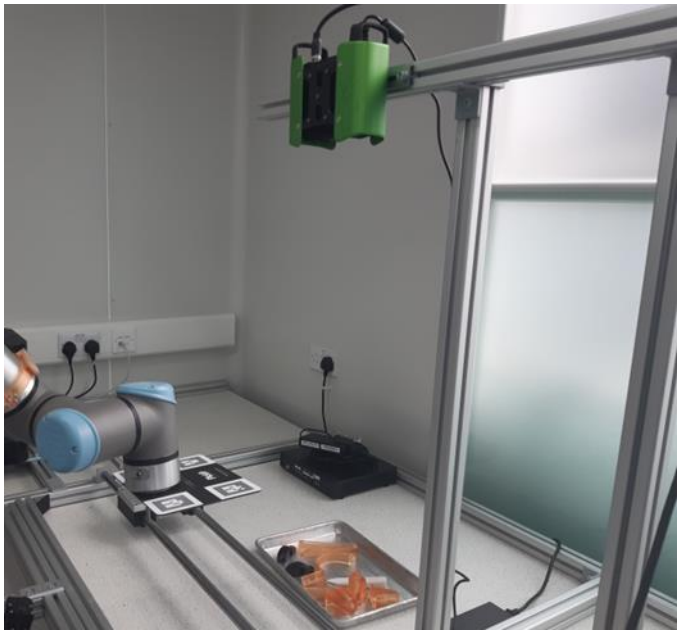


Figure 37: Zivid camera location

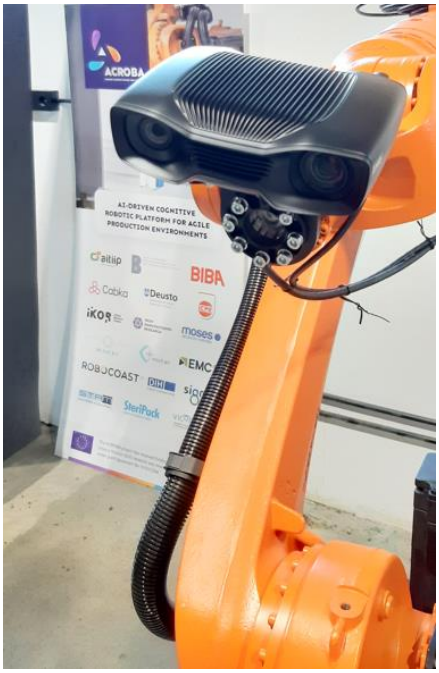


Figure 10 Zivid Two Camera onboard the robot.

In the CABKA use case, the Zivid Two M70 camera is mounted onboard the robot 1 for the inspection process and the Intel RealSense Depth Camera D435i onboard of the robot 2 for the deburring process.

SENSOR INTEGRATION

TCP Board

In order to work with a TCP board is necessary to set up a communication board that will manage the connection to the board IP and port and also to manage the information in the channel.

This node will as for the status of each of the I/O signals in the board and it will publish the information in an exchange topic.

This exchange topic will be used also for reading the command for the relay actuation and send it to the board.

In this way, the TCP Board topic will be always synchronized with its corresponding topic.

Distance

Among other things, a laser distance sensor has been integrated into the head. This sensor is connected to the robot via a SLIO analogue input module, which allows the robot's position changes to be reflected in real time.

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

This code is integrated with a raster function that allows to perform rasters of specific areas in regions of interest and return a point cloud that will be published in a log file.

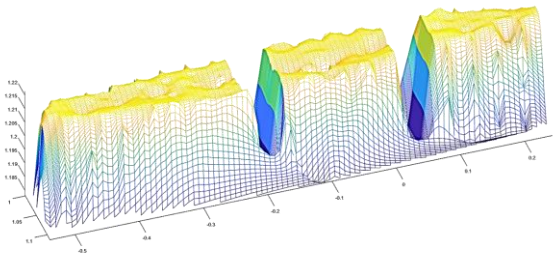


Figure 38 Distance Raster representation.

Torque sensor

The SteriPack cell currently does not have the need for a torque sensor anymore as this was originally due to the use of the ABB robot. The cell is using the force feedback from UR10e robot, which was another reason for choosing this robot as there was no need for an additional sensor in this case.

ROS INTEGRATION

All of the above systems have been integrated into the Acroba ecosystem and have been programmed to work together in the ROS environment. Through calls to action servers, publishers, subscribers, and services.
To do all the systems integration with the robot, Ros Industrial, Motoros, Moveit and Descartes are used to manage the movement of the robot. In this way, with Ros Industrial and Motoros it is possible to make the connection between ROS and the robot by means of the following command:

```
1 [redacted]
```

Moveit, on the other hand, performs point-to-point movement management. And Descartes manages the movement of concatenated trajectories.

Describing more in detail de integration process:

Motoman Driver Installation

The packages in the main repository have been released into ROS Indigo on Ubuntu, making installation through `apt-get` or synaptic possible. **Other Linux distributions will have to build from source**.
To install all released Indigo packages, run the following on the command line:

```
1 [redacted]
```

This installs all the dependencies as well. The controller must be installed/configured to work with ROS-Industrial as well.
As in our case the system works based on ROS Noetic it will be necessary to build the package from source.

Be sure than the firewall is not blocking external connections.

Activating the workspace

Finally, activate the workspace to get access to the packages just built:

At this point all packages should be usable (i.e., roslaunch should be able to auto-complete package names starting with motoman...). In case the workspace contains additional packages (i.e., not from this repository), those should also still be available.

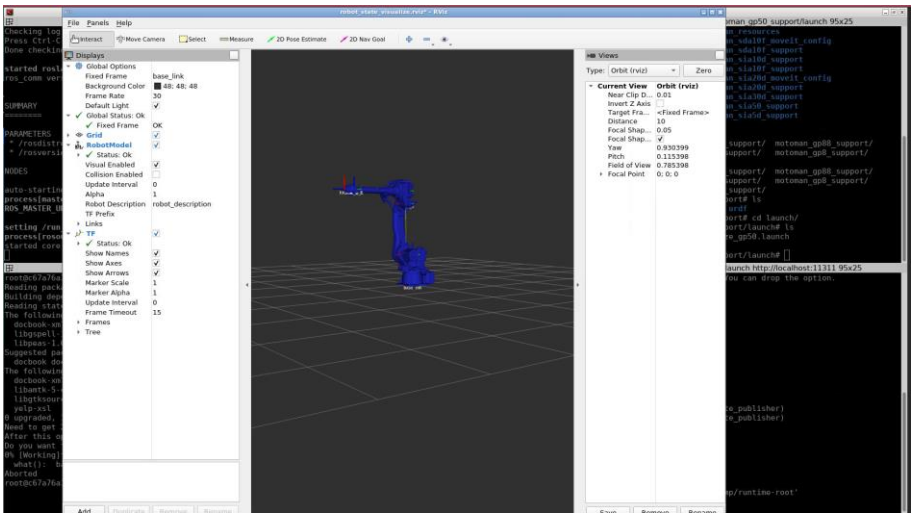
Test motoROS instalation

test_gp.launch

The `test_gp50.launch` file allows users to inspect and verify the urdf in RViz. It automatically loads the appropriate urdf (by including `load_gp50.launch`) and starts

the `joint_state_publisher` and `robot_state_publisher` nodes to allow users to interact with the model. This launch file does not need access to any real hardware.

```
1 [redacted]
```



robot_state_visualize_m10ia.launch

The `robot_state_visualize_gp50.launch` allows users to visualise the current state of a real (or simulated) robot in RViz. It does this by starting the relevant nodes from the appropriate ROS-Industrial robot driver package, providing them with any parameters they might need to correctly interface with this specific model (in this case a Fanuc M-10iA), and by starting RViz and the `robot_state_publisher` node.

