industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Development and Automation of a Robotic Welding Cell Using Machine Vision in Halcon Programming Environment

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: José Manuel Pastor Alcaraz
Director: Juan Suardíaz Muro

Cartagena, Julio 2015

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Development and Automation of a Robotic Welding Cell Using Machine Vision in Halcon Programming Environment

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **José Manuel Pastor Alcaraz**
Director: Juan Suardíaz Muro

Cartagena, Julio 2015

Acknowledgements

There are an enormous number of people who I would like to express my gratitude to.
Firstly, I would like to pay special tribute to the research group ACRO for their invaluable assistance and support during this project. It is a pleasure to express my thanks to Geert Leen (my internship coordinator), Nico Bartholomevis and Maarten Verheyen who have guided me in Robotics and Machine Vision respectively. Also, the rest of the team, which have been there for me every time I needed: Roel Conings, Veronique Theunis, Eric Demeester and Geert Moonen. I want to thank all of them for their kind assistance during the developing of this project. I owe an immense debt of gratitude to them for having considered me as one more of the staff during my internship.

Secondly, my regards go to Greet Raymaekers (my Erasmus coordinator) and Juan Suardíaz Muro (my internship coordinator in Spain) for their invaluable information, advice and consideration in all the referent about my internship in KHLim.

I am also indebted to several people for giving me encouragement to develop this project. Firstly I would like to express my profound gratitude to Irene, my girlfriend, and my family for provided me the necessarily strength the moments I struggled. Finally, to all the Erasmus students who studied with me in Hasselt and have made with their company one of the most fulfilled experiences in my life.

Index

Chapter 1

1. Abstract and Objectives

The current Project is developed in ACRO*, Automatisering Centrum Research en Opleiding*. ACRO is a Research and project Group in the field of automation, it offers a complete package of trainings and services in automation.

The project consists in the upgrade of a robotic welding cell into a complete automated application through the implementation of a visual recognition system. In order to achieve this big objective the total project have been segmented into three different task:

1. The installation and functionality of the robotic welding cell without machine vision.
2. Introduction, development and achievement of a vision solution that provides the position and orientation information of the recognised pieces to the industrial robot.
3. Encapsulation of the vision solution deployed into a visual basic environment to offer a friendly interface to the different users and operators.

Following the technology used in the project it can be encompassed into three different systems (they will be extensively described in section 3 of this paper):

- Robotic System.
- Welding System.
- Vision System.

The final objective piece to recognise and weld is a metal cylinder that will be fixed into a flat square piece. This piece has been selected attending to its welding and visual recognition challenges, which can represent an acceptable example of the potential of the final welding cell once the solution is properly developed.

Actually, the current project isn´t an isolated development carried out by ACRO, it is also inside a bigger industrial project developed by different partners and it has the company Sirris as a main contractor.

Sirris is the collective centre of the Belgian technological industry. They help companies in the implementation of technological innovations, enabling them to strengthen their competitive position over the long-term. Their employees visit companies on site, offer them technological advice, launch innovation paths, and provide guidance until they reach the implementation phase. It is their aim to find concrete solutions to the real challenges facing Belgian entrepreneurs.

The project is called *“Smart Factories. Towards the Factory of the Future”*. It began in 2012 and it will finish in May of 2016. The goal of the project is support the manufacturing industry in Flanders by the development of intelligent factories increasing substantially the manufacturing production. The result is create a flexible production system able to produce small series with productivity in order to response to the current market trends.

A list of concrete steps have been defined in order to achieve the purpose of the project. There are a total of seven technological phases:

1. Zero ramp-up: production of small test series or trial products to check that the specifications of the project are satisfied.
2. Safe human-robot interaction: safe human-robot work in order to the production remain accessible for operators.
3. Auto programming: challenge of achieve the automated programming of the robot according with the information captured by the vision system.
4. Intelligent automated quality control: integration and automation of quality control where the series are controlled 100 per cent.
5. Offline robot programming: development of the required software to ensure complex robot can be programmed remotely.
6. Remote monitoring production: generation of feed-back in order to achieve real-time monitoring.
7. To stand-alone to network manufacturing cells: cells created in the project doesn't work as isolated islands there are communication with each other and with a Smart Factory.

In that way as a final objective once the project is finished, we are focus on the achievement of a real robotic welding cell that presents small, flexible and functional characteristics for companies that does not have the necessarily incomes to invest in the expensive robotic welding solution already implemented in the market.

Chapter 2.

2. Introduction

The situation of the project is outlined in this section of the paper.

Furthermore, in the following lines is outlined a brief description of the concepts of robotic welding, the systems implemented in it and the acquirement technology applied to upgrade the robotic welding cells to a complete automatic tools.

Also, the state of art of robotic welding nowadays in the industrial environment is properly explained. Finally, we will define the solution adopted in our project and how it is contextualized in the sector of Robotic Welding Industry.

2.1. Situation of the Project

This section outline a briefly description of the situation of the project. It is developed in the ACRO Centre of the KHLim university .The laboratory (ACRO) and the university (KHLim) are located in Diepenbeek, a little village situated in the Flemish region of Belgium, almost in the frontier with Netherlands. Diepenbeek belongs to Hasselt the capital of the Flemish province of Limburg.

2.1.1. Science and Technology in Belgium

Many discoveries and new technologies that radically changed our everyday lives and how we view the world originated in the work done by Belgian scientists: Zénobe Gramme's dynamo, Ernest Solvay's soda, Leo Baekeland's Bakelite, Georges Lemaitre's Big Bang theory and Edouard Van Beneden's research into cell division are an example of this. Their work led to significant advances in science and technology.

Belgian research has focused on such areas as medicine, biochemistry, statistics and astronomy. Belgian researchers have received prestigious international scientific prizes for their work in these areas.

Belgium is active in space research, North Sea research and climate research in Antarctica. It is also involved in ambitious research projects focusing on sustainable development. Many Belgian scientists are currently at leading universities around the world. Conversely, the high scientific standard of Belgian universities and the quality of life in Belgium attract many foreign students and researchers to scientific institutions.

In Belgium, science has traditionally been linked to education and the public sector. But for some time now, industry has also been heavily involved in scientific research, focusing on space travel, biochemistry, medicine, pharmaceuticals and IT. Belgium is a federal state. As such, it is mainly the communities and regions that are responsible for scientific research, although the federal level does have some powers in this domain too.

2.1.2 University KHLim

KHLIM (Katholieke Hogeschool Limburg) or its official English equivalent, Limburg Catholic University College, is a university college with campuses in Hasselt, Diepenbeek and Genk. It offers a vast range of bachelor degree courses and also advance bachelor's programs, postgraduate certificates and other continuing studies. As an institution of knowledge, KHLim also takes part in research and provides community.

Some important information about the university are showed as follows:

- Founded 1994.
- 6 Departments: Department of Commercial Science and Business Administration, Teacher Training, Industrial Sciences and Technology, Health Care, Socio-Educational Care Work and Media and Arts & Design Faculty.
- Number of Students: 6750
- Number of staff: 800
- Headquarters: Zuivelmarkt, City centre of Hasselt
- Campuses in Hasselt, Diepenbeek and Genk.
- Member of KU Leuven Association (close cooperation between Leuven University and 12 Flemish university colleges).

