6DMIMIC

User and Developer Manual

June 9, 2022

INESCTEC - CRIIS

Contents

1. Hardware

1.1. Architecture

The laboratory set-up is depicted in Fig.1.1 and it is constituted by the following main components:

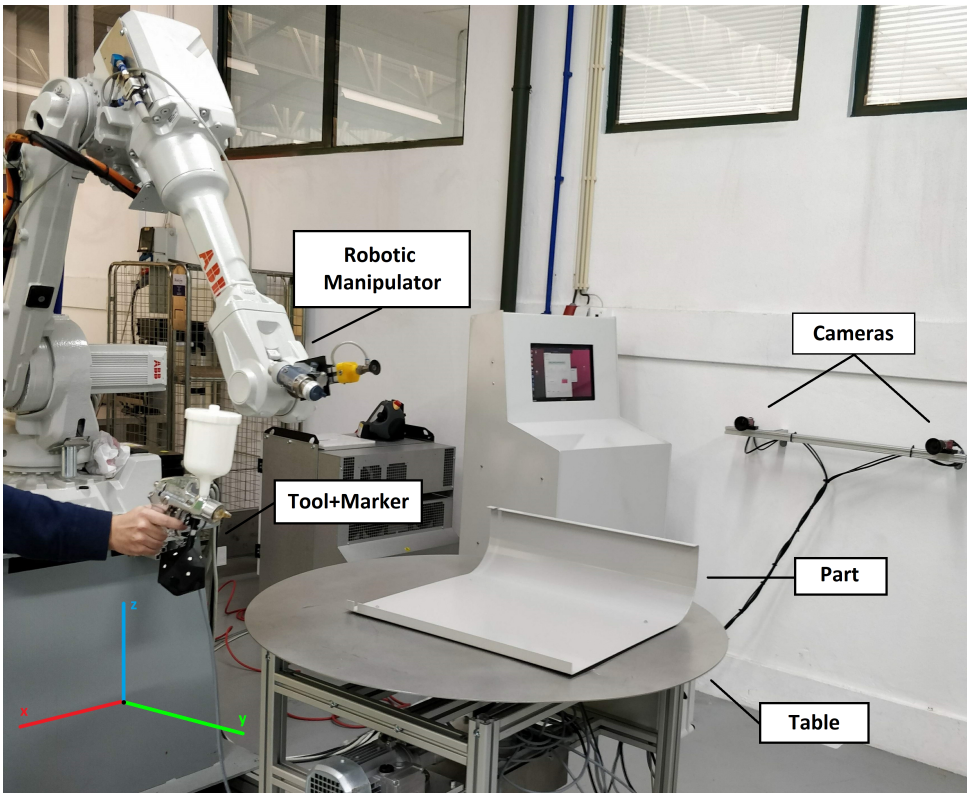


Figure 1.1: System Hardware Architecture and its interconnections.

- an **ABB IRB 2600 industrial manipulator** and its controller. This robot is equipped with an automatic Spray Gun, an AirPro from GRACO, which is actuated by

solenoid valves and the coating paint is injected through a pressurized vase. These two valves are actuated by the ABB controller directly.

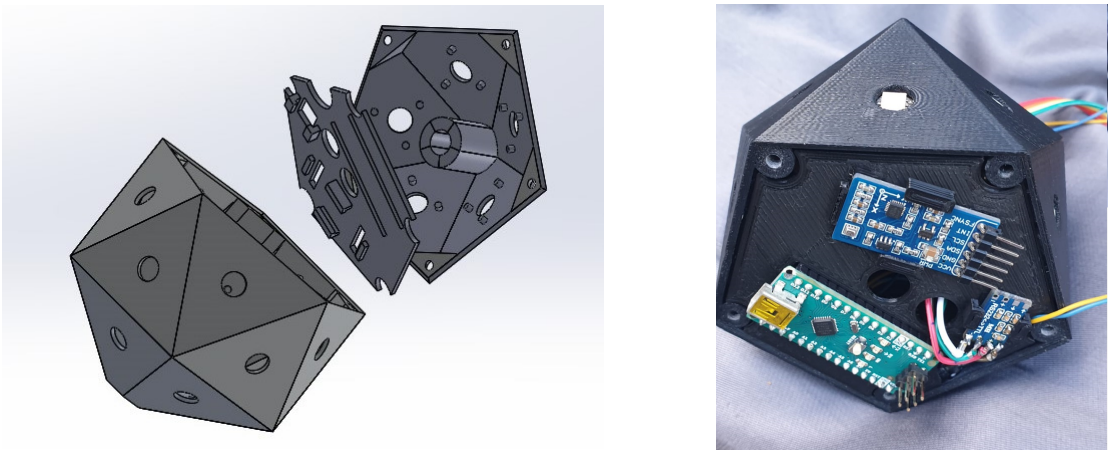
- **automatic rotating table**, with an incorporated PLC module. It allows the operator to always be painting on the same position, and just uses a pedal for the table to rotate the part. The rotating mechanism is placed in the center of the table and is actuated through an induction motor. To ensure that the table rotates exactly 90^{o}, a magnetic sensor and 4 plates were strategically placed under the top. This is controlled by an adjustable frequency AC drive connected to a Programmable Logic Controller (PLC). The PLC is in charge of dealing with the signals coming from the pedal, the manual coating spray gun on/off switch, the magnetic sensor and the signals from the ABB controller. In addition, it sends signals regarding the switch and the pedal to the SincroBox to inform when the operator acts each one of these. The PLC is an M-DUINO 42+ PLC from Industrial Shields.

- **the 6D Mimic system, endowed by an IMU**. The 6D Mimic is mainly constituted by (1) an industrial PC (NEUOSYS Nuvo-3005TB) where the machine vision processing software is installed, (2) the Sincrobox, whose main purpose is to be the central component of the system making the communication bridge and synchronization between all the signals (e.g. the pedal and manual spray gun switch signals) from the different hardware components of the entire solution, (3) the Stereoscopic vision system, constituted by two industrial cameras (Mako G125C PoE), mounted on a fixed rail, and that are capable of seeing the working field of the robotic arm, and finally (4) the 6D Marker, that is attached to the manual coating spray gun, as well as a simple switch to check whether the operator is pulling the trigger or not. Also, inside the 6D Marker, support for the IMU and the rest of the electronics associated with it was added, which can be seen in Fig.1.2. This piece supports an Arduino Nano, a TTL to RS232 Converter, and an IMU with 10 Degrees of Freedom from Waveshare (MPU9255) set to its highest possible bandwidth.

NOTE: In order to upload a new program to the arduino nano, the RX and TX pins need to be disconnected. The compiler's configuration should be:

- Board: Arduino Nano
- Processor: ATMega328p (not old Bootloader)
- Programmer: AVRISP mkll

- Board Manager-> Arduino AVR Boards v.1.6.21



(a) (b)

Figure 1.2: (a) CAD Representation and (b) Real 6D Marker with Support

The final solution's hardware architecture is depicted in 1.3.