This launch file obviously requires access to real (or simulated) hardware, and has (at least) one required argument: the ip address of the industrial controller. The controller must already be running the appropriate ROS-Industrial server programs.

```
1 [redacted]
```

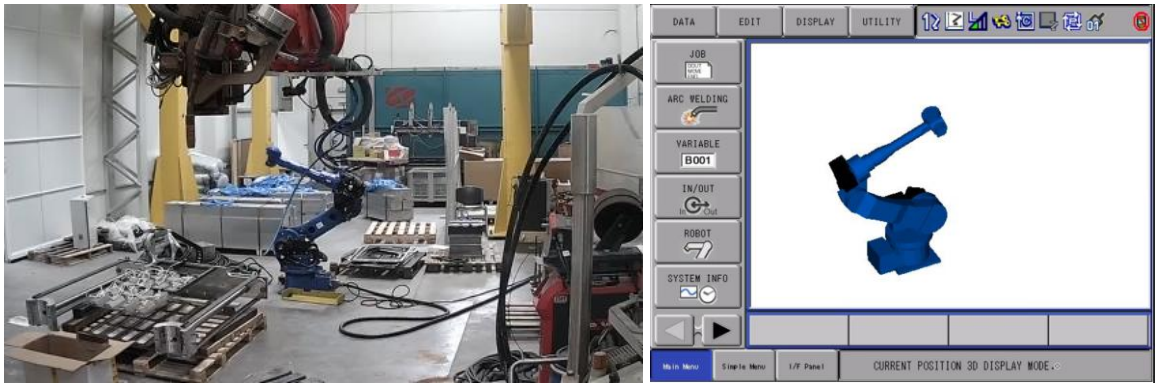


Figure 39 Representation of the robot in the Yaskawa Virtual Environment vs the real scenario

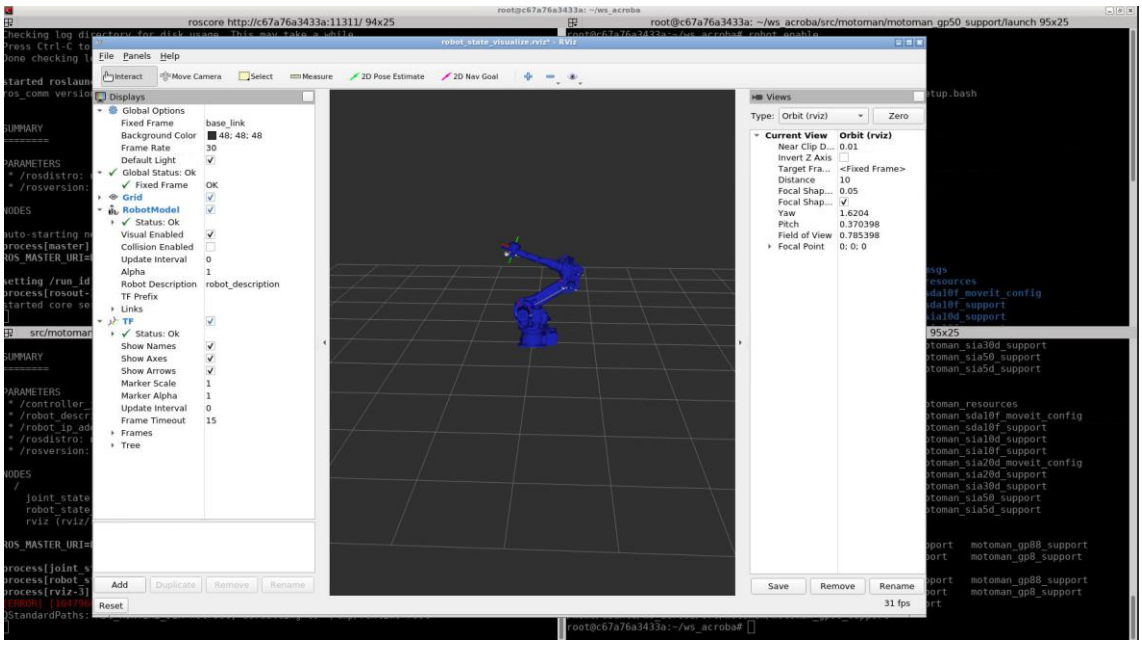


Figure 40 RViz Virtual Environment showing the current pose

robot_interface_streaming_m10ia.launch

The last file, `robot_interface_streaming_gp50.launch`, starts all ROS nodes required for setting up a full bi-directional connection between ROS and the industrial controller.

Both state reporting and trajectory relay nodes are started (see Industrial Robot Client - Design). In most cases, this launch file is a wrapper around a similar file provided by the

ROS-Industrial driver package for the controller, but with any additional information the driver might need to correctly interface with a specific model (again, in this case the M-10iA).
This launch file requires access to real (or simulated) hardware, and has (at least) one required argument: the ip address of the industrial controller. The controller must already be running the appropriate ROS-Industrial server programs.

```
1
```

It is possible to launch like this as well:

```
1
```

Asking for the joint states topic:

```
1
```

Figure 41 Report of robot position using the topic /joint_states

Kuka Driver Installation

The Kuka driver installation start with the joint configurations with the file: “joint_names_kr16.yaml”

```
1
```

The first step is needed to load the program in the ROS. Then we create the file: “load_kr16_2.launch”

```
1
2
3
4
5
```

The second step is testing the launch in ROS with the file:
“roslaunch_test.xml”

```
1
2
3
4
5
6
7
8
9
```

To test the complete function, is call the following file: “test_kr16_2.launch”

```
1
2
3
4
5
6
7
8
```

UR10e Driver Instalation

To carry out the installation and integration of the UR10e robot and the Robotiq 2F-85 gripper, the following pre-requisites are needed:

- PC configured for ROS, with Ubuntu 20.04 LTS installed.
    - The ROS version installed is ROS Noetic.
- URCAPS installed:
    - External Control URCAP.
    - RS485 URCAP.
- A live connection to the UR10e robot.
    - The Robotiq 2F-85 gripper is connected to the UR10e robot via the tool I/O port. The Ur10e robot is in turn connected via Ethernet to a LAN, communicating with the Ubuntu PC.

The following steps were followed to install, setup and control the gripper:

- URCap Installation:
    - Download the URCap from Github
    - Install the URCap on the UR robot (e-Series, cb-Series)
- To connect and control the gripper via ROS the rs485 URCap needs to be installed on the Robot:
    - Download the URCap from Github
    - Install the URCap on the UR robot (e-Series, cb-Series)
- The UR10e Robot installation settings are as follows:
    - On Robot teach pendant, go to Installation menu, General tab, and select Tool I/O
        - Set the tool output voltage to 24V
        - Enable “communication Interface” and use defaults
- Network setting are configured with the following parameters:
    - The Ubuntu PC and the UR10e robot controller are connected to a network switch. To enable communication, both devices must be on the same network.
- Ubuntu PC installation:
    - Create the ROS workspace and install the following packages from github
        - Install the official UR ROS driver in the ROS workspace by using git clone command.
        - Install the robotiq driver being maintained by Clearpath by using the git clone command.
        - Make sure you are on the master branch for the UR ROS driver and on the branch Noetic for the robotiq ROS driver.
    - In the UR ROS package of the workspace, open the robot launch file (e.g., Universal_Robots_ROS_Driver/ur_robot_driver/launch/ur5e_bringup.launch)
        - Define the UR's static IP address
        - Set “use_tool_communication” to true