Fig. 1.-KHLim symbol

2.1.2.1. ACRO Centre

The current Project is developed in ACRO (Automatisering Centrum Research en Opleiding). ACRO is a Research and project Group in the field of automation, it offers a complete package of trainings and services in automation.

The topics of the research group ACRO are:

- Industrial real-time networks (fieldbus)
- Real-time vision applications
- Sensor based robotics

- Real-time camera hardware
- Real-time operating systems

Furthermore, ACRO is a PROFIBUS and PROFINET COMPETENCE CENTRE. It is focused on in training activities, but also it takes active part in support, advice and technical activities in issues related with this kind of communication technology.

This centre is gathered in the research and innovation group of KHLim University "KHLim Quality Quadri".

Fig.- 2. ACRO symbol.

2.1.2.2. KHLim Quadri

KHLim Quadri forms a group of cells where the research and investigation can be delivered together taking part internal services of KHLim and private companies as external partners.

Quadri participates in researches based in learning and education. In this way, students can learn by themselves carrying out the researches, this group of laboratories and centre are driven by the idea of quality (Quality Driven= Quadri). Some of the centre gathered in this research association are presented as follows:

- Education & ICT
- I-Net
- ACRO
- Kunststoffen
- Social Spaces
- Onderwijs en zorg
- Art. Object & Design
- Lab4U

Fig. 3.-KHLim Quadri Symbol.

2.1.2.3 Location

ACRO's facilities are located in Diepenbeek, in the University Campus. In this space, three universities (PXL, UHasselt and KHLim), some private companies and many research and innovation centres coexist in order to provide a perfect environment for students, professors and researchers.

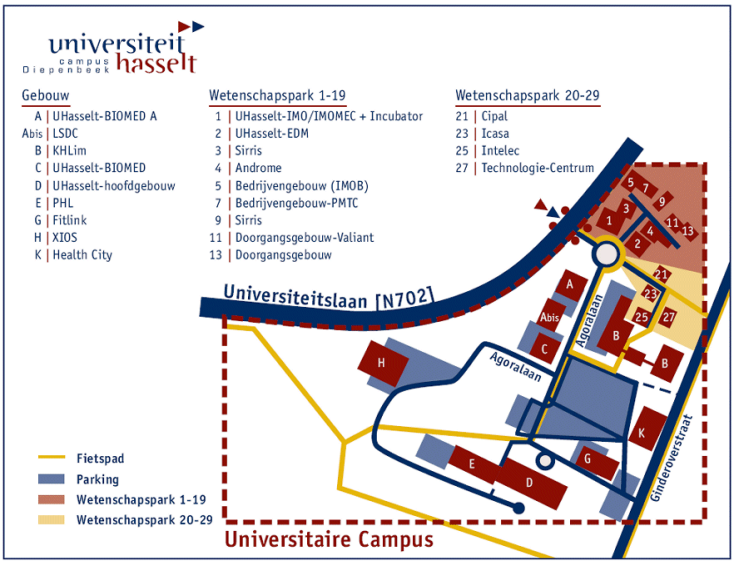


Fig. 4.-Map of Universitaire Campus.

ACRO centre is situated in "Technologie-Centrum" building. It is divided in many laboratories of different research and investigation aspects. A part of ACRO, another investigation centres allocated in building are: Kunststoffen (plastic and composites centre), NuTEc (Nuclear Technological Centre) and EMAP Elektromecanica.

Fig. 5.-Technologiecentrum building

Fig. 6.- ACRO's facilities.

2.2. Robotic Welding

Modern manufacturing faces two main challenges: more quality at lower prices and the need to improve productivity. Those are the requirements to keep manufacturing plants in developed countries, facing competition from the low salary regions of the world. Other very important characteristics of the manufacturing systems are flexibility and agility of the manufacturing process, since companies need to respond to a very dynamic market with products, exhibiting very short life-cycles due to fashion tendencies and worldwide competition. Consequently, manufacturing companies need to respond to market requirements efficiently, keeping their products competitive. This requires a very efficient and controlled manufacturing process, where focus is on automation, computers and software. The final objective is to achieve semi-autonomous systems, for example, highly automated systems that work requiring only minor operator intervention.

Industrial robots are essential components of today's factory and even more of the factory of the future because the flexibility and adaptability achieved and the involve adding of intelligent machines that can perform repetitive task in a high ratio quality- price. The most active industry

in the application of robots is the automobile industry and there is great interest in applying robots to weld and assembly operations, and material handling.

For the sake of competitiveness in modern industries, manual welding must be limited to shorter periods of time because of the required setup time, operator discomfort, safety considerations and cost. Thus, robotic welding is critical to welding automation in many industries. It is estimated as much as 25% of all industrial robots are being used for welding tasks.

2.2.1. Parts of a robotic welding cell

Although, there are many kinds of robotics welding applications all share the same basic structure with a predefined sub-system. Thus, an automated welding cell based on an industrial robot is formed by the following subsystem:

- Welding sub-system: it the sub-system that governs the welding action, in it formed by the power source, wire-feeder, welding torch, cooling unit and the filler wire metal. It makes possible to customize all the influential parameters such as welding voltage, welding speed, arc length correction, etc.
- Robot Controller: this sub-system control all the geometrical parameters of the cell, ensuring the correct orientation and movement of the welding torch and also the correct moving through the welding path.
- Industrial Robot: make the movements commanded by the robot controller real. It has to be strong and robust in order to endure in the un-friendly environment in welding applications.
- Sensor System: it measured the relative position of the welding torch and the work-piece and it send this information to the robot controller or a remote controller.
- Remote PC: it interprets the data information sent by the recognition system and in conjunction with the robot controller makes the movement decision in order to maintain the correct orientation and path of the welding gun.
- Communication System: it ensures the correct flow of information over all the different sub-system, it can be combined by different technologies such as Profibus, Serial Communication, Ethernet, etc.

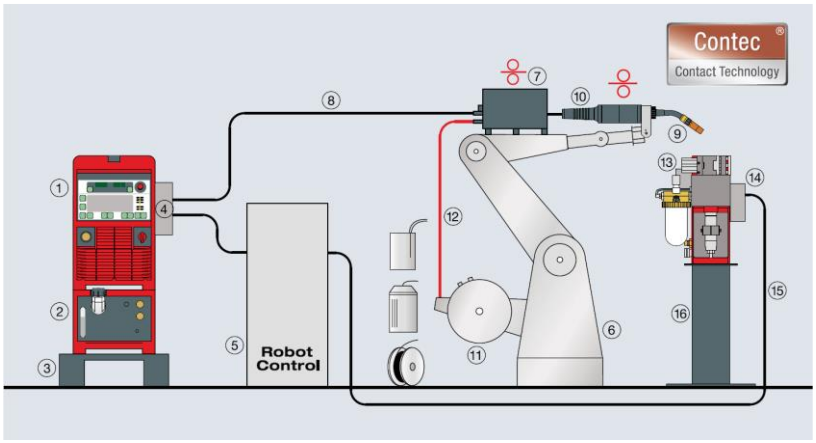


Fig. 7. - Basic Scheme of Robotic Welding Cell without recognition system

2.2.2. Robotic Welding Classification

Robotic welding applications can be classified according of the temporal development that they have had themselves. Thus, we can difference between three generations:

1. First Generation: welding system was a two-pass weld system, where the first pass is dedicated to learning the seam geometry followed by the actual tracking and welding in the second pass.
2. Second Generation: The second generation of welding systems, on the other hand, track the seam in real-time, performing simultaneously the learning and the seam tracking phases.
3. Third Generation: The third generation of welding systems not only operates in real-time but also learns the rapid changing in seam geometries while operating within unstructured environments.