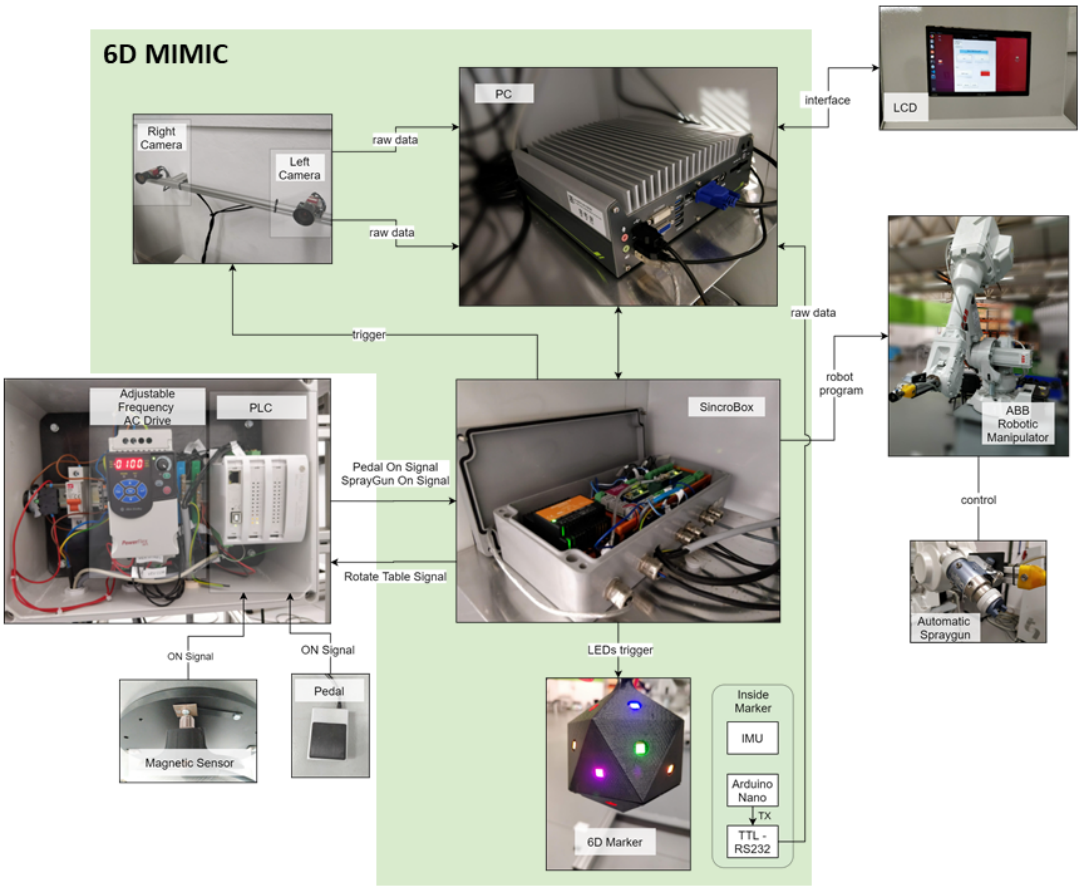


Figure 1.3: System Hardware Architecture and its interconnections.

2. Software

In this section, all software/firmware setup will be discussed. It will be presented the computer, robot and sincrobox setup procedure.

2.1. Setup Prerequisites

- Lazarus 1.6 or higher. In Ubuntu download Synaptic Tool and find Lazarus 1.6. Automatically all dependencies will be set do download either. Then, download and install it with the synaptic tool. (x86 version is recommendable)

- External components for Lazarus, namely: epiktimer; GLSceneLCL; lnet.

  Source can be found at:

  ```
  https://github.com/graemeg/epiktimer
  https://sourceforge.net/p/glscene/code/HEAD/tree/branches/GLSceneLCL/
  https://lnet.wordpress.com/download/
  ```

  These packages can be installed using the Lazarus' Online Package Manager in Package>Install/Uninstall Package. If some, or any, of these packages is not found into the tool, use the manual procedure described in the "Package Setup" section.

- SDK for Allied Vision's GigE Vision cameras. (driver used by the SdpoPvApi wraper). UBUNTU: Download the dynamic libraries LibPvAPI.so and LibPVJNI.so and place them into usr/local/lib. WINDOWS: Just run PvAPI_win_1.28.exe

  This package can be found at:

  ```
  https://www.alliedvision.com/en/support/software-downloads.html
  ```

5dpo components for Lazarus, namely: SdpoDebayer; SdpoDynmatrix; SdpoFastForm; SdpoFreenect; SdpoJoystick; SdpoPvAPI; SdpoSerial; SdpoVideo4L2. Use the manual package procedure described in "Package Setup" section.

Source can be found at:

```
https://sourceforge.net/projects/sdpo-cl/
```

- Visual Studio Code or Arduino IDE (or equivalent) to transfer the arduino firmware to the board.

2.2. Package Setup

The modules package can be found at `https://gitlab.inesctec.pt/CRIIS/6D_MIMIC` and are compiled using the Lazarus with the required packages. To do a package setup in a current module, open the module's project go to the Package>Open Package File (.lpk) find the required package .lpk file and open it. In the pop up window, compile the package and select Use>Add to project. Then, just compile the project and run it. For FORM the packages are need to be installed in the Lazarus. Therefore, select Use>Install, wait and reload the Lazarus (it is recommended to use Lazarus with sudo in case of /usr/lib installation).

2.3. Arduino Setup

« TODO: improve this section »

To the connected arduinos have the same name every time they are reconnected, to the /dev/udev/rules.d and copy the script 99-arduino-sincrobox-udev.rules. Adapt any rule in this file if necessary.

The arduino firmware is placed in SincroBox folder. Upload Arduinos' nano firmware using Visual Studio Code, Arduino IDE, or equivalent.

2.4. Robots Setup

« TODO: improve this section »

2.4.1. ABB models

Configure ABB IOs at RobotCodeGen/codeGen/omappingABB.cfg. This file is structured by signal type following by the port name (the comma is the delimiter). The supported types are LEVEL and TRANSITION.

The robot need to be running with mimic codes (in Robot Programs Folder, Section 2.5.3). This code allow to the robot be controlled by the HMI module. See Section 2.5.1.1

Running the robot programming

The ABB robot's programming need to be running for Calibration and Usage process. The steps of the process is describe as follow:

1. Turn on the ABB controller (top button - ON position). See Figure 2.1;
2. Check if the safe button is unpressed. See Figure 2.1;
3. Unlock the motors by pressing the White button. See Figure 2.1;;
4. Set it to automatic mode (Key button - left position). See Figure 2.1;;
5. In some cases, an advice window will pop up in the ABB console. Press 'Acknowledge', then 'Yes';
6. On the console: set the robot speed to 25 % (safety precaution) by clicking the button on the bottom right corner, selecting the second to last option, then '25%'.See Figure 2.2;
7. Open the ABB menu (top left), see Figure 2.3;
8. Click "Production Window" , see Figure 2.3;
9. Check the program name (top) to make sure it's the 6DMimic program, otherwise select the program ;
10. Press the button "PP to main", see Figure 2.4;
11. Press the Play button on the console (physical button), see Figure 2.4. Now the robot's program is running