        - Change the “tool_device_name” to /tmp/ttyTool
        - Change the "tool_voltage" to 24 (volts)
    - In the robotiq ROS package of the workspace, open the robotiq launch file of the gripper, for example:
        - robotiq/robotiq_2f_gripper_control/launch/robotiq_action_server.launch)
        - Set respawn=true
        - Set the comport to /tmp/ttyTool
- Start-up order:
    - Power on the UR10e robot using the teach pendant
    - On the Ubuntu PC, open a terminal at the ROS workspace root
    - Source the workspace (source devel/setup.bash)
    - Launch the UR robot driver (eg.roslaunch ur_robot_driver ur5e_bringup.launch)
    - Open a new terminal and source the ROS workspace
    - Launch the robotiq gripper driver, the file you customized (eg. roslaunch robotiq_2f_gripper_control robotiq_action_server.launch)
    - Wait for the robotiq driver to start. The light on the gripper should turn from solid red to solid blue and the gripper should close and re-open (initialise)
    - Start the external_control program on the UR with the teach pendant
    - To control the gripper, open a sourced terminal of the workspace and call the ROS actionlib “command_robotiq_action”. Change the position (m), speed (m/s) and force (N) values to the required values, for example:
        - rostopic pub /command_robotiq_action/goal robotiq_gripper_msgs/CommandRobotiqGripperActionGoal “header: seq: 0 stamp: secs: 0 nsecs: 0 frame_id: ‘’ goal_id: stamp: sec: 0 nsec: 0 id: ‘’ goal: {emergency_release: false, emergency_release_dir: 0, stop: false, position: 0.085, speed: 0.05, force: 5.0}”

To start the required drivers to run the cell, the following steps are followed:

- Open a Terminal (Ctrl + Alt + t)
- **cd acroba_ws**
- **source devel/setup.bash**
- **roslaunch ur_robot_driver ur10e_bringup.launch robot_ip:=172.28.60.3**
- Open a new window or terminal
- Source the terminal:
    - **source devel/setup.bash**
- Start the driver for the robotiq gripper:
    - **roslaunch robotiq_2f_gripper_control robotiq_action_server.launch**
- On the Ur10e Robot, open Acroba External control.urp and press play, (play program)
- Open a new window or terminal
- Source the terminal:
    - **source devel/setup.bash**

- Launch realsense driver:
  - **roslaunch realsense2_camera rs_camera.launch align_depth:=true enable_pointcloud:=true publish_tf:=false**
- Open a new window or terminal
- Source the terminal:
  - **source devel/setup.bash**
- Launch moveIT:
  - **roslaunch ur10e_gripper_camera_moveit_config steripack_cell.launch**
- Open a new window or terminal
- Source the terminal:
  - **source devel/setup.bash**
- Launch Zivid driver:
  - **ROS_NAMESPACE=zivid_camera rosrun zivid_camera zivid_camera_node**
- Open a new window or terminal
- Source the terminal:
  - **source devel/setup.bash**
- Launch Zivid take snapshot:
  - **rosrun zivid_samples sample_capture_cpp**

MoveIt! Installation

Installation & Configuration

1. Install MoveIt!

```
$ sudo apt install ros-noetic-moveit
```

2. Launch the setup assistant:

```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```

3. Create a **new** package and select the path to your robot's URDF file
   - it is okay to reference URDF files in previously-created `robot_description` or `_arm_navigation` packages
   - you can directly import XACRO files in addition to URDF files
4. Calculate Self-Collision matrix:
   - Set Sampling Density fairly high (~80,000?). For most robots, this doesn't take long. ⇨ 100,000
   - press "Regenerate Default Collision Matrix" button
   - review (select) the various collision-pairs to see if they make sense
5. Add required Virtual Joint:
   - MoveIt requires at least one virtual joint, connecting the robot to the world
   - Add Virtual Joint:
     - `name = 'FixedBase'` (arbitrary)

- `child = 'base_link'` (should match the URDF root link)
- `parent = 'world'` (arbitrary, until robot is placed in an environment)
- `type = 'fixed'

6. Add Planning Groups:
   - Add manipulator (arm) group
     - Add Group, `name = 'acroba_robot'`, `kinematic solver = 'KDLKinematicsPlugin'`
     - Add Kin. Chain
       - this method is generally preferred over Add Joints or Add Links
       - assign `base_link` and `tip_link` (link_6_t)

   - Add end-effector group **[OPTIONAL]**
     If you define an End-Effector, MoveIt can use this in grasp-planning and visualization methods.
     If your robot doesn't have an end-effector, just leave this blank.
     - Add Group, `name = 'gripper'`, `kinematic solver = 'none'`
     - Add Links
       - assign end-effector links
7. Add Robot Poses **[OPTIONAL]**:
   - Add one or more default poses, to reference in later planning code
     - (all-zeros, home, etc.)
   - The `MoveIt!` button will cycle the virtual robot through all defined poses
8. Add an End Effector **[OPTIONAL]**:
   - `name = 'gripper'` (arbitrary)
   - `group = 'gripper'` (should match group created above)
   - `parent = 'flange'` (last arm link)
   - `parent group = 'manipulator'`
9. Configure Controller:
   - Auto Add `FollowJointsTrajectory` Controllers For Each Planning Group
10. Generate Configuration Files:
    - specify a path to create the new moveit package for your robot
      - recommended: `.../motoman_gp50_moveit_config`

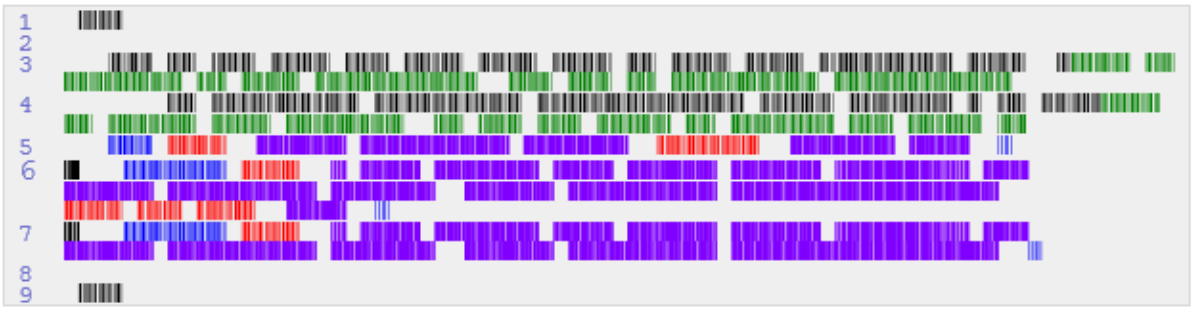
    - press "Generate Package" button
      - check for the "Generated Successfully" message!

        You may see a warning that an End-Effector has not been defined. It is okay to continue, if this is the desired configuration. MoveIt will work just fine with robots that have no end-effector (just a bare tool flange).
    - if the robot's URDF file is located in a rosbuild package, the Setup Assistant will place an absolute file path to the URDF in the `.setup_assistant` file. This is not portable, but can be fixed with a manual edit to that file:

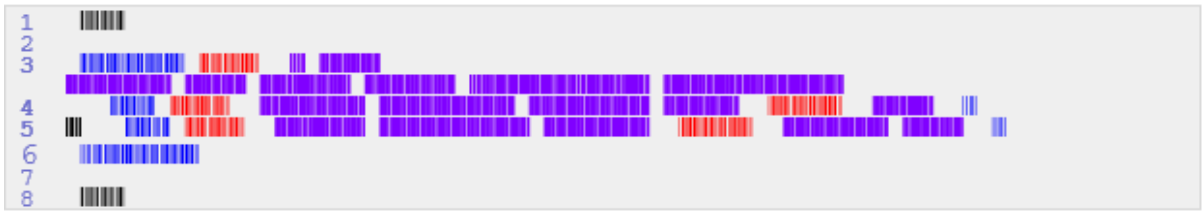
Fixing generated code

Over the generated code by `setup_assistant.launch` is needed to modify a little the following files:

`trajectory_execution.launch.xml`



`moveit_planning_execution.launch`



Pylon Driver Installation

In order to work with GigE cameras is needed the following driver. Here is listed the steps for installing the driver and make the cameras work in the ROS environment of ACROBA. the

- Clone this repository in your catkin workspace (e.g., catkin_ws):

- Clone drag&bot public common messages:

- Install ROS dependencies:

```
1 [illegible]
```

- Compile the workspace using catkin build or catkin make:

```
1 [illegible]
```

- Start the driver:

```
1 [illegible]
```

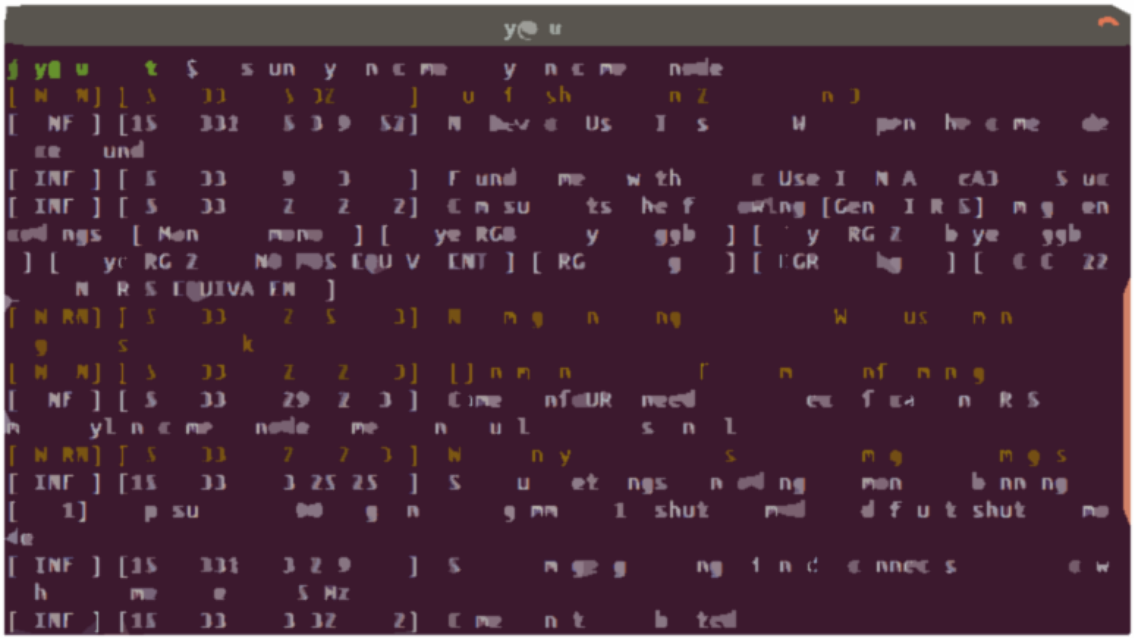


Figure 42. Interfacing Basler Cameras with ROS.

- GigE Cameras IP Configuration can be done using the command:

```
roslaunch pylon_camera pylon_camera_ip_configuration.launch
```

After doing that, it is possible to access to the camera configuration through the Pylon Viewer

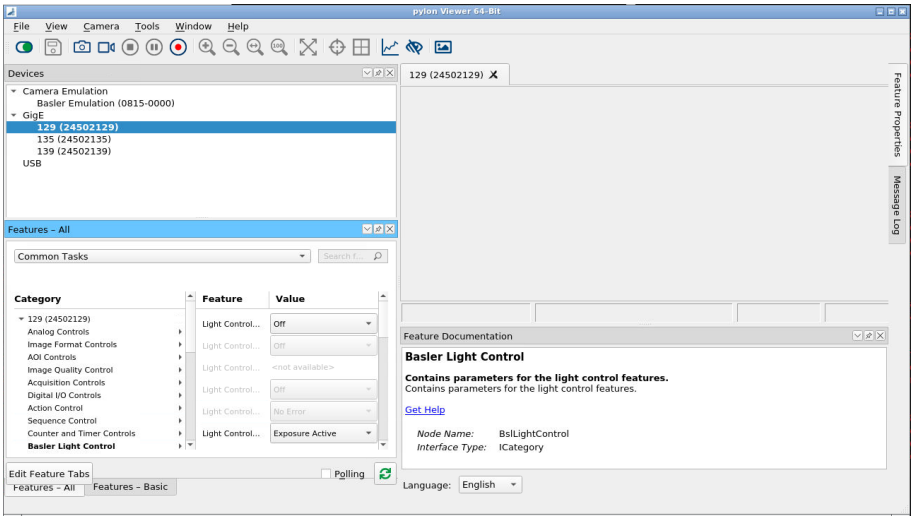


Figure 43 Capturing configuration tool provided by Basler

Or you can just to execute the camera driver with this command:

For being able to use that command, a yaml file with the configuration need to be defined and written. In this file, the main parameters for the camera will be set up.

- Image Width in pixels
- Image Height in pixels
- Camera Name
- Camera Matrix with the Camera projection matrix used for computing field-of-view
- Distorsion Model, Distortion Coefficients and Rectification Matrix describe how is the distortion of the lens when you move away from the centre of the sensor and how needs to be modify the image to fit better reality
- Projection Matrix with the camera position and orientation respect a reference frame
- Camera Frame with the name of the current frame
- Device User ID with the ID of the camera
- Image Encoding with the pixel type: rgb8 or mono8 by default
- Binning X and Binning Y for downsampled the image
- Frame Rate with the desired publisher frame rate
- Trigger Timeout in milliseconds
- Grab Timeout in milliseconds

Grab Strategy with the buffer strategy

White balance Strategy

Mode of Camera's Shutter

Exposure

Gain

Gamma

Brightness (exposure auto and gain auto)

MTU Size to prevent lost frames for GigE cameras

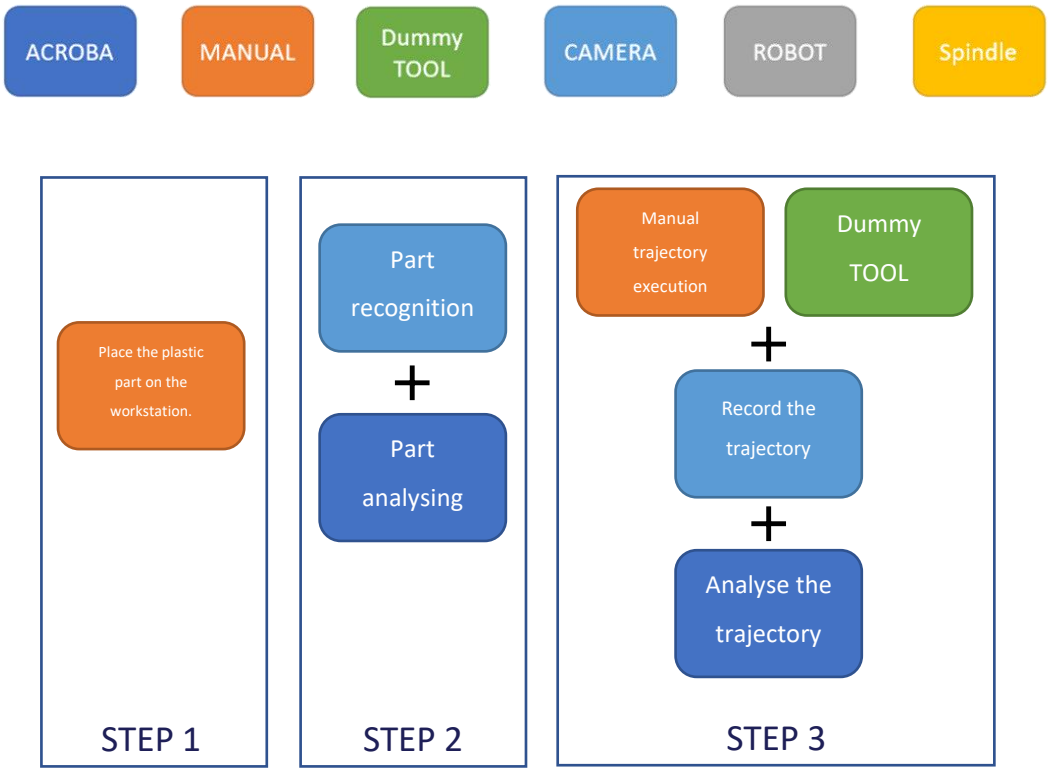
Figure 44 Fragment of camera configuration yaml file

This file needs to be filled by calibrating the camera (process introduced in section 20 MPX BASLER CAMERAS).

Processes and Flow Charts

MOSES

The main activity of MOSES is the production of parts by injection moulding and rotational moulding, both processes are intended for the production of thermoplastic parts. One of the main markets is the production and personalisation of container lids for the recovery of each type of waste. Up to now, what is done is to produce lids without personalisation (only the size and colour are personalised) and then they are introduced into the robotic cell to make the necessary cuts depending on the type of waste to be introduced into the container. This is where the ACROBA system is being developed to provide greater flexibility to the process. The flow chart shows the real operation that the cutting process will follow after the implementation of the ACROBA system.



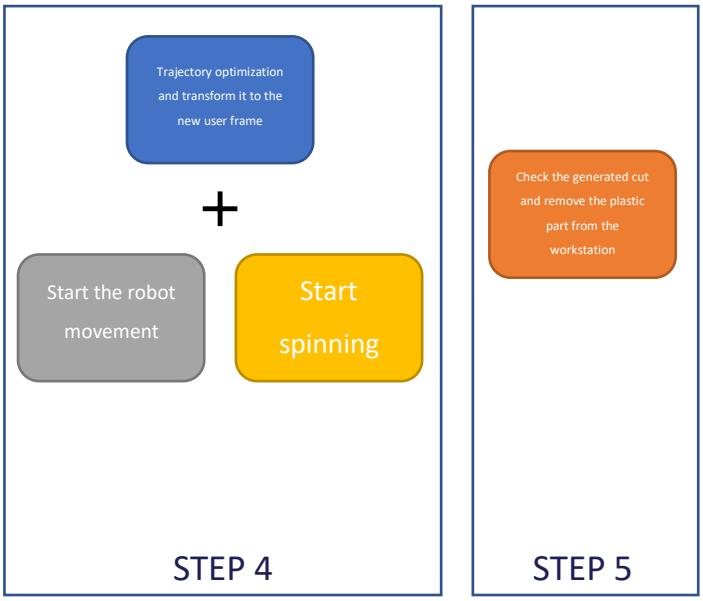


Figure 45 Updated MOSES flowchart.

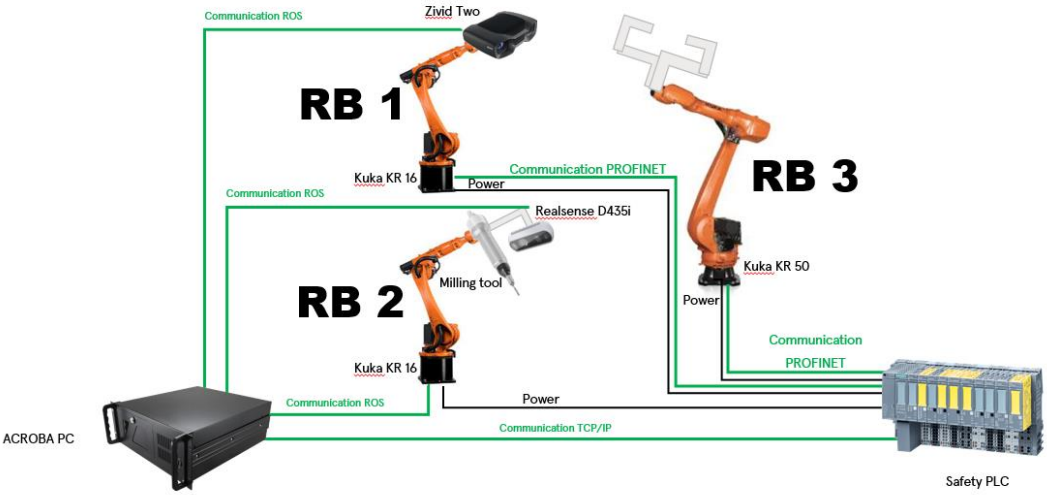
Figure 45 shows the updated flowchart of MOSES plastic cell. The steps are the following:

- STEP 1: Once the container lid to be cut has been selected, the operator takes the lid and places it on the worktable, fixing it to the tooling by means of the pneumatic grippers explained above.
- STEP 2: The system recognises the entered container lid, using the cameras and ACROBA skills the corresponding user frame is generated. In this step, the user frame is generated by locating the geometry of the inserted lid. Once the geometry has been recognised, the new user frame is inserted into the robot's memory via ROS.
- STEP 3: The AROBA system notifies the operator that the lid has been recognised and the User Frame has been created correctly. At this point, the operator, using the Dummy Tool, executes the required trajectory, the ACROBA system, using the cameras, records the entire trajectory and, once the recording is finished, analyses the trajectory, comparing it with the inserted lid.
- STEP 4: After analysing the trajectory, the ACROBA system optimises the trajectory according to the container lid selected and matches it with the User Frame created previously, so that the Robot works as precisely as possible. Once this user frame has been established, the ACROBA system executes the robot program, which starts the robot and the cutting head (spindle) until the trajectory is completed. At this point, the spindle stops, and the robot moves to the rest position until it acquires new motion instructions.
- STEP 5: Finally, after the robot has performed the necessary movements to carry out the cut, the system indicates to the operator that the lid can now be removed. The system activates the pneumatic gripper release signal, and the operator removes the cut container lid.