The flexibility was achieved in The Third of welding systems but at the expenses of a considerable amount of programming work of high skilled people in system's integration directed to specific applications. However, Modern manufacturing industries have added agility and availability inside their key issues. Thus, it is possible to reveal a high grade of flexibility to the normal operator without the need of extra skills from him.

Fig 8. - Third Generation Welding System Application

2.2.3. Reason of Robotic Welding

Nowadays, robot manipulators include the following features:

1. Programmable control system, using powerful programming languages and environments.
2. It is possible to define positions/orientations, define reference systems, parameterize trajectories and other actions, and play that continuously with high precision and repeatability.
3. Advanced PLC capabilities are also available, namely, IO control and data acquisition, and several communication interfaces and protocols. These functionalities enable robots to coordinate actions with other equipment and sensors, and being integrated with other computers and manufacturing systems existing in the setup.

Repeatability	Up to 0.03 mm (0.1 is common)
Velocity	Up to 5 m/s
Acceleration	Up to 25m/s^2
Payload	From around 2-3 Kg up to 750 Kg
Weight/ Payload	Around 30-40
Axis	6
Communication	Profibus, Can, DeviNet, Ethernet and serial channels (RS232 and RS485).
IO Capabilities	PCL like capabilities to handle digital and analog IO

Fig. 9. - Robot manipulator main characteristics

Since most welding techniques require motion control, sensor integration and coordination with the welding power source (controlled using IO digital and analogue signals, or *fieldbuses*), then robot manipulators are an almost perfect match for the vast majority of welding processes.

Difficulties may also appear in automating the welding process, using robots. In fact, introducing robots means increasing complexity in the manufacturing process, and requires skilled personnel to handle programming and maintenance. That may constitute a major drawback, not allowing companies to take full advantage from the flexibility stored inside the robotic manufacturing machines. This puts focus on human-machine interfaces (HMI) for control, command and supervision, leaving space for the software architecture used to develop the HMI solutions.

Despite all the interest, industrial robotic welding evolved only slightly and is far from being a solved technological process, at least in a general way. The welding process is complex, difficult to parameterize and to monitor and control effectively. In fact, most of the welding techniques are not fully understood, namely the effects on the welding joints, and are used based on empirical models obtained by experience under specific conditions. The effects of the welding process on the welded surfaces are currently not fully known. Welding can in most cases impose extremely high temperatures concentrated in small zones. Physically, that makes the material experience extremely high and localized thermal expansion and contraction cycles, which introduce changes in the materials that may affect its mechanical behaviour along with plastic deformation. Those changes must be well understood in order to minimize the effects.

In conclusion, the majority of industrial welding applications benefit from the introduction of robot manipulators, since most of the deficiencies, attributed to the human factor, are removed with advantages when robots are introduced. Also, the welding process is very dangerous and demanding in precision and operator attention, requiring substantial physical efforts from operators, which makes it a good candidate for robots.

2.2.4. Robotic Welding Sensor

Raising of automation and new advanced materials in welding process have provoked the growing of better control system. This system is necessarily in order to achieve the desired weld with productivity and quality. Because of that, there is a needed to control precisely all the

aspects in the welding process. In doing so, sensors play a crucial role contributing with external information to the control system that manage and control the behaviour of the output of the welding system.

The development of welding technologies makes use of new materials with possibilities to decrease thicknesses. As a result of this new possibilities there is a need to work with tighter tolerances. Thus, new sensor systems have to be able to meet the requirements from the new process and new specifications.

For sensors, its main task to provide the control system with information is not as easy as we think. According to welding process perspective, it is performed by two subsystems: the welding equipment (power source, wire feeder, weld torch…) and the robot. In this way, sensors have to provide with information to both subsystems, Due to this control diversification sensor system is also sub-divided in sensor for technological parameter, measuring welding parameters such as welding voltage, welding current, welding speed, arc length and sensor for geometrical parameters, measuring relative position respect to the TCP of the robot.

In this paper we only outline the group of Sensor for Geometrical Parameters because of the objective of this project is to upgrade a functional robotic welding system into a completely automated welding cell and in the initial system we have an intelligent power source (Fronius PulseSynergic 4000), that provides all the necessarily information about technological parameters.

2.2.4.1. Geometrical Parameter Sensors

Historically the use of sensor for geometrical parameters was not feasible for non-automotive industrial company. The typical prohibiting factors were one or more the followings:

- The high initial investment cost.
- Extended processing times.
- The physical size of the sensor equipment.
- The inability to achieve the desired accuracy process.
- The “stack-up” of tolerances that traditionally have precluded automated welding.

Recent and ongoing advances in sensor technology are eliminating this prohibiting factors and moreover, the technology has become performance and cost competitive for general industrial organizations.

Inside the welding process there are three challenging phases where the use of sensor has ensured the use of robotic automation in the welding of inconsistent joints:

1. Joint Edge Detection-finding the edge or start of a weld seam.
2. Joint Seam Tracking- maintaining the desired welding path.

-Joint Edge Detection.

The tools nowadays available to the manufacturing industry are Tactile Sensing and Proximity Sensor.

Tactile sensing solution is well developed and established. The principle it is quite easy, it is a voltage applied to the gas nozzle, welding wire or independent probe, also this small circuit is connected to an input of the robot controller, ensuring a high value in the input when the

voltage is short out as it makes contact with the work-piece and then the controller makes the necessary programming decisions. The voltage differ from as low of 42Vdc to the maximum specified in the regulation of the country or region. With three simple touches the controller has enough information to situated the work-piece and the star of the weld seam in the TCP of the robot: one per axis (X, Y and Z)

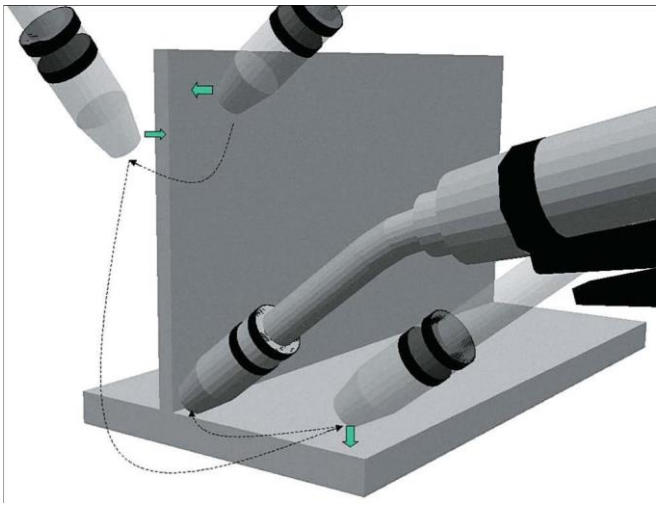


Fig. 10. - Voltage Tactile Sensing System

Proximity sensors have the same principle as tactile sensors but only there is no contact between the sensor and the work-piece and also it provides with an interrupt to the robot controller to make decisions. It operates through analogue inductive field. The advantage of this sensor system lies in the innovative skills of the robot programmer. A well achieve strategic route over the work-piece will provide reduction of sensing time and more reliable information data. It is attached adjacent to the welding torch in order to not compromise the desired welding torch angles in operation. As well as the tactile sensing with different readings from the sensor and logical instructions it is possible to determine the position of the work-piece in the robot TCP.