Figure 2.1: ABB Controller

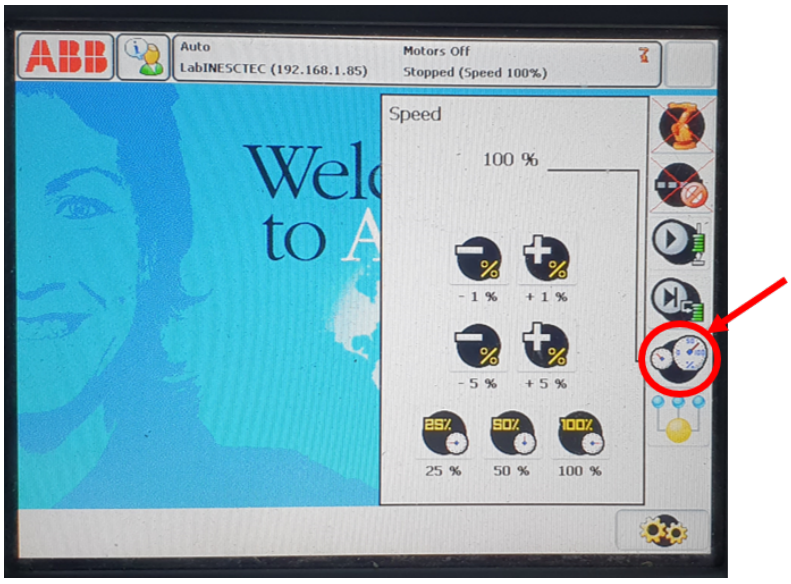


Figure 2.2: Selecting safety speed

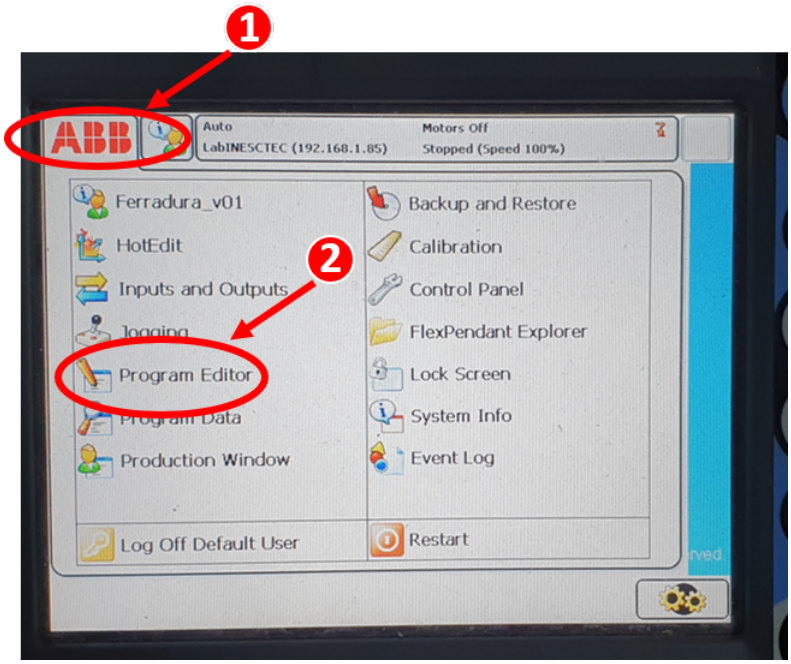


Figure 2.3: Selecting the program

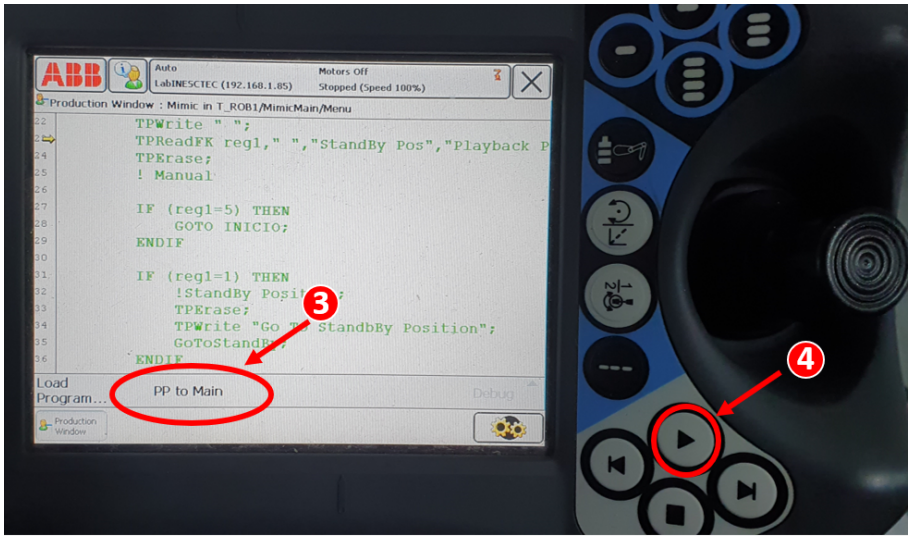


Figure 2.4: Running the program

2.5. Architecture

The 6DMimic project is composed by Modules, Scripts, Robot Programs and Sincrobox Folder distributed in the processing computer, the robot and the sincrobox (see Chapter

1), respectively. These components are presented in next sections and comprise the two steps of the 6DMimic: the calibration and the mimic procedure. The calibration is necessary to setup the cameras, and the marker detection to be, latter, used in the mimic procedure.

The main software architecture of the calibration and the mimic system is divided in programmed modules in Pascal. Figures 2.6 and 2.5 represents the software architecture for the mimic and calibration process, respectively.

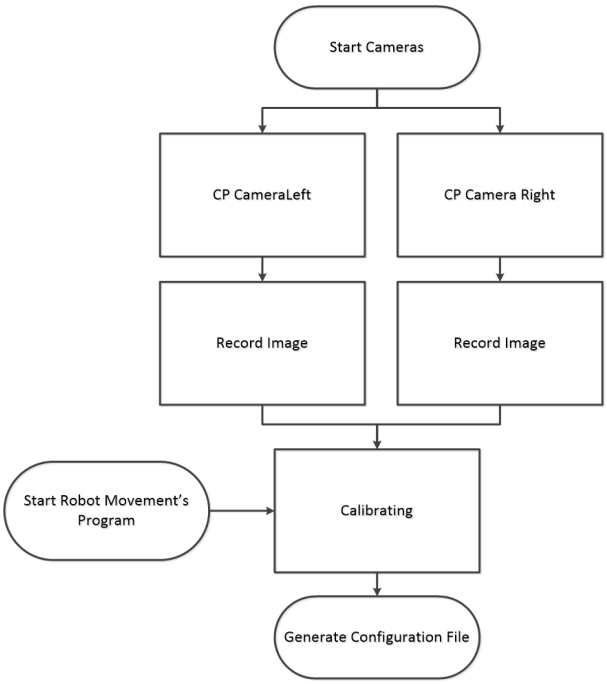


Figure 2.5: The 6DMimic modules control flow for stereo calibration

All the project can be found at

```
https://gitlab.inesctec.pt/CRIIS/6D_MIMIC
```

2.5.1. System Modules

HMI

The HMI is the GUI application to start the Mimic system. It has the option to start recording the movement (Demonstration), automatically send a program to the robot and run the it (Start Mimic). It also allows remote control of the ABB robot (A specific robot program needs to be running in order to accept communication). For Yaskawa

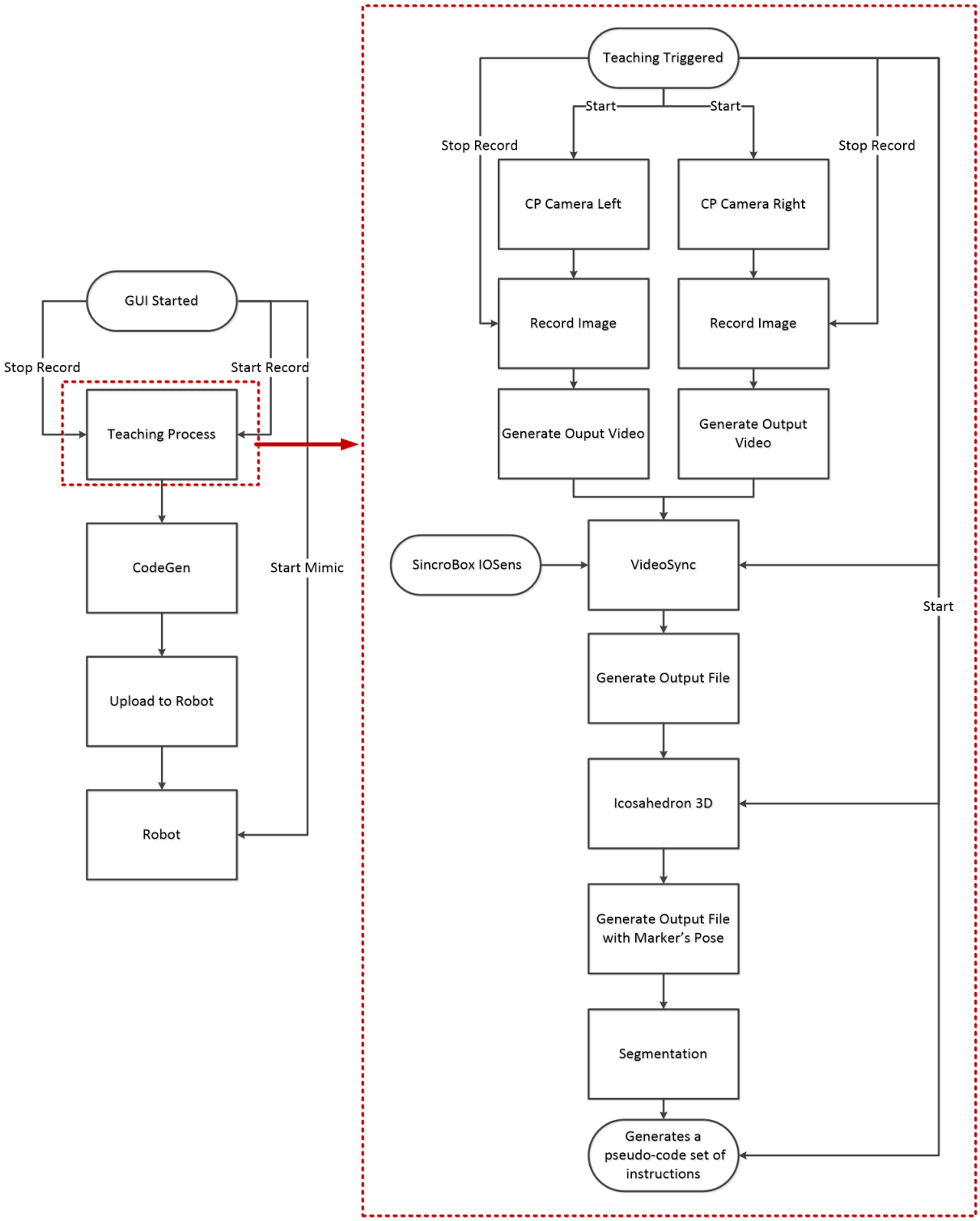


Figure 2.6: The 6DMimic modules control flow for mimic.

models, it is necessary to manually transfer the code. The usage have a more detailed description in Section 4.

Teaching

Management module. It is responsible for integrating and execute the 6DMimic pipeline. Standalone usage: the module is set to run with a given, thus run:

```
./SA -p "profile_name"
```

Camera Player

The CameraPlayer is responsible to grab camera images, allows live video, record raw video and, adjust cameras' parameters. This module is used in Calibration and Mimic process.