The sequence described above should be carried out for each of the types of lids (geometry, type of waste, colour...). Once the trajectory has been recorded using the Dummy Tool, the operator would only have to enter the cell to remove and place the cut lids.

CABKA

The following image describes the communication tree between all hardware. The PLC control the peripherals and work as a mirror with the PC for the inputs and outputs.



The sequence in the automation lines is defined in the next steps:

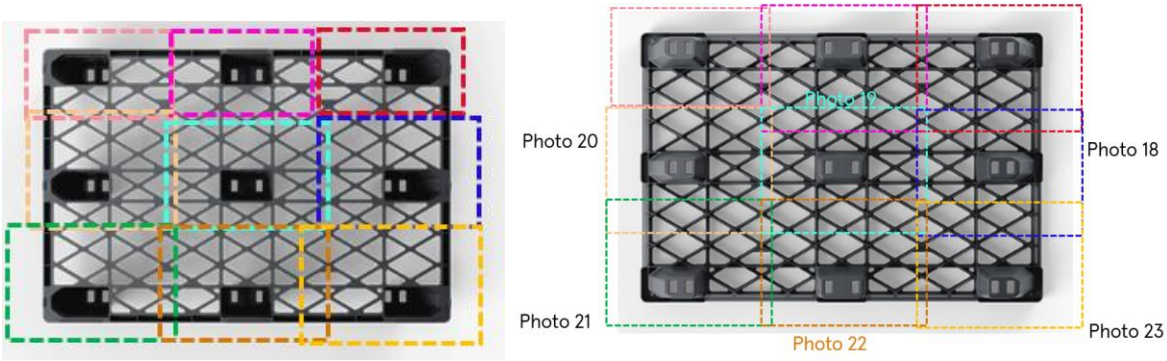
1. Operator drop the part in the belt conveyor and press the button to confirm the task complete.
2. The belt conveyor start run until the part arrive to the end, the vision station.
3. The pneumatic clamps move forward to fix the part indexed.
4. Robot 1 move to position 1 and then send the signal “ready photo 1” to Acroba. Wait in this position until the signal “Photo 1 OK” or “Photo 1 Nok” is switch on.
5. The robot 1 move to the position 2 and repeat the same process as the photo 1.
6. When all the top side of the part is completed, the robot 3 pick the part and turn it to drop it again over the vision station.
7. The robot 1 repeat the same process as the top side for the bottom side.
8. When the part has been fully inspected, Acroba send the summary and depending on the three options, the robot 3 will drop the part in different station.

a. Part OK → With this signal the robot 3 will move to the stacker station and drop the stack on top.
b. Part NOK→ This signal means that the part has lack of material and in discarded of the production. Then the robot 3 drop the part in the reject station.
c. Part with burrs→ This option is active when the part needs a deburring process. Then the robot 3 will drop the part in the deburring station.

9 The clamps in the deburring station move forward to fix the part in a indexed position. Then the PLC send the signal "Part in bottom position" to Acroba.
10 Acroba manages robot 2. This robot goes directly to the burrs and with the milling tool remove the flashes.
11 When the bottom side is complete clean of burrs, ACROBA send the signal "bottom finish" to the PLC. With this signal the PLC release the robot 3 to enter in the deburring station to pick the part and turn it to drop the part on the same station. The PLC coordinate the clamps movement in the table and in the robot gripper.
12 When the part is fix it in the deburring table and the robot 3 is out of the deburring zone, the PLC send the signal "Part in top position". Then the Robot 2 move to the remove the burrs on the top.
13 When the Robot 2 return to home position, ACROBA send the signal "top finish" and the summary of the deburring process:
    a. Part OK → With this signal, the robot 3 will drop the part in the stack
    b. Part Nok→With this signal, the robot 3 will drop the part in the reject station.

The sequence described is lineal: it is possible to start a new sequence owhile executing a sequence . For example, when the parts introduced by the operator arrive to the vision station and the part is fixed it, the operator has the release to load the next part.
The part will wait the release from the robot 3 to run inwards. Other case is when the robot 1 is doing the inspection and the robot 2 the deburring. In this moment, both stations are requiring the ACROBA attention.

The inspection of parts is split it in different zones to adapt the camera resolution with the defects size required.



STERIPACK

Currently, the SteriPack cell follows the structure described in the next figure:

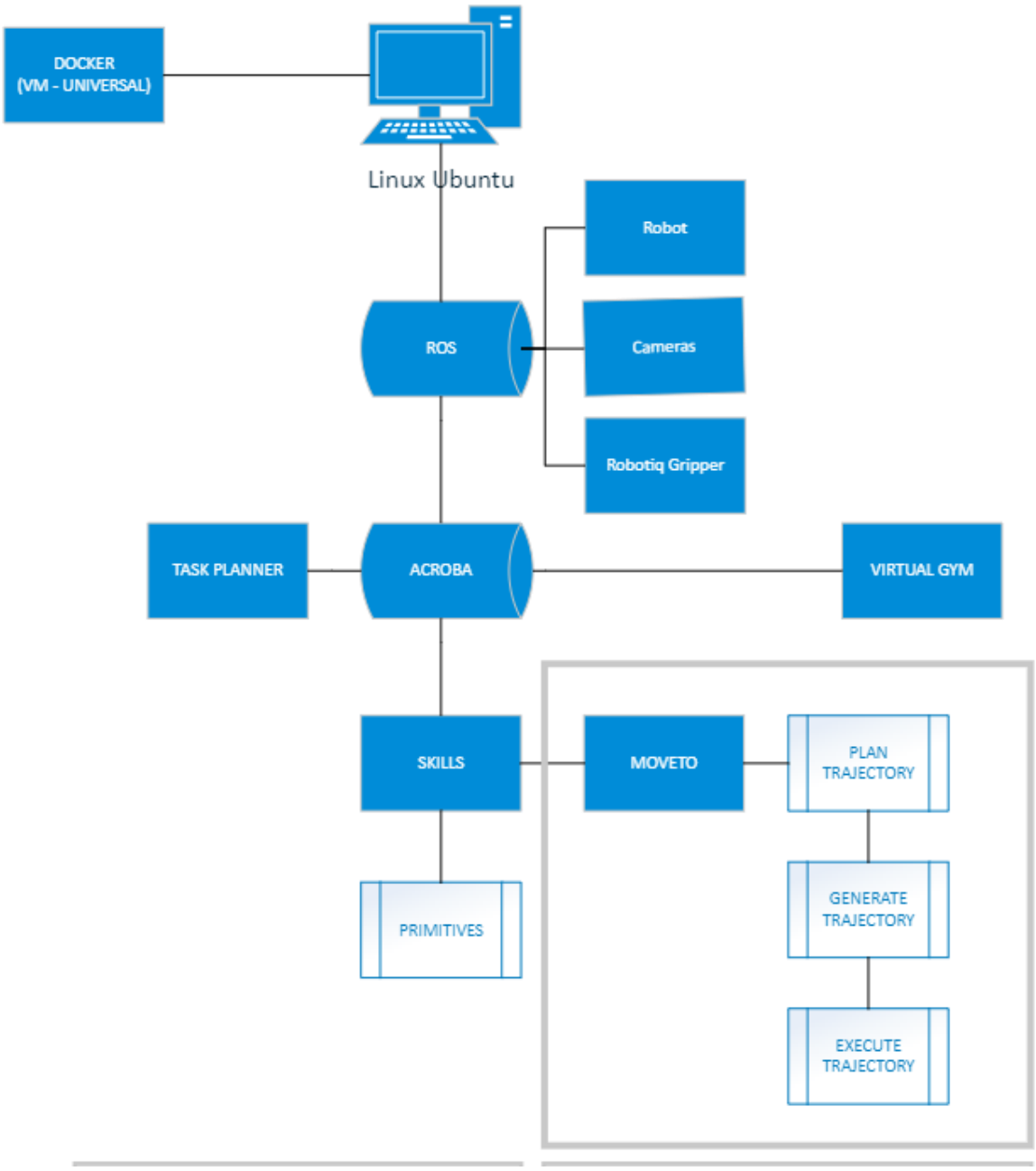


Figure 46 High level integration overview

The overall process steps and their description can be found in the next figure. It must be noted that, although the process remains the same, some steps, such as the curing and supports removal, might occur in a slightly different order depending on the type of print as well as the

type of materials used for the 3D printing process at the time. However, the overall process steps remain the same and the skills needed will also remain unchanged.

Figure 47 3D printing process steps overview

Skill Integration

SteriPack

The SteriPack cell currently has the following skills implemented:

- Moveto
- Bin picking task, which include the following skills:
  - Locate that relies on CAD matching, Point Cloud Filtering, Moveto and AI grasping
- Pick skill
- Place skill

A full sequence is currently implemented to carry out part of the 3D printing process all the way to the washing process step.