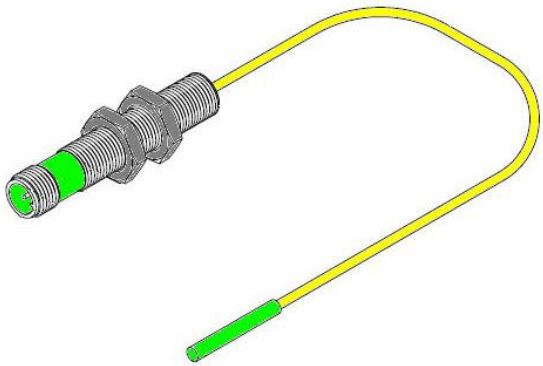


Fig. 11. - Proximity sensor

In conclusion, the detection of the start of the seam can be faced as another recognition vision application and all the success solutions achieved in this processes can be mounted in it.

-Joint Seam Tracking

A pre-programmed path cannot be obtained with the desired precision, since deviations from the programmed path are likely due to deficient path definition, but also due to material plate deficiencies and to the effect of heating the plates. Consequently, an on-line joint recognition and seam tracking system must be available. Several techniques have been used for joint detection and seam tracking, namely for welding robotic systems.

Control the start of edge of the joint may be relatively simple within the adequate tool. The seam, instead, it is a quite more difficult, it may vary due to manufacturing methodology or thermal influences during the process. However, there are well developed solutions that are able to track in real-time the weld seam ensuring the desired weld results. The most popular solutions of seam tracking are vision, laser scanning and through-the-arc.

Vision system are equipped with special cameras compatible with the bright, light and environment in welding applications. Software with 2-D guidance calculates the relationship between the camera and robot location. When the picture is taken in space, the robot knows exactly where that feature is relative to itself, and it can detect any deviation from the program originally taught. Consider a muffler application. Because the muffler's shape, elliptical halves assembled with end caps, changes ever so slightly from part to part, so does the weld path. A vision system can see the part and gives the robot the information it needs to adjust the weld points accordingly.

The cameras produce 2-D gray scale images, so everything is a shade of gray. Dark or shiny surfaces can present problems. Some cameras have an auto-exposure feature that gathers more or less light depending on ambient conditions. Others can take multiple pictures to build up an accurate part profile from varying exposures. Thin stainless steel lap joints, with highly reflective surfaces and very small shadows, can require a system like this. However sometimes filters may need to be added to ensure the camera sees certain part features.

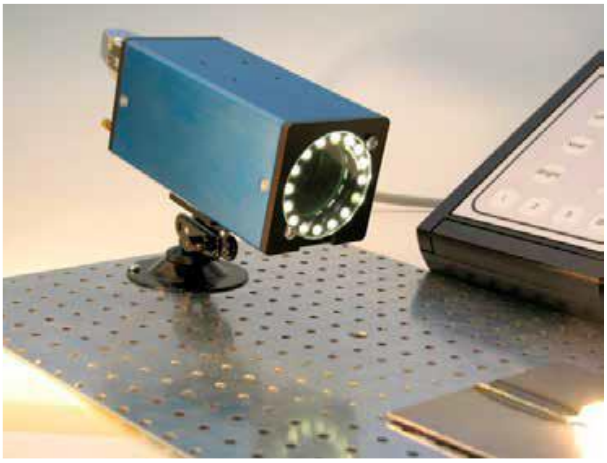


Fig. 12. - Welding Camera

Technology of laser scanning in gathered inside the categories of 1D, 2D, 3D, spot, line and circle. Although there are multiple uses of the various kinds of laser scanning options, there are also some commonalities among them. Normally, this laser systems are combined with some kind of CDD style camera in order to capture images for processing with algorithms. The system is managed by a remote PC that makes the operations to give the correct data to the robot depending on the inputs given by the laser system. The most recent vision system are mounted

the PC and the camera together in order to make a smart space and reduce space. To measure the distance, the method of triangulation is used which is of great importance in welding.

A laser beam is focused on an object, and then the reflection from the object as seen from a lens in the laser sensor is determined by the distance between the sensor and the object. If the object is close to the sensor then the angle between the outgoing beam and the reflection through the focusing lens of the detector is large, while it is small if the object is farther away. The detection of the distance between the sensor and object is made by focusing the incoming beam on a detector, in most cases a CCD array. Depending on which of the pixels of the array are illuminated, it is possible to calculate the distance to the object.

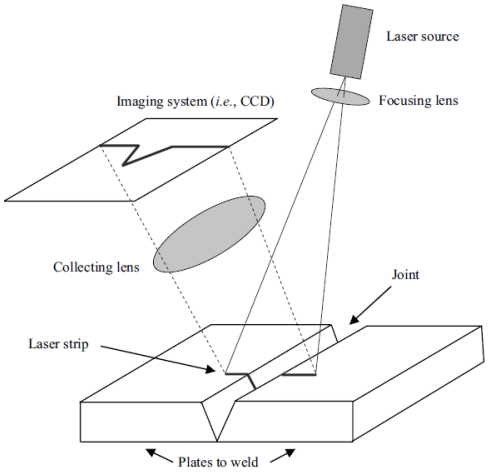


Fig. 14. - Working description of triangulation method

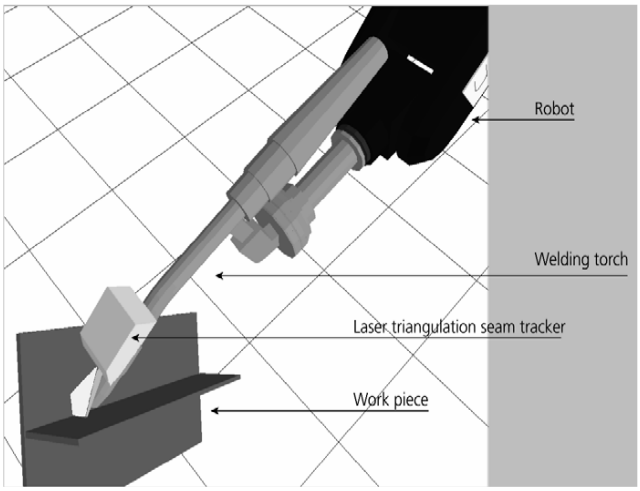


Fig. 13 - Typically Laser Scanning Solution

Seam tracking using a weaving motion and the arc itself as the sensor, sometimes referred to as through-arc sensing, it was introduced in the 1980s. The principle behind the method is to make use of the change in current when the distance between the contact tube and the work-piece varies. The underlying principle is relatively easy and cost-effective to use and is a common sensor for tracking methods in robotic welding based on gas metal arc welding and related processes. The approximate relationship between arc voltage (U), arc current (I) and the contact tube to work-piece distance (l) is expressed by:

$$U = \beta_1 I + \beta_2 + \frac{\beta_3}{I} + \beta_4 I \quad (1)$$

Where the constants β_1- β_4 are dependent on factors like wire, gas and the characteristics of the welding power source. In most cases, the welding power source is set up to maintain a constant voltage and thereby a more stable welding process.