To be correct played, two instances must be run, one for each camera, as bellow:

```
./CameraPlayer -d "profile_name"
```

Each instance of the app runs on a pre-defined profile that needs to be passed as a executable's parameter. Failing to explicitly provide this will default to profile "data". Each profile has a dedicated folder located in: /CameraPlayer/"profile_name". For instance, "iilableft" and "iilabright" are the profiles for each camera on the system.

CPCalib

The CpCalib is a fork of CameraPlayer used for 6DMimic system Calibration. Therefore, there exists one instance for each camera. This module has the same functionalities of CameraPlayer but it's optimized to track one colored LED (cluster) in the image. It records a raw video or a data structure with the coordinates of the cluster. It uses the same profile folder structure as CameraPlayer and, the same command line.

VideoSync

Module that synchronizes images from two cameras and the SincroBox IOSense.

Icosahedron3D

The Icosahedron3D is responsible to track the 6DMimic icosahedron marker, i.e., allows live marker tracking. Therefore, it is used in stereoscopic camera and marker orientation calibrations.

```
./sceptreProc -d "profile_name"
```

Segmentation

Module that generates a list of positions, in pseudo-code. Reads the data from Icosahedron3D, i.e, a set of poses, applies filtering, and integrates I/O activation into a single pseudo-code instruction set.

RobotCodeGen

Module that generates a robot program given a set of positions in pseudo-code. For now two languages are supported: INFORM and RAPID, ".JBI" and ".mod" file extensions, respectively.

2.5.2. SincroBox Folder

« TODO: improve this section »

The SincroBox electronics firmware is composed by three modules: two programmed in C to Arduino and embed into it (SB_IOSsense and SB_Triggers); one programmed in Pascal to be run in a computer (SincroTriggerConfig). They will be explained in the next subsection.

SB_IOSsense

The I/O acquitision device.

SB_Triggers

The device that generates the synchronized signals for cameras and the marker.

SincroTriggerConfig

A GUI app that allows fine tuning of the timing parameters on SB_Triggers.A serial communication channel must be provided between the PC and the SB_Triggers Arduino Nano. « TODO: maybe check if exist any trick in this part »

2.5.3. Robot Program Folder

In this folder is located all robot programs (for ABB and Motoman/Yaskawa models) necessary to run the 6DMimic system. No instance of the 6DMimic system tries to access this location. The Motoman folder has only the stereo calibration movement program. The ABB folder, besides the calibration movement program, also locates the server folder, allowing to the robot be commanded by HMI application.

2.5.4. Scripts

The 6DMimic system is organized in modules, however its execution and usage are made by several shell scripts. The main ones are presented as follow:

- **Record Calibration Points:** It is used in Stereo Calibration. It runs CpCalib modules for both cameras;
- **Stereo Calib:** It is used in Marker Calibration. It is loaded with calibration data from the stereo calibration procedure. It runs Icosahedron3D module;
- **6DMimic:** It runs the 6Dmimic iuns the entire 6dMimic system. More specific it loads the HMI_RUN and Move Capture Auto scripts
- **HMI_RUN**: It loads the GUI interface of the HMI module. HMI provides a user friendly GUI to use all the features (recording, generating program, upload to robot and remote robot control).
- **Move Capture Auto:** It loads the Teaching module with auto configuration «???». Teaching controls acquisition. This app records the movement and generates a pseudo-code position list. Use this if robot code generation is not needed.

Auxiliary scripts file are also used, like:

- **calibRight** and **calibLeft:** It is used to calibrate only one camera. It runs CpCalib modules one of the cameras;
- **Run Left Camera** or **Run Right Camera:** It is used to run a specific camera interface. It runs the CameraPlayer module;
- **Terminate Apps:** It simply kills all processes bound to 6DMimic, in case some of them are dangling. Use it if anything goes wrong;
- **Move Capture Manual:** It loads the Teaching module with manual configuration «???».

3. Calibration

3.1. Hardware Configuration

Before initiating the calibration procedure, it is necessary to make sure the connections on the SincroBox are correct, including the calibration LEDs. To correctly configure the hardware, the following steps should be taken (SincroBox pinout in Figure 5.3):

1. Disconnect the cables that are shunting the P9.1/P9.2 and P1.1/P1.2 (Important note: completely disconnect and remove these cable as they have a permanent 24V voltage running through them);
2. Remove the 6DMarker connection on pins P6.3/P6.4;
3. Carefully connect the calibration LEDs into the P6.3/P6.4 pins taking into consideration their polarity (P6.3 - VCC / P6.4 - GND);
4. Carefully connect the ABB cables to trigger the calibration to pins P9.1/P9.2 taking into consideration their polarity (P9.1 - VCC / P9.2 - GND);
5. Make sure the LED cables have enough length to reach the calibration position and the manipulator's motions. Assure that there is no risk of the cables getting stuck in nearby corners.

After executing these steps, the calibration procedure can start. When it is finished, all the previous connections should be re-established (P1.1 shunt with P9.1 and P1.2 shunt with P9.2, 6DMarker cables connected to pins P6.3 - VCC e P6.4 - GND).

3.2. Stereo Calibration

The Stereo Calibration consists in moving the robot in 3 dimensional space, like Figure 3.1, with a Green Calibration LED attached in the robot's TCP and run the calibration

acquisition of the stereo vision. Thus, the system will automatic generate the calibrated parameters since the robot pose is already known. A detailed description can be checked at [**?**]. The focus of this section is how perform the Stereo Calibration.

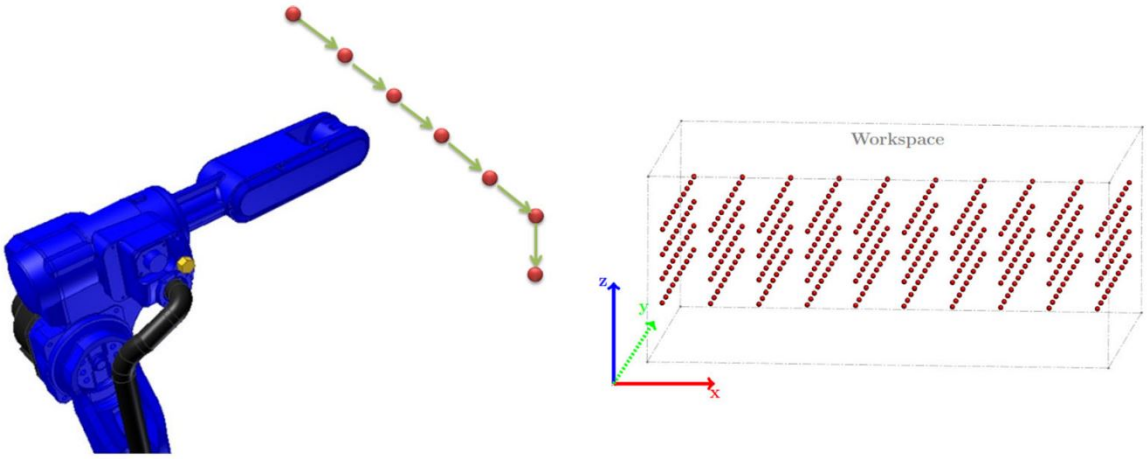


Figure 3.1: Robot's movement during the stereo calibration process

First it is necessary to build the calibration hardware. Disconnect the icosahedron marker from the P6.3 and P6.4 of the SincroBox IOSense Module. In these pins, connect the Green Calibration LED. Check Section 3.1 for detailed description.

Then, program the robot to move inside the working area and lit the LED at specific/known positions. It is also possible to use, or use as reference, the already made robot programs inside the Robot Program Folder, SVCalib method in MimicMain.mod. Follow the procedure described in Section 2.4.1.1 to run the robot program. (Section 2.5.3).