This sequence can be launched from the Ubuntu PC by following these steps:

- Open a Terminal (Ctrl + Alt + t)
- cd acroba_ws

- **source devel/setup.bash**
- Launch skills to move robot:
    - **rosrun skills start_move_arm.sh**
- Open a new window or terminal
- Source the terminal:
    - **source devel/setup.bash**
- Launch python programme:
    - **rosrun skills steripack_demo_V3.py**

CAD matching and grasping example can be seen in the following figures:

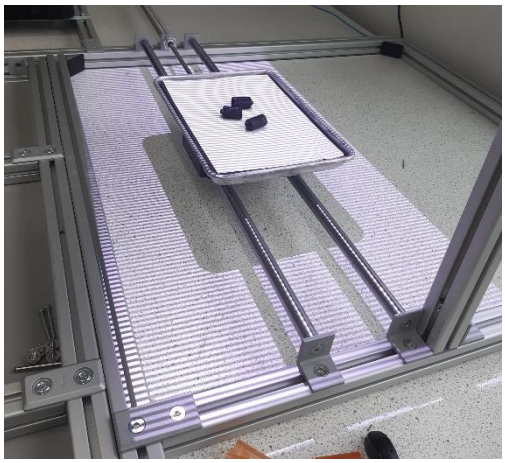


Figure 48 Zivid camera acquires imagery

Figure 49 Parts position acquired

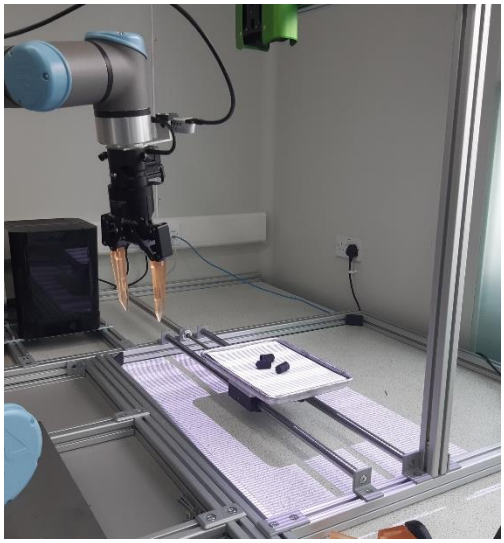


Figure 50 Grasping command sent to robot to pick up part

Testing and optimizing of the above-described skills is still on-going and the remaining steps of the process are currently being implemented., which includes supports identification (CAD matching), supports removal (Moveto) and final inspection (CAD matching).

MOSES & CABKA

MOSES cell has achieved the integration of the movement primitives and the detection primitives base on perception modules. Additionally, save and write primitive has also been tested satisfactorily.

Movement

ACROBA platform is able to move the robot around based on the skill `MoveTo` that make use of the primitives `GenerateTrajectory` and `ExecuteTrajectory`.

This group of primitives manage the the movement trajectory from a group of points to the real movement going through the generation of the `moveit_msgs/RobotTrajectory` that is the undertandable array type to the MoveIt package.

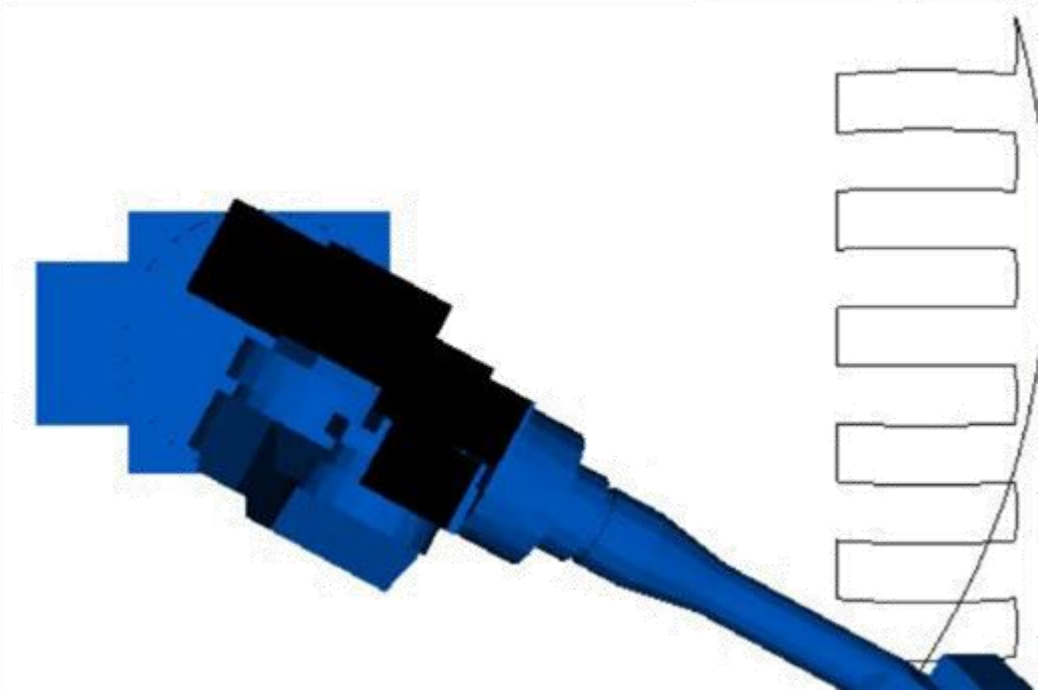


Figure 51 Execution of Trajectory using a trajectory_msgs::JointTrajectory generated by Descartes

DummyTool CAD Matching

The primitive for the CAD Matching and DummyTool tracking has been tested with the test `test_cad_tracking_and_save.py` script that use the action servers `CADTracking` and `SavePoses`. The way of working for this routine is simple, one action server is, making a cad matching from an icg file with information about the element to measure, is correction a default pose for the object in base to the object in the scene. The output of this primitive is the pose of this element that is being detected in the camera. Then, the resulting output is being passed by the second action server that will store the 6D points into a text file.

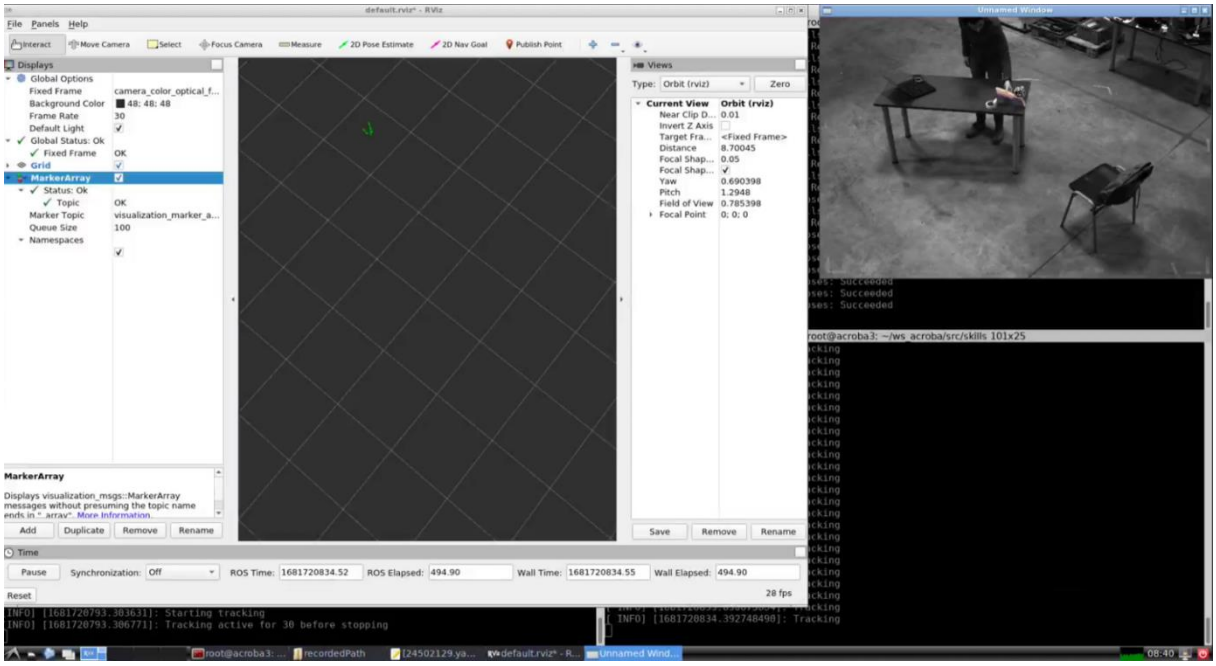


Figure 52 CAD-base tracking.

Figure 52 shows the testing process. The little box with the camera is showing the tracked element highlight, while the RVIZ window is graphing the points into a tridimensional space. On the back, the different servers are giving their outputs through console.

Dummy Tool ArUco Tracking

Using a tracking system based on ArUco markers in a bifrustum geometry instead of a tracking system based on a CAD model and best fit offers several advantages. ArUco markers are simple, black-and-white markers that are easy to print and attach to objects. They are lightweight and cost-effective compared to creating detailed CAD models and using best fit algorithms. ArUco markers can be applied to various surfaces and objects without the need for complex modelling or calibration.

In terms of robustness, ArUco markers are designed to be highly robust against occlusions, changes in lighting conditions, and partial visibility. They can be reliably detected and tracked even when they are partially obstructed or viewed from different angles. On the other hand,

tracking based on CAD models and best fit algorithms can be more sensitive to changes in object appearance or occlusions, leading to less reliable tracking results.

Another advantage of using ArUco markers is real-time performance. They can be detected and tracked in real-time using computer vision techniques. The simplicity of the markers' geometry and distinct visual features allows for efficient and fast detection algorithms. On the contrary, tracking based on CAD models and best fit algorithms can be computationally intensive and may not achieve real-time performance, especially when dealing with complex objects or scenes.

Regarding the use of multiple cameras instead of just one, there are also notable advantages. Multiple cameras provide redundant views of the tracked objects, leading to improved accuracy and reliability. By triangulating the positions of the markers or objects from multiple camera perspectives, errors can be reduced, and overall tracking precision can be enhanced.