The operating principle for through-the-arc sensing is to monitor the arc in order to track the seam. Through-the-arc sensing primarily uses “impedance” (not just simple volts or amps), which provides a more reliable value within the arc physics to determine accurate data. The rules of the chosen or programmed weave data will determine how the robot reacts to the changing impedance. The changing impedance is used to modify in real time the wire feed speed (amps and fill ratio) and the robot travel speed. This combination of attributes increases greatly the variety of joints and materials that can utilize the system.

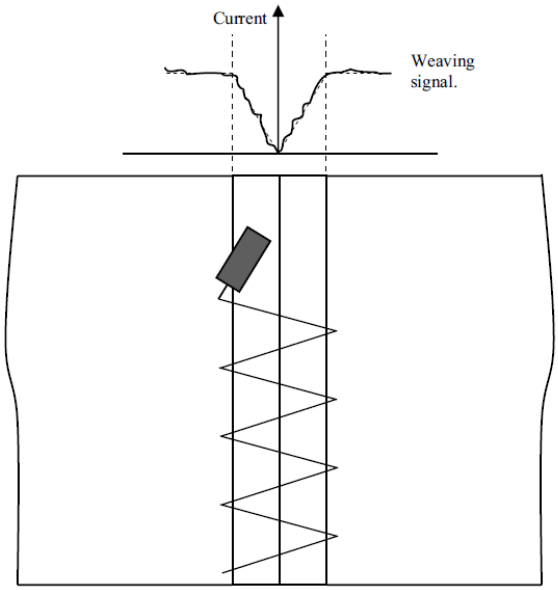


Fig. 15. - Through-the-arc operation

2.3. State of Art

Robotic welding has had to deal since its beginning with the relevant technical and scientific aspect involved in the reproduction task of the experienced and skilled work of the human welder. Because welding tasks have been performed by only humans for a long time, automation welding is challenging to achieve the desired goals. This systems have to present a very flexible and adaptive behaviour.

Almost any welding application is constituted by three well defined phases:

1. Preparation Phase: in this phase the operator sets up the different parts of the welding cell such us: power source, robot and robot program. Also, in this phase the settings of the welding applications are selected (welding current, welding speed, welding current, type of gas, etc.). If in the application there is some kind of offline programming or any CAD/CAM information, they have to be developed and implemented in the total system in this phase.
2. Welding phase: it corresponds with the pure welding action carried out by the robot in conjunction with the welding system. In this stage, the system has to be able to maintain the settings defined in the first phase, in order to achieve that, some sensor and recognition systems are implemented in the cell giving information in real-time to the robot controller.
3. Analysis phase: normally this has been a post-welding phase, but in the new automation welding systems it is possible to examine the obtained welds even during the welding action thanks to the innovation in sensor and recognition issues. Some recognition

system such as laser 3D cameras execute on-line setting analysis during the welding phase.

Consequently, when designing a fully automated robotic welding cell all the phases mentioned before must be considered in order to achieve the proper performance and quality results. In this section we outline how this issues has been faced in the manufacturing industry.

2.3.1. Robotic welding fundamentals issues

The following lines detail some of the most relevant issues in robotic welding namely: modelling and control the welding process.

2.3.1.1 Modelling the welding process

Modelling the welding process is a conjunction of a theoretical problem and a technological problem. In order to understand the welding process it is required theoretical studies but also extensive experimentation to obtain proper models. The welding process from the automation point of view, presents a vast range of characteristic and requirements.

The first step to design a welding robotic system is to identify the related parameters in the process, which should be controlled in order to obtain the desired welding quality. The input related in the process can be classified into three different categories:

1. Primary Inputs: variables that can be modified on-line during the welding process such as welding voltage, wire feed rate, torch speed. Some of them have to be changed in the welding power source and another in the robot software. For example the speed of the welding torch can be changed according to the TCP of the robot to obtain coordinated motion whereas welding voltage can be modified in the settings of the power source.
2. Secondary Inputs: variables that can be customized before the beginning of the welding service but no during it. An example of then are thy kind of the shielding gas, flow of gas, torch angle or the characteristic of the wire used.
3. Fixed inputs: parameter that are fixed with parameters of the previous two groups, they cannot be customized because they are determined as an imposition in the moment of the selecting the welding process and the welding current procedure. Parameters of this group include the joint geometry, plate thickness, physical properties of the plate metal, etc.

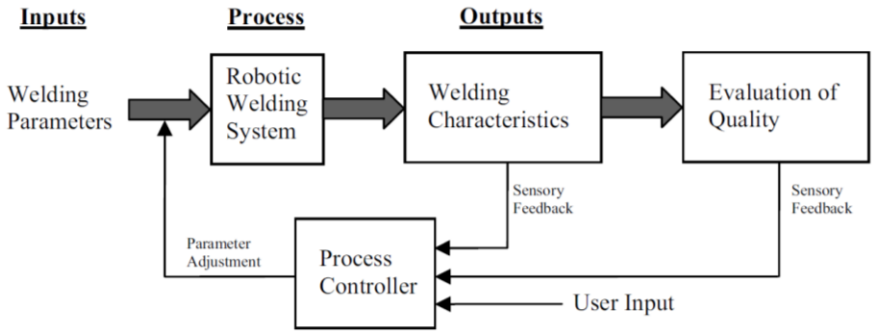


Fig. 16. - Overview of a Robotic Welding Control

All of this parameters must be handled carefully and correctly defined. Also the correct preparation of the setup and the selection of the secondary inputs are fundamental to control the primary inputs efficiently.

Another important set of parameters are the output parameters. Those parameters define and characterize the weld and are used in order to evaluate its quality. In general, there are two type of this output parameters: geometrical and metallurgical.

Geometrical process result from the mass balance originated in welding process and basically defined the way the filler material is transferred to the welding joint. This parameters, in consequence of its nature are used in order to determine the validity of a welding. The basic parameters used depend on the kind of welding process. For V-groove welds, the parameters are: the penetration, the bead width and height and the cross-sectional area. Moreover, for fillet welds the parameters are the penetration and the length of both legs.

The penetration is one of the most important parameters to evaluate the quality of the weld, because its relation with the way the filler metal is combined with the work-piece during the welding process. Unfortunately, there are no sensors, in the market able to measure it on-line. Because of that, it is very difficult to control it during the welding process.