In PC, run the **Record Calibration Points** script to start the calibration procedure. Two instances (with two associated GUI interfaces) of the application "cpCalib" will start, one for each camera. Do the following settings in each instances (Figure 3.2):

1. Set the cameras' device number if they have been changed;
2. Set the check box "Proc"
3. Set the check box "Clusters Only";
4. Set the checkbox "VideoActive";

Hit Record to start the calibration process and run the robot's procedure for calibration, however **BE AWARE OF**:

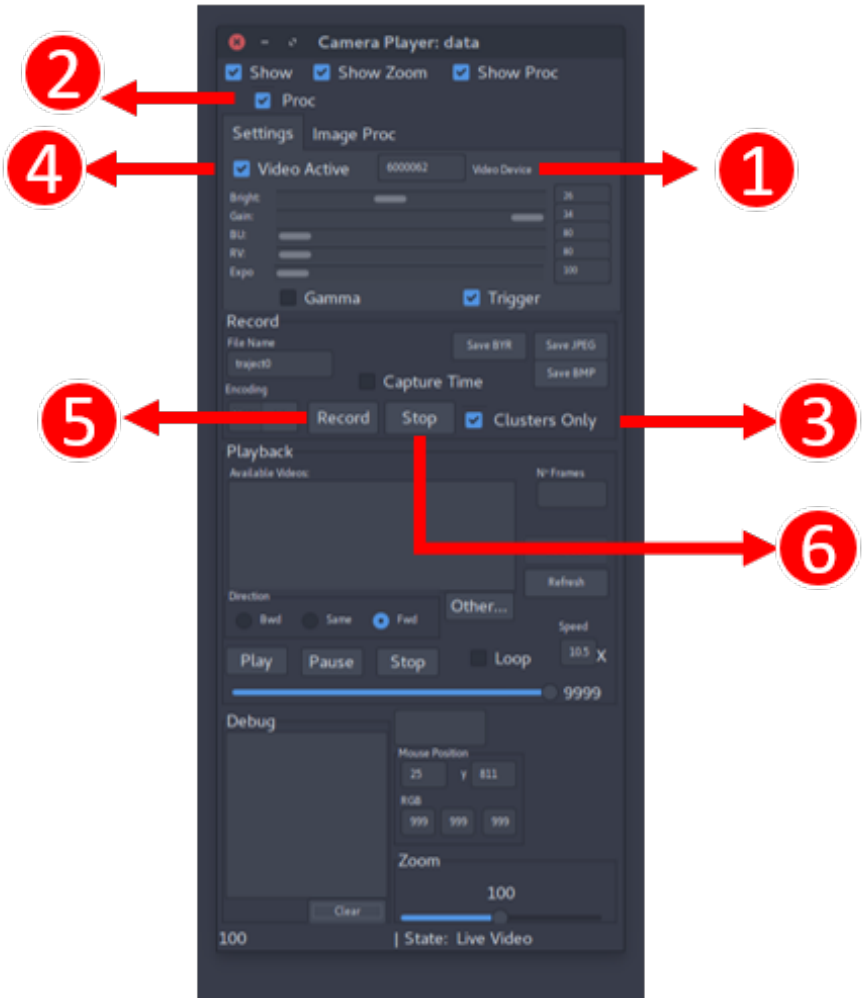


Figure 3.2: The steps of cpCalib window configuration

- When you click the calibration button, the robot will start moving;
- Check if you have all the system prepared for the calibration movements and the area around the robot is cleared.
- To be safe reduce maximum velocity of the robot's motors in the teaching pendant.

Stop recording when the robot finish the movement program. The outputs will be located at at /cpCalib/[camleft|camright]/clusters. Inside this, find the list of detected led positions.

Then, it is necessary to build a list of X,Y,Z positions according to the robot's calibration procedure. Using as reference the file calibpoints.cs of Icosahedron3D/iilab, copy the points of the ".pts" files from both, the left and right cameras of /cpCalib folder and place them in the copied ".csv" file, (see Figure 3.3),. Some hints bellow:

1. the file column's separator of ".pts" file is a single space;
2. file name needs to be "calibpoints.csv";
3. the data is structured in ".csv" as: WorldCoordinates(x,y,z) RightCamera(u,v) LeftCamera(u,v), see Figure 3.3;
4. the WorldCoordinates(x,y,z) are the poses of the robot. If the robot program movement has been changed, these file's values need to be changed.
5. after creation/modification of the ".csv" file on its proper editor, open it with simple text editor, and replace every "," with a space " ". The 6DMimic does not support the comma.
6. cameras' calibration point list have an additional column with timestamp. Use this to match/synchronize left and right cameras' points.

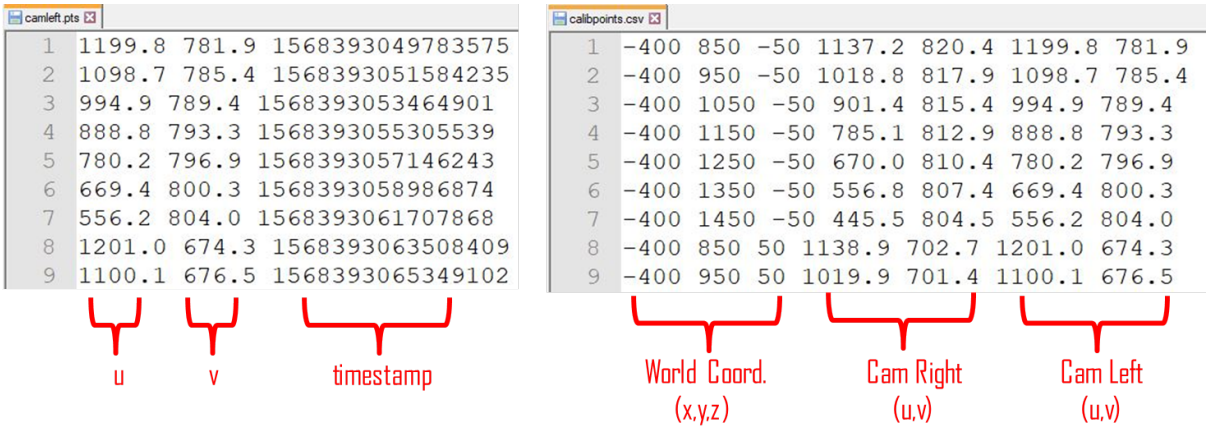


Figure 3.3: Examples of ".pts" and ".csv" files

Once the calibpoints.cs is made store it into /Icosahedron3D/iilab folder (note that iilab is an example profile folder used by latter codes, modules and scripts). Inside this folder, remove all old calibrations files if it exists: "F.mat" and "cameras.grd".

Run **Stereo Calib** script ignoring the warning pop-up saying that Fundamental Matrix is undefined. A GUI will be shown, then press "Calibration" button: a new window will pop up, do the following (see Figure 3.4):

1. insert the file name, calibpoints.csv
2. press "Load All"

3. press "Fundamental Matrix" and "3D"
4. press "OptK1"
5. repeat "Load All"
6. press "Fundamental Matrix" and "3D"
7. press "OptK1K2UcVc"
8. repeat "Load All", "Fundamental Matrix", "3D"
9. check the maxABS error and evaluate result

Now the stereo calibration procedure is realized, and the next step could be done: the Marker Calibration.