Furthermore, using multiple cameras increases the robustness of the tracking system. In scenarios where occlusions occur or a camera loses sight of the markers, other cameras can still provide valuable information for tracking. This redundancy helps overcome challenges and improves the overall performance and reliability of the tracking system.

In this way, this test has been performed with three cameras and a DummyTool with a ArUco marker bifrustrum integrated. For this purpose, modifying their ID for being able to operate together without clashes has been needed. In **Figure 53** it is shown the three drivers working getting the images and writing the corresponding topics with the information.

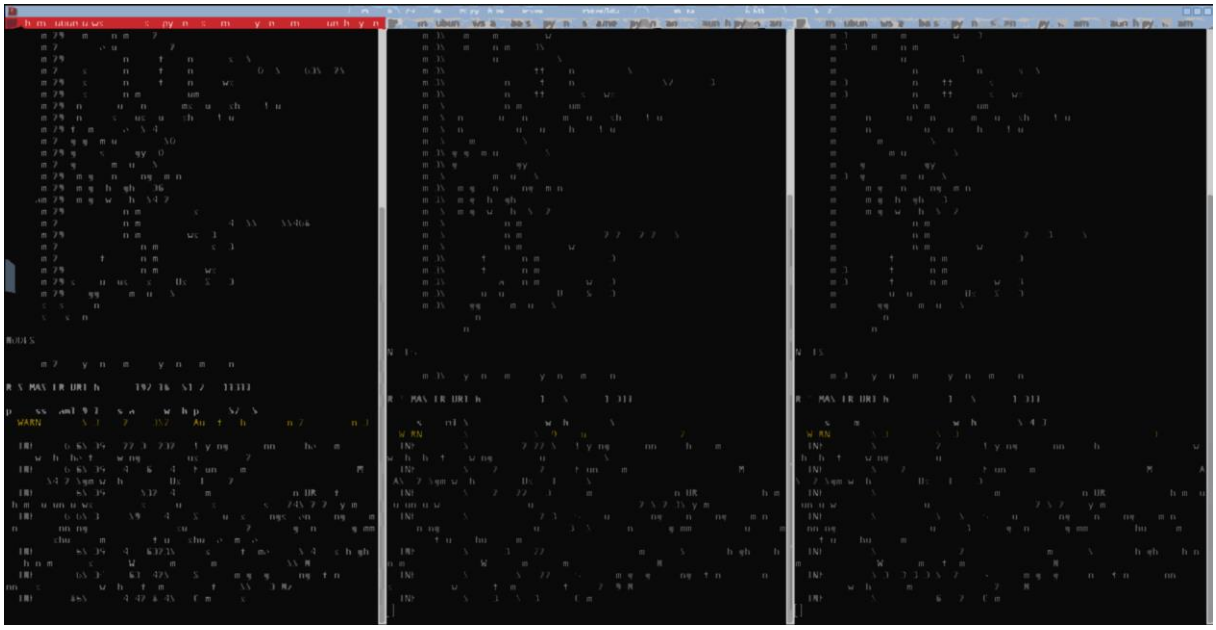


Figure 53 CAM129, CAM135 and CAM139 launched and runing in parallel

On the other hand, there has been loaded a serie of action servers:

- Primitive `DummyTool_ 129`
- Primitive `DummyTool_ 135`
- Primitive `DummyTool_ 139`
- Primitive `Position_Publisher`
- Primitive `SavePoses`

All these servers are shown in **Figure 54**. The first three are instances of the same primitive, they are in charge of reading the image topic, computing the Bifrustrum center position depending on the sent goal with the Bifrustrum ID (groups of 13 ArUcos). These primitives write in a topic the center position of the tool.

`Position_Publisher` primitive is catching these three topics and it is computing the composition of the values resulting in a `tf` array that will go as result of this action server in order to `SavePoses` primitive takes that result and writes it into a file.

The execution of this primitives is triggered through a button in the DummyTool that is connected into a TCP board that writes in a I/O topic.

Figure 54 Detector primitive for echa camera (top terminals). Collector of points and save points primitives (down)

At this point the collection primitive is having some problems in getting a good result in base of the three inputs. That is why, in the first terminal of the second row in the **Figure 54**, the result is abort while all others are resolved the action satisfactorily. This problem will be tackled and solved in the coming weeks.

Figure 55 Dummy Tool Usage

In **Figure 55** is shown the track of the DummyTool in base of one camera and is showing in the image the distance, the position, the orientation and a weight factor that should allow to the collecting primitive to compute the homogenization of the data. It is showing a red dot as well where the reprojection of the recorded point is being detected. This point is a translation from the center of the detection element base on the geometry of the part.

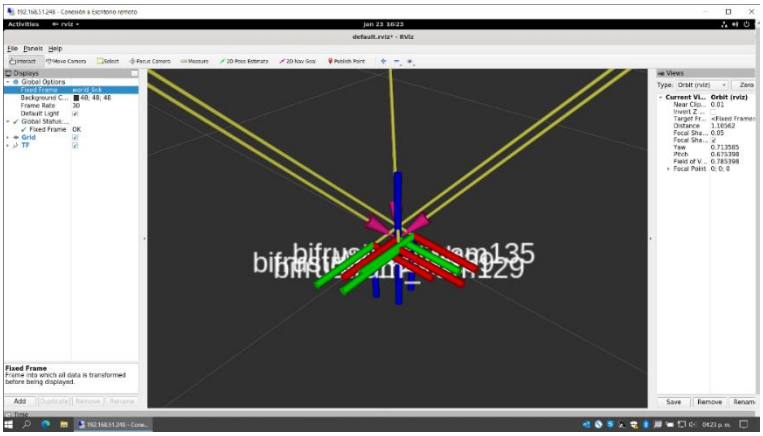


Figure 56 Multicamera track

Coming back to RVIZ, in **Figure 56**, the interface is showing the `tf` of the detected bifrustum coming from every camera. Here it is possible to see the misalignment that should be corrected using the collection primitive and the group of cameras.

These points are as well registered in the marker topic in order to be represented in group by RVIZ as is shown in **Figure 57**, where has been recorded the whole table with the metal plate in a side in order to be able to check dimensions and proportions.

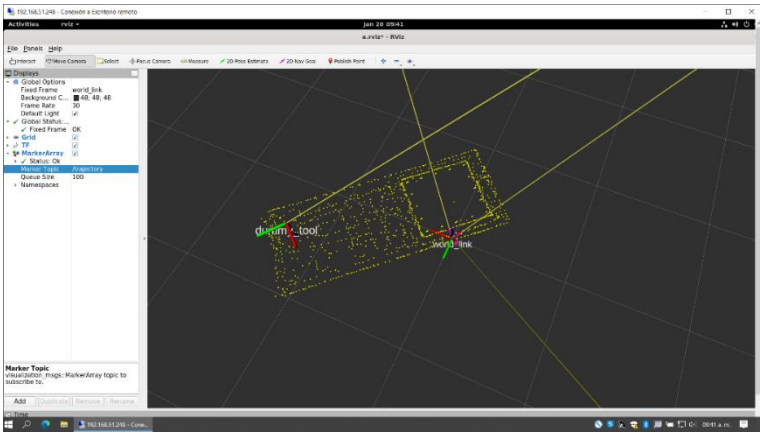


Figure 57 RViz with captured point visualization

Conclusions

This deliverable includes all the actions carried out during task 4.2. For this, in the first place, each of the use cases (MOSES, CABKA and SATERIPACK) have been explained in detail, the integrated cell designs have been explained, addressing all the general aspects, mechanical, electrical, pneumatic and security aspects.

Going into more detail, the integration of each of the devices has been explained, taking into account the type of robots required for each of the cells (YASKAWA, KUKA and ABB), the different cameras installed in the cells, taking into account what will be its use once the ACROBA system is completely installed. The integration of each of the sensors included in these cells has also been addressed, from the mechanical point of view (hardware integration) and from the computer (software) point of view. In addition, the actions carried out to bind ROS with the operating systems of each of the robots, as well as Moveit and Pylon, have also been explained.

All actual workflows (after ACROBA implementation) for each of the use cases have been addressed. And finally, the successful integration of some skills has been explained.