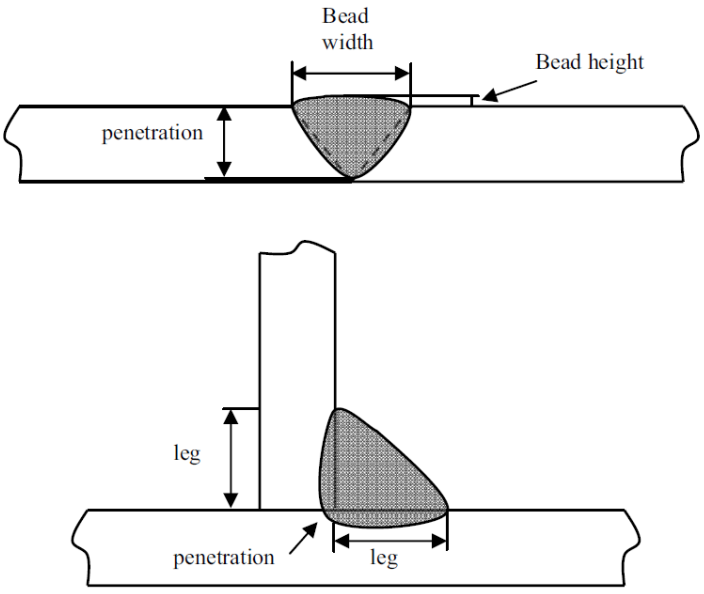


Fig. 17. - V-groove and fillet welds geometrical parameters

In order to control this geometrical parameters during the welding process and in the inspection phase it is crucial that the sensors of the system are correctly defined, installed and calibrated. In chapter 2.2 we have described some type of sensors available to satisfy start beam detection and seam tracking. However, sensor system as a laser scanning or vision can be used also in the post-weld phase. Since sensor systems have been mounted together with powerful APIs for general use, with standard interface for the robot controllers and current computer hardware, they constitute a powerful tool in robotic welding.

Following the previously commented, the penetration is one of the most important parameter in welding applications but also one of the most challenging to monitor. Basically, good welds have constant penetration along the weld path. According to that, the global automated system has to be capable to keep the penetration constant despite possible variations in the joint

geometry. The majority of the methods defined to measure the penetration require back-face bead measurement, requiring access to the back of the work-piece and it is not always possible. There is also another method that ensures obtain an estimation of the penetration and it depends on the front-face bead geometrical characteristics and the weld bead temperatures. Another solution for penetration is use ultrasonic techniques but a full model based in this solution has not been achieved yet.

Obtaining the penetration in conjunction with a correct seam tracking and definition of the start beam it is possible to reach a good understanding of the welding process behaviour according to the geometrical parameters of it.

Metallurgical aspects also play an important role in the welding quality. They determine important mechanical characteristics of the final weld seam such as hardness, soundness, strength and residual stresses. This mechanical parameters are not easy to measure online. However, all of them are results of the heat generated during the welding process. Because of that, in order to achieve proper metallurgical characteristics some aspect of heat have to be guaranteed:

1. A certain peak temperature is needed to achieve a good metal fusion and penetration.
2. A roughly uniform distribution, centred in the weld joint, is required to achieve a constant weld.
3. Acceptable cooling rates, compatible with the required metallurgical characteristic of the final work-piece, are also needed.

All of this requires focus on the need to monitor the thermal events of the welding process, it is added to the other geometrical requires that a complete automated welding cell must satisfy.

2.3.1.2 Welding Process Control

Like the human welder does, the automatic robotic system should adapt to the variant condition continuously, and also, be able to move efficiently the welding torch. Basically in order to achieve this flexibility in automated welding system, there are three issues that have to be correctly satisfied by the robot system:

1. A knowledge base.
2. Sensor and interface.
3. Programmable and flexible control system facility.

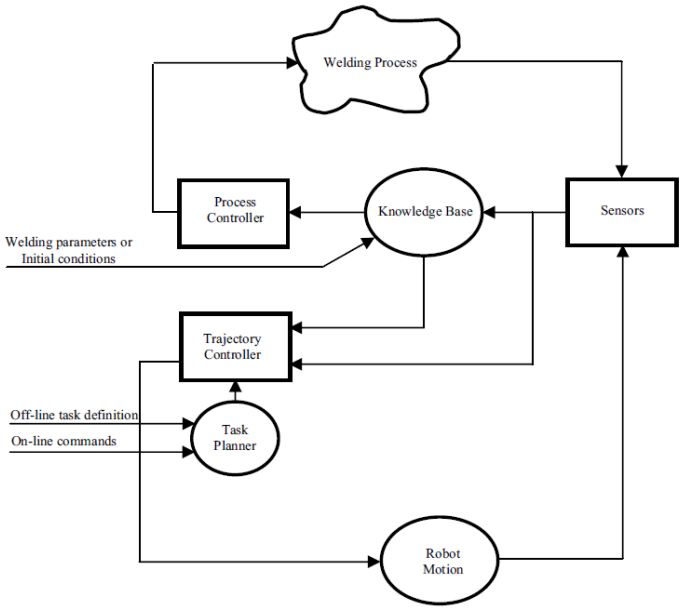


Fig. 18. - Basic Scheme of a robotic welding control system

- Knowledge base

Model the welding process is very difficult, because of that, it is advisable a rule base approach instead of an explicit mathematical model approach. Following this affirmation, the relevant information and setup for a specific welding process have to be export to a knowledge base in order to obtain an adaptive behaviour. This exportation can be done in several ways: fuzzy rule base approaches, neural networks or simple lookup tables for the relevant parameters. Although, there are a lot of model already based in the experienced, in any specific welding process the solution will grow with the experience, adding new with more training.

- Sensors and interface

As we have mentioned many times before, the existence of sensor is fundamental in order to achieve a good solution for a robotic welding system, we are going to focus on the interface and the system architecture of them.

Following the parts of a semi-automatic robotic welding commented in point 2.2, now we upgrade this scheme introducing the sensor system, the communication system and the remote PC that operates all the sensor architecture and monitories the total system.

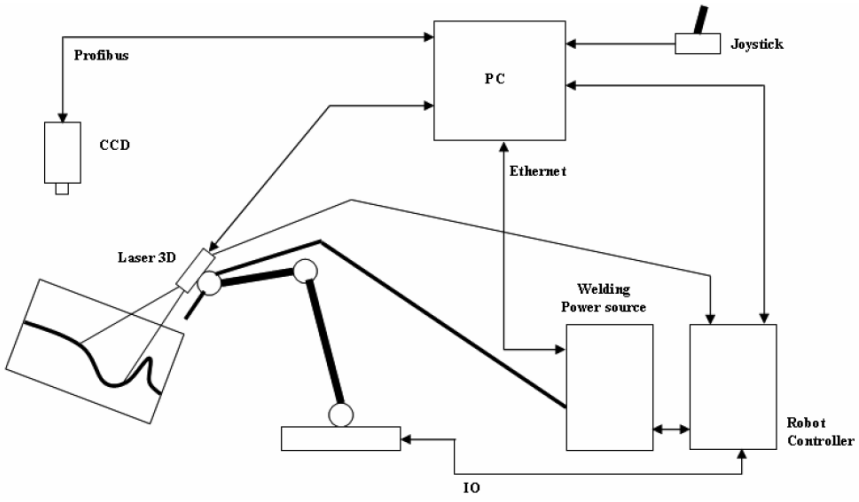


Fig. 19. - Upgraded robotic weld system schematic example

When the sensors are considered in particular, the option of reach intelligent sensors has to be emphasized, for example sensor systems that have a microprocessor in its system and they are able to deal with Remote Procedure Calling (RCP), this sensors can be programmed in order to send specific data at a specific timing. This means that robotic welding systems must be ready to implement distributed software architectures and must be event driven systems, or at least must allow events to be used for obtaining asynchronous information and influence the system behaviour.