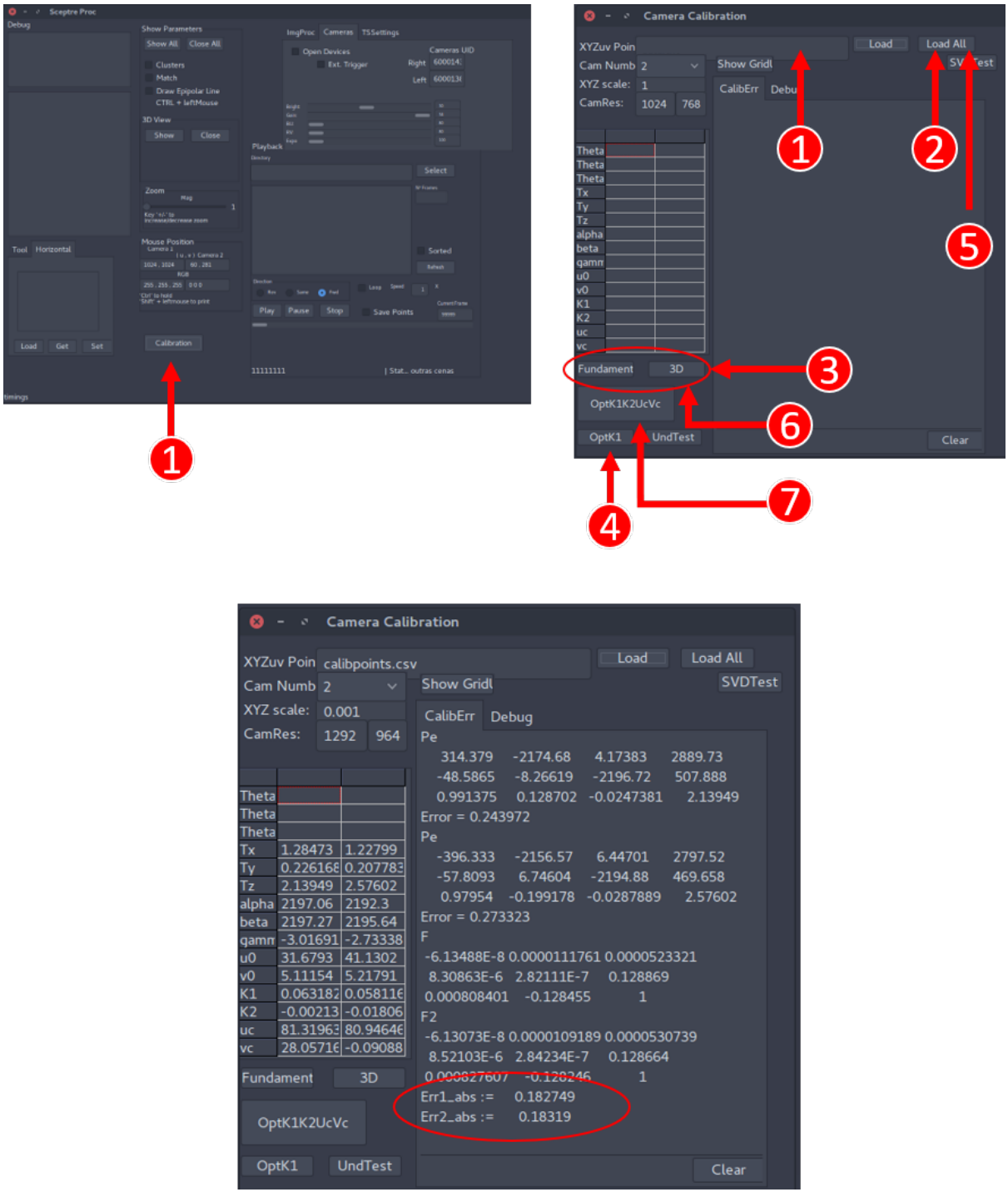


Figure 3.4: Steps of the StereoCalib's window procedure

3.3. Marker Calibration

This procedure is needed to calibrate the marker offset related to the robot's TCP, see [**?**]. To perform this procedure, the stereo camera needs to be calibrated before and the Icosaedron Marker need to be attached into robot's TCP. Follow Section 3.1 to installed icosaedron to Sincrobox.

First, provide at Icosahedron3D/iilab the "tip.pnt", a text file with the offset from the marker centre to the pistol tip. Run the **stereo Calib** script. In the GUI window, check "opendevices" and "exttrigger". Constant values must appear in the left-center window at the lower left corner. Then, define the "Tool" and "Horizontal" rotation matrices. To do this do the following (check Figure 3.6:

1. activate cameras and light up the marker using the GUI;
2. move the robot to a known position;
3. fill the "robotStandBy.rot" file with the orientation of the robot at previously position;
4. manually, put the manual spray gun with the same orientation as the automatic (robot) spray gun, see right image of Figure 3.5;
5. capture the marker orientation, using the button "Tool -> Get-> Set". This records the marker orientation when the manual and auto spray guns have the same orientation
6. move the manual spray gun to a position where the TIP offset has been defined, see left image of Figure 3.5 ;
7. record the orientation using the button "Horizontal-> Get-> Set";
8. **note**: make sure to not obstruct the LEDs during the procedure;

Now the calibration procedure is finished.

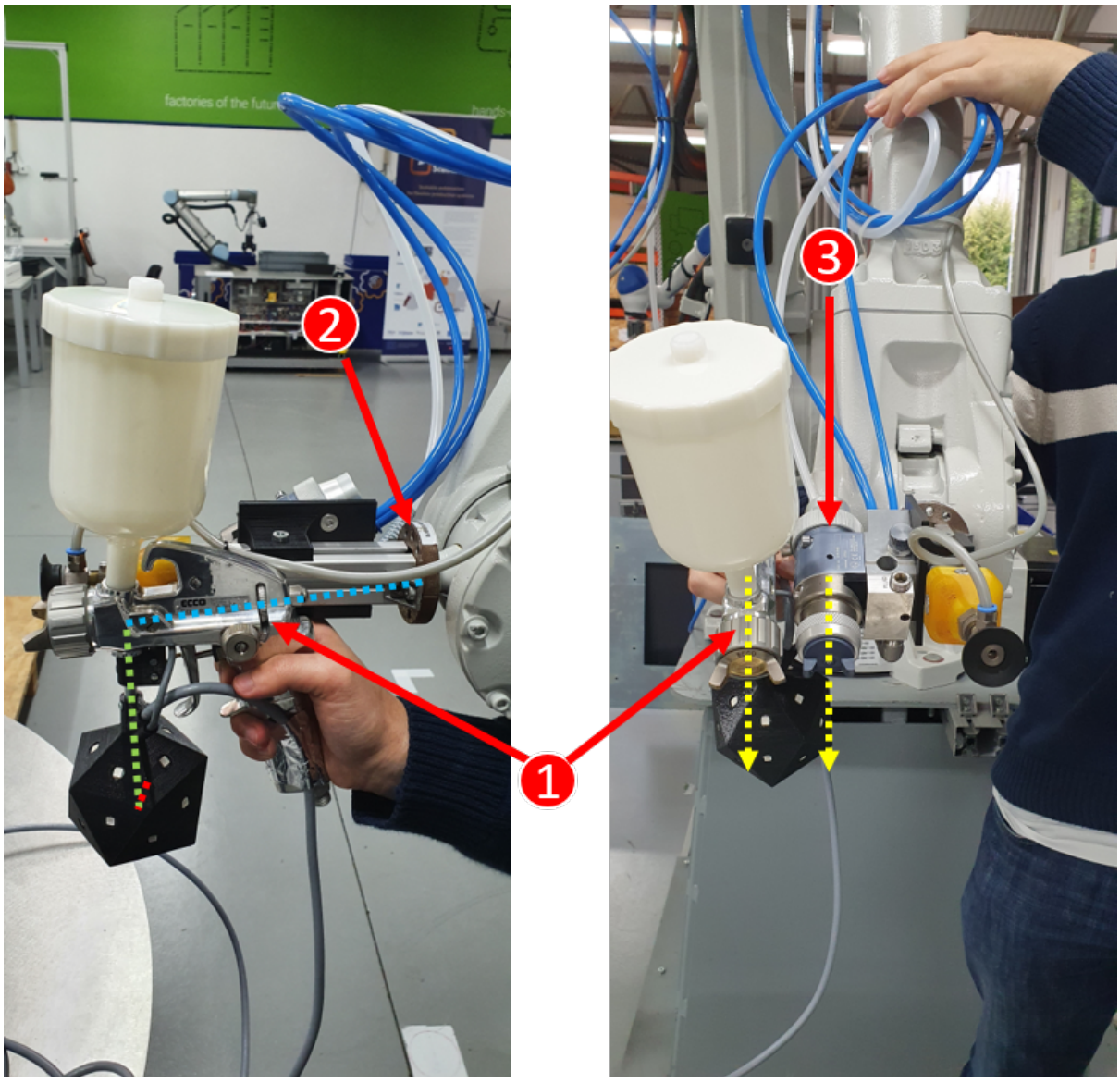


Figure 3.5: Alignment of the tools. (1) The manual spray gun; (2) Robot's wrist used as reference by the TIP offset (RGB dashed lines) (3) Automatic spray gun (yellow dashed arrows show both guns' orientations aligned)