- Programmable and Flexible Control Facility

The software architecture used as a developing an important role in the correct definition and function of the total automated system. The challenges posed by a robotic welding system are not different from the ones posed by a typical flexible manufacturing system based on robots. Consequently, the software architecture presented here was designed to be used with general robotic manufacturing cells that may include several types of equipment like robot manipulators, mobile robots, PLCs, CNC machines, vision systems and several types of sensors, *etc.*

Usually, when the cell is complex and it is formed by many sub-system, it is very reasonable that many components use different programming languages. This provoke that sometimes is difficult to adapt new requirements to the project, male adjustment to the cell functionality or introduce new technology in the cell.

Several researchers have been made studies in order to overcome this problem. One idea it to create a software structure compatible with all the sub-system in the cell, creating its own code that can be read and interpreted by any equipment of the cell. Nevertheless, this could limit the potential and flexibility of the system. Consequently, some parameterization is not used, special non-grouped functions are not used and the generated code takes always a uniform structure which may not be the best for all machines.

Another possible solution is quite different, the basic idea is to define for each individual machine a collection of software function that expose all its basic operational features. In order to reach the objective there are some requirements: the local processing capabilities, availability

of communication channels and support for the standard technologies used. However, current robotics systems meet these requirements fully and they do not suppose a serious limitation.

The above-mentioned services are to be offered through, on a distributed software framework based on the client-server model. Several approaches can be used and are currently available from various robot manufacturers, with specific details and implementations. Nevertheless, the following objectives are pursued by any of the above-mentioned software architectures:

1. Represent the robot manipulator motion based on the kinematic and dynamic models, but also based on real-time data coming from the real robot. That can be done using available mathematical and graphical software packages such as MATLAB.
2. Develop applications to explore remotely the entire installation (robot and welding application) using standard programming languages (C, C++, C#, Visual Basic, *etc.*).
3. Integrate and explore intelligent sensors used to obtain information from the process under control.
4. Let users to explore the advanced programming capabilities of actual robot controllers, complemented by extensive libraries of functions, and the optimized manipulation capabilities based on trajectory planning software that takes advantages of optimized kinematic and dynamic models.
5. Enable users to build flexible manufacturing cells, which leads to the possibility to explore the available industrial data network, the possibility to distribute software to the various components of the system, and the capacity to build remote software applications to control and monitor industrial manufacturing cells;
6. Advanced Human Machine Interface (HMI) solutions to operate with industrial systems, hiding from the users all the tricky details about implementation, allowing them to focus on the operational details, *i.e.*, to focus on how systems work and how they can be explored efficiently.
7. Provide ways that could allow developers to focus on the important things about the application they are building: the control algorithm, program functionality and HMI. All the details related with communications, sensor integration, *etc.*, should be hidden from the user.

Taking into consideration these objectives the following programming models are required:

1. Client-server model: there should be server code running on each cell equipment, namely on the robot controllers and coordinating PLCs, that could receive calls from the remote client computers, execute the commands and return the results;
2. Remote Procedure Calls: this is the most usual method used to implement communications between a client and a server on a distributed environment. The client makes a call to a non-local function and the selected RPC mechanism configures the call so that the proper computer, server program and function are addressed, adding the necessary network headers. The server program, running on the server machine, receives the call, executes the selected function and returns the results obtained to the client computer.
3. Data sharing: most of the services require data sharing, files and databases between the client and the server. Consequently, the mechanism provided by the RPC technology to implement data sharing must be used.

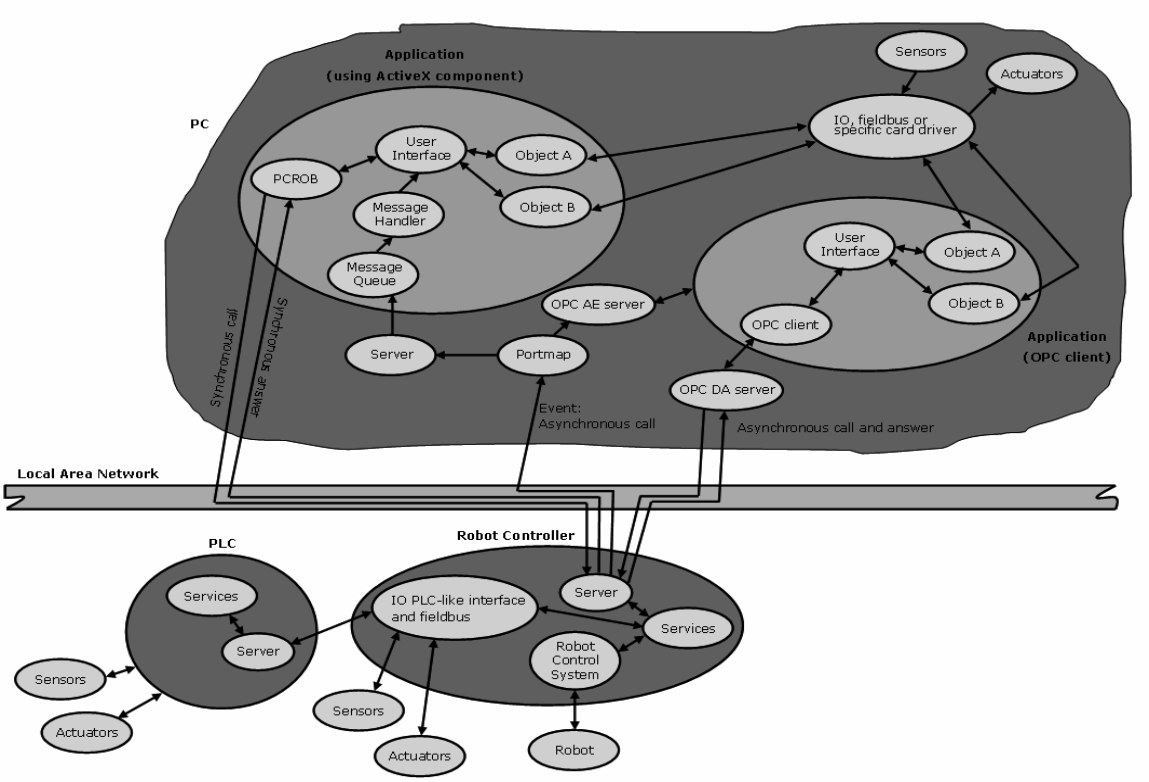


Fig. 20. - Software Architecture used in Robotic Welding

2.3.2. Solutions Implemented in Robot industry

Current industrial robots are controlled by advanced multiprocessors computer system, based on standard parallel buses such as VME, or a serial internal communication system (CAN, DeviceNet, etc.). Normally, this robots use real-time operating system (Vxworks, RTOS) for low levels of interface and more friendly operating system for users interface (Win32 based OS.)