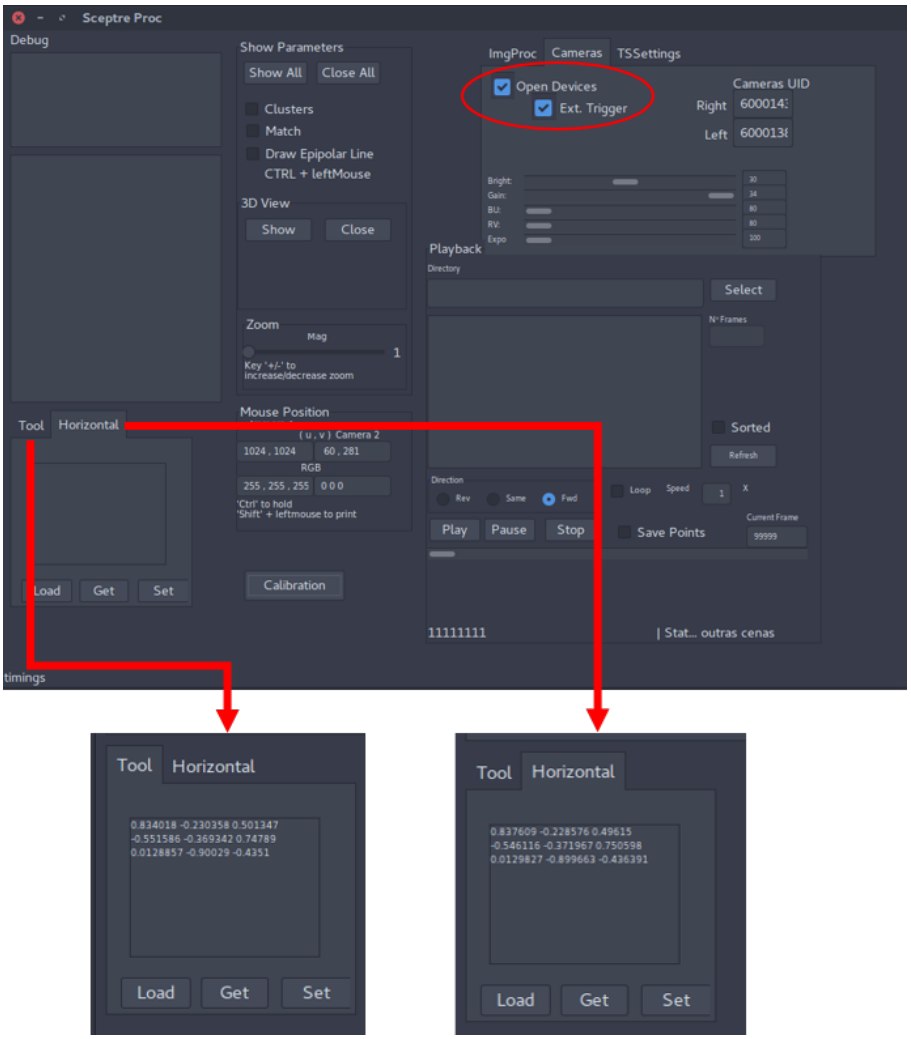


Figure 3.6: Example of StereoCalib's window after the Marker's calibration

4. Usage

Before the 6DMimic usage, the hardware system needs to be correctly configured. It also needs to be properly calibrated, see Section 3.

Two modes exist to run the 6DMimic process: the complete and the teaching. The teaching mode is only used to generate a pseudo-code without any translation. It is executed using the **Move Capture** script that opens a simple interface where it is possible to start and stop the record the demonstration. It is also possible to adjust some parameters and debug the system. In the end it will generate a pseudo-code inside the Teaching folder. The complete 6D Mimic is executed by the **6DMimic** script following by a GUI interface opening. See Figure 4.1.

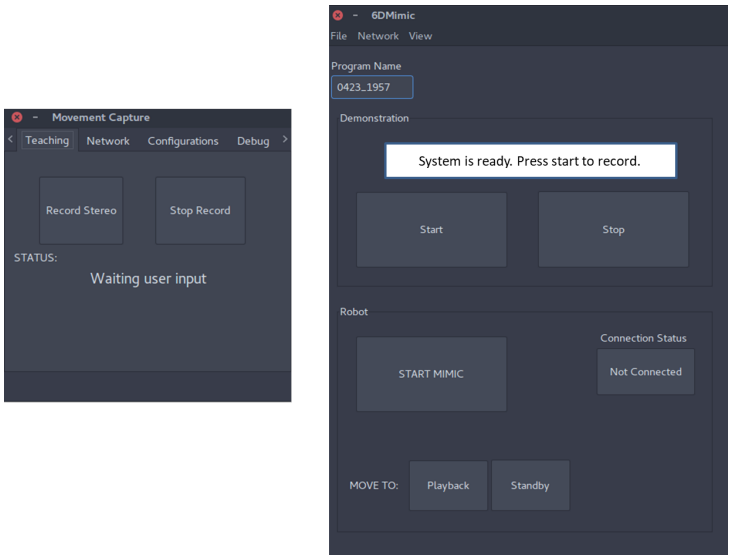


Figure 4.1: 6DMimic interfaces: (left) Teaching mode. (right) Complete mode

4.1. ABB complete mimic procedure

In this tutorial it will be used the ABB robot and the complete 6DMimic procedure. The complete procedure instructions is presented bellow. A support Figure 4.2 is also presented.

1. Start the robot's program like presented in Section 2.4.1.1.
2. In the PC, start the 6DMimic shortcut button in the desktop (or run the HMI_RUN script 2.5.4).
3. On the main window, wait for a few seconds and make sure the connection status is 'Connected' (A green box should appear);
4. Set the program name;
5. To perform a demonstration the user just needs to press the "Start button";
6. With the manual tool in hand (attached with the marker) perform the movement in front of the stereo cameras. Be aware of do not occlude the marker from the camera's view and do not move the tool in a manner that exceeds the robot workspace (even thought pressing the start/stop button, otherwise, if the cameras get the marker, the robot will try to mimic any movement);
7. The automatic table can turn the object facilitating the process. For this, just press the pedal under the table and the SincroBox will sync the signals to be, latter used by the playback process.
8. At the demonstration end, just press "Stop";
9. Start the mimic procedure by clicking in "Start Mimic" button;

The "Playback" button allows the user to set the robot to the initial position while the "Standby" set the robot to zero location. A complete demonstration of the procedure is showed in Figure 4.2.

OBS: The connection button does **NOT** need to be GREEN to upload the code to the robot but is is necessary to control the robot remotely.

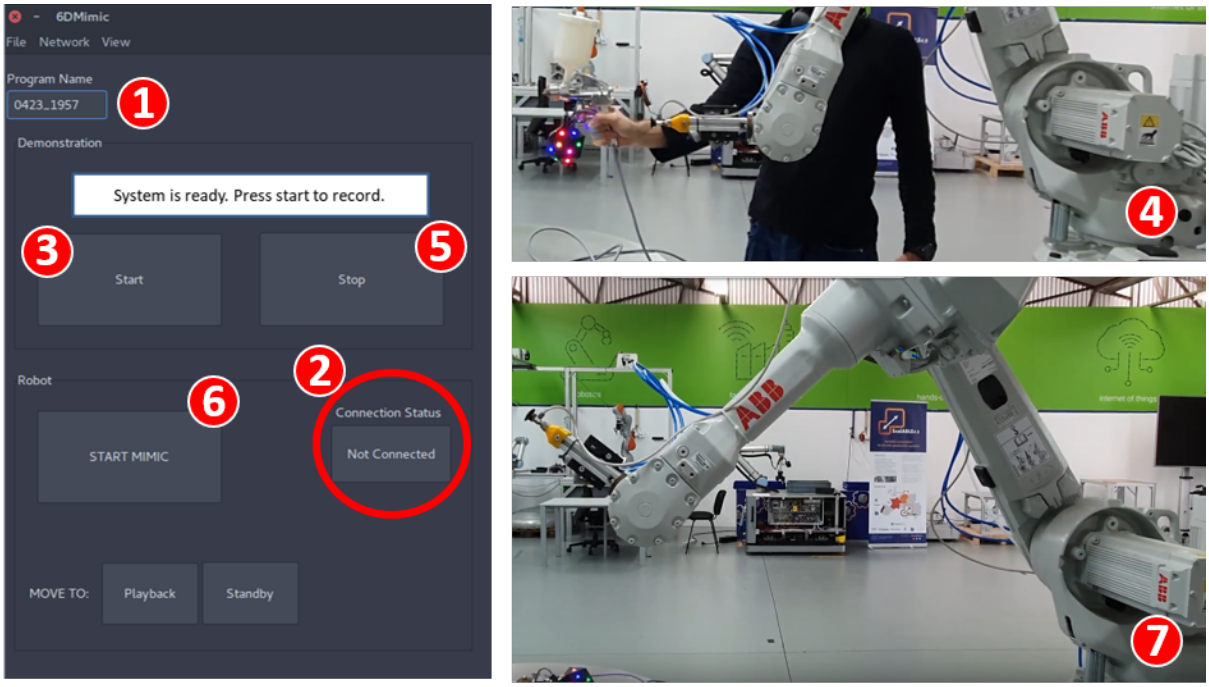


Figure 4.2: 6DMimic usage. (1) Set program´s name; (2) Check robot's communication; (3) Stay in the cameras FOV and start demonstration in robot's workspace; (4) Demonstrate the mission; (5) Stop demonstation; (6) Start mimic procedure; (7) Robot will perform the mimic procedure.

5. IMU integration

5.1. Proposed software architecture

In order to introduce the IMU into the 6dmimic system, we propose the architecture in Figure 5.1, adding two new modules to the system:

- IMU module - Collects the IMU data during the Demonstration process and exports it to a file. Timestamps for the IMU readings are generated from the CPU's internal clock, making it simple to synchronize with the cameras' output;
- Occlusion module - Examines the output from the Icosahedron3d Module (marker pose based on the stereovision system) in order to detect the presence of occlusions by comparing the differences between the timestamps of sucessive poses;
- IMU Estimation module - Estimates the pose of the marker during the time period in which an occlusion took place.

The IMU module will run in parallel with the camera recording modules. The other two modules will enter the system pipeline between the *Icosahedron3d* and *Segmentation* modules.

The IMU Estimation module will have three inputs: The marker poses given by *Icosahedron3d*, the occlusion time interval detected by the Occlusion module and the IMU data file given by the IMU module. After calculating the position and orientation of the marker during the occlusion, the result is added to the original marker pose file with the corresponding timestamps. This output is then used as the input file for the next module in the pipeline, *Segmentation*, allowing us to use the remainder of the 6dmimic system without any modifications.

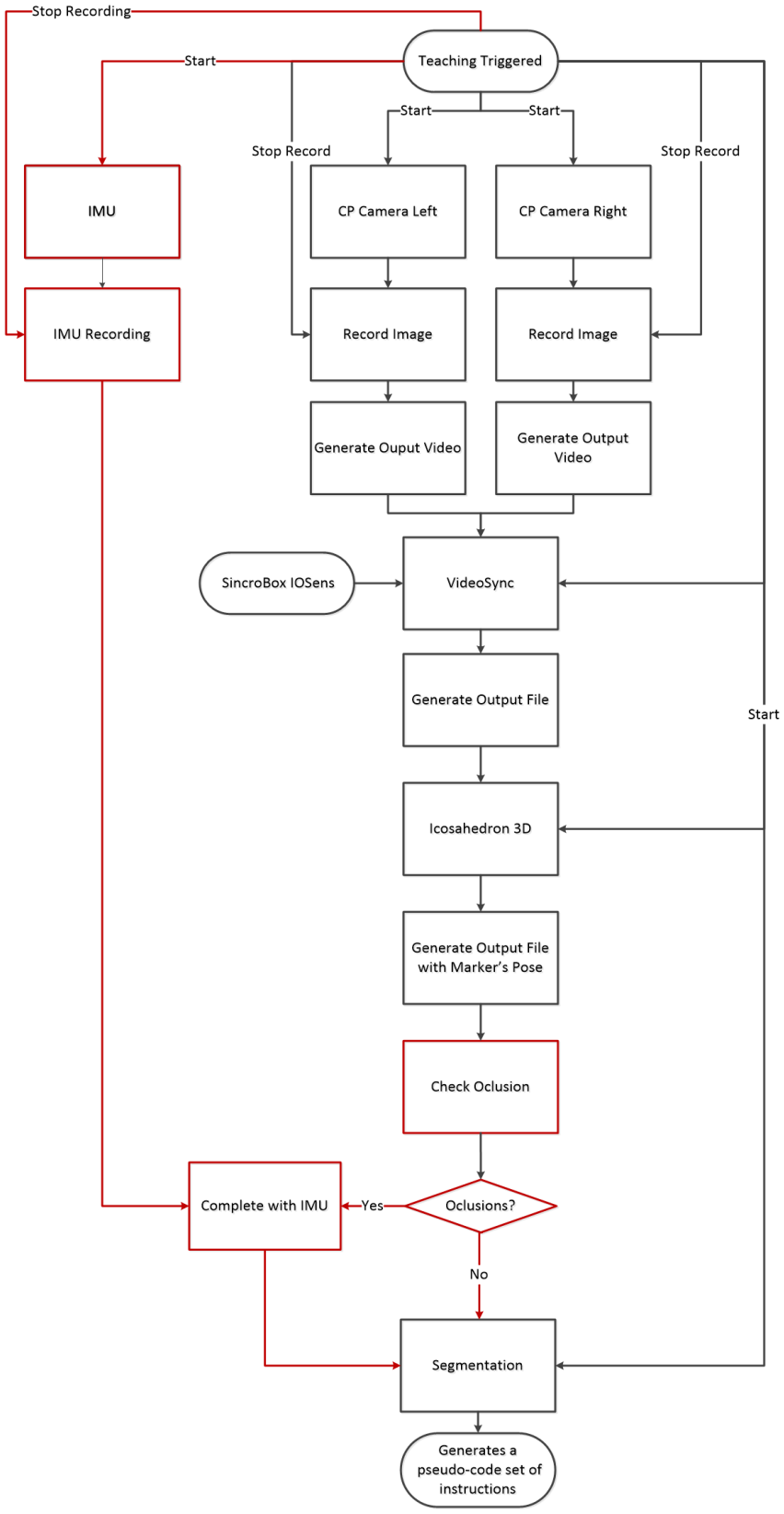


Figure 5.1: The new 6DMimic architecture.

References

Acknowledgements

Appendix

System connection diagram

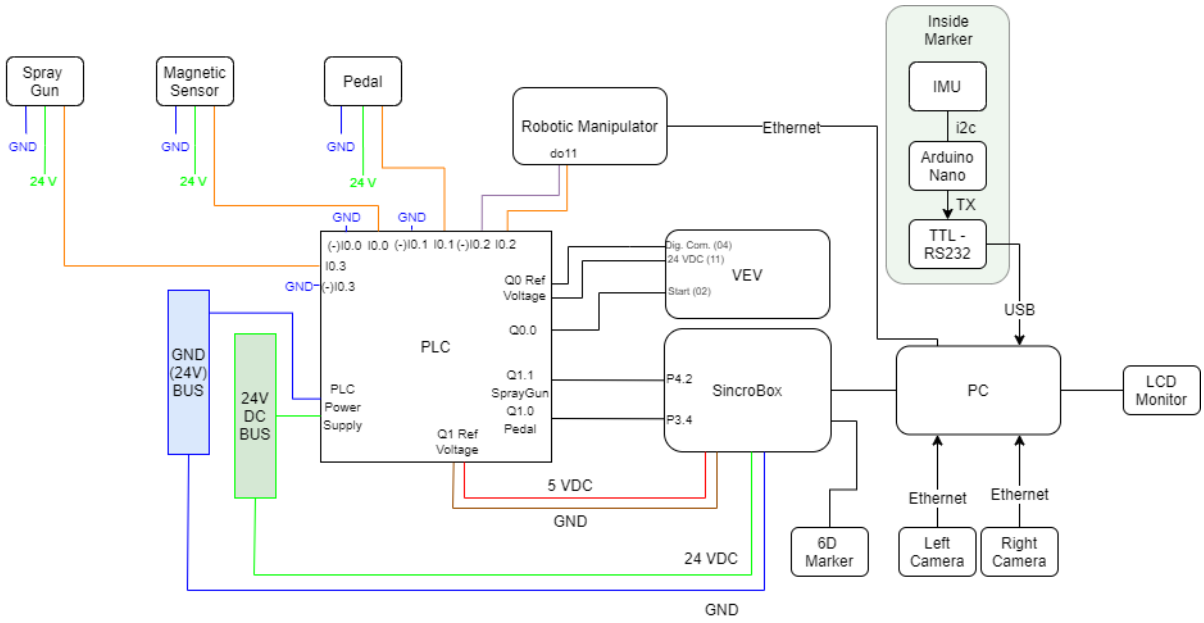
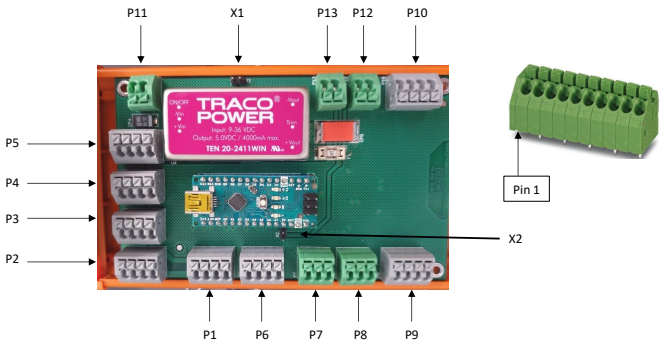


Figure 5.2

Pinout Diagram



Group P1 (General Purpose)	
Pin No	Descr.
1	+24VDC
2	GND (024VDC)
3	+5VDC
4	Q1.0

Groups P2 … P5 (Cameras' Trigger)	
Pin No	Descr.
1	+24VDC
2	GND (024VDC)
3	+5VDC
4	Trigger (Q2.0 … Q5.0)

Group P6 (Marker)	
Pin No	Descr.
1	+5VDC
2	Q6.2
3	+5VDC
4	Q6.4

Groups P7, P8 (DO - Isolated)	
Pin No	Descr.
1	Q7.1, Q8.1
2	Q7.2, Q8.2
3	Q7_Vcc, Q8_Vcc

Groups P9, P10 (DI - Isolated)	
Pin No	Descr.
1	I9.1, I10.1
2	Ix.COM
3	I9.2, I10.2
4	Ix.COM

Group P11 (Power In)	
Pin No	Descr.
1	+24VDC
2	GND (024VDC)

Group P12, P13 (Power Out)	
Pin No	Descr.
1	+5VDC
2	GND (05VDC)

X1: Insert Jumper to connect different ground signals (024VDC to 05VDC)
X2: Insert Jumper to power Arduino from +5VDC PSU

Figure 5.3