Industrial robots also provide local programming environments based on structured Pascal-form languages (KUKA-KRL, ABB-RAPID) along with a set of libraries that ensures to the operator build and custom applications, interface with other machine and exploring fully the robot and controller facilities.

Furthermore, every software developed in order to operate with this kind of system must comply with actual standards in term of communication protocols, remote interface and software components. The reason of this is to avoid incompatibilities and limit the users to use specific technology because of the excessive dependency originated.

Below this lines we outline two possible solution applied in an industrial robot as a distributed applications, the solution are presented as follows:

1. Using RCP services: a set of function that includes: variable access services, file management services, program management services, IO control, robot controller state services, etc.

2. Based on TCP/IP sockets: function designed in order to operate with a TCP/IP server running as a task on the robot controller. It implements the same robot controller access than the system using RCP.

2.3.2.1 Using Remote Procedures Calls (RCP)

In this solution the architecture client-server was adopted. The robot controller plays the role of server, exposing to the client the collection of RCP services that constitute its basic functionality. RCP offered services such as: files and program management services, robot status and access services. The remote controller (client) in order to access to those services parameterize remote procedures that call to the robot controller through the network.

From all the RCP services available in the robot controller (server), the really needed in order to implement the software architecture described in the previous figure are the variable access services. However, calls to the rest of services are implemented for completeness. The procedure is simple and based on a XDR (Extender Data Representation) file obtained by defining the data structures, furthermore, the service identification and the service syntax specified by the RAP protocol.

2.3.2.2 Using TCP/IP sockets

TCP/IP sockets is one of the most interesting ways to create a network connection between computer systems. This corresponds with a standard client-server procedure, not dependent on the operation system technology used on any of the computer system. This standard only requires the proper definition of a syntax message to be reliable and safe. Also, the user defined messaging protocol must specify the commands and data structures adapted to the practical situation under study.

There are some services that have to be available at the robot controller in order to command and monitor welding application from remote computer. They are implemented by the TCP/IP socket server and presented as follows:

1. Robot and robot program control and supervision services: the remote user has to be able to change the robot operational state (start/stop), selected robot programs (tasks), and read robot controller and robot programs.
2. IO read/write services: the remote client needs to monitor and change the current state of the selected IO signals. Also, some IO analogic/digital signals such as wire speed, welding current are generated by the robot controller. Consequently, the possibility to read/write any signal is needed.
3. Variable read/write services: variable read/write are needed in the case that the remote client has to parameterize the welding action, even the welding trajectory (from a CAD software package), control the welding task and monitor the welding operation.

2.3.3. Solution implemented in our Project

On one hand, following the statements outlined in the previous lines, the solution adopted in our project follows a TCP/IP sockets interface. We have selected a TCP/IP interface because of its flexibility, adaptability and its facility to create a connection between systems.

Although, this first solution implemented in this paper is very simple (the external computer is used only for the recognition of the piece to weld not for online monitoring of the process) in future upgrades of the project this kind of communication procedure will ensure the proper transmission the data between system with the required parameters in safety and speed once real-time communication will be needed in order to monitor and adapt the welding process along the beam.

The construction and customization of the connection between the remote computer and the robot controller will be extensively described in the point 6.4.3 of this paper.

On other hand, the communication between the robot controller and the welding system uses DeviceNet. This communication procedure ensures the total adaptation of both system with its standard conception. This communication and its principal characteristics will be described in section 4.2 of this paper.

2.3.4. Contextualization of the project in the industrial environment and market

Nowadays, the projects available in robotic welding have a high level of technology, investigation and invest. Because of that, this projects are only available for companies in industrial sector with a high levels of billing and investments.

As established in the abstract of this paper, the goal of this project is not improve the achievements of the robotic welding available in the industrial sector nowadays. We have defined the purpose of this project as present a small, flexible and functional tool for companies that does not have the necessarily incomes to invest in the expensive robotic welding solution already implemented.

Although, this project is conceived for small companies as a cheap solution, the desired characteristic of accuracy and robustness are in the highest levels in order to avoid future expenses to improve the system. Furthermore, this project is designed with a modular conception that makes possible to adapt the project to reduced work places.

Chapter 3.

3. Technology Used in This Project

In this section all the different technologies applied and used in the project are outlined in order to provide a global vision of the total system. All the components can be classified into three big groups depending on their field of action:

- Robotic System.
- Welding System.
- Vision System.

Every group is formed in majority by the products of a specific company, in that way, products of KUKA, FRONIUS and HALCON are used in Robotics, Welding and vision system respectively.

Firstly, in the robotic system section, the information about the robot, robot controller and software control are described and also the KUKA application software especially designed for developing arc welding actions.

In a second sub-section the concepts of the welding system provided by FRONIUS and its technical data are briefly described. Also in this section, there is an explanation of the welding torch cleaner cell and the protective gas system available in ACRO's facilities.

Finally, all the concepts of the vision system are outlined, the functionality of the HALCON's software, the programming environment HDevelop and the characteristic of the camera.

3.1. Robotic System

All system's component are provided by the robotic company KUKA, this collaboration has been favoured thanks to the great relationships between ACRO and Belgian branch division of KUKA. Even KUKA Belgium takes an active part in the development of Sirris's project Smart Factories.

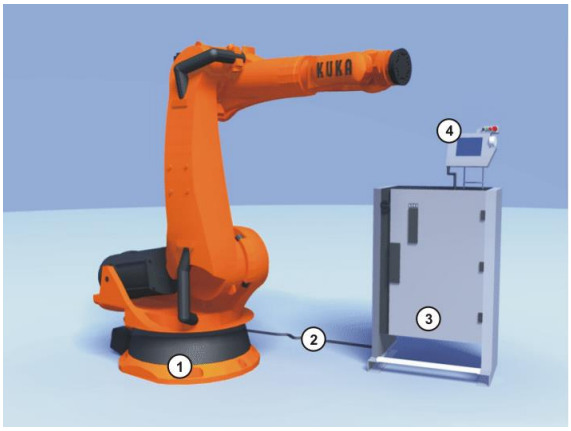


Fig. 21. - Parts of the Robotic System

The robotic system is formed by the following devices:

1. Robot: KUKA KR5 ARC
2. Cable set: Serial Communication
3. Robot controller: KUKA KR C2
4. Tech Pendant: KCP
5. Control Software: KUKA KSS 5.5
6. Arc Welding application software: KUKA.ArcTech software package

3.1.1. Robot

The robot used in the project is the KUKA KR5 arc, it fits perfectly with the desired final results of the project, it presents high levels of accuracy and with its payload of 5 Kg is specially designed for develop arc welding applications.

Furthermore the KUKA KR5 ARC offers the longest live service life in its class thanks to the introduction of low-viscosity lubricants instead of grease. The combination of the six motors with the optimized gear ratios and a weight of only 127 Kg give the KR5 arc a very high dynamic performance and this greater production capacity.

A further plus point is its machine data, which are optimized for continuous-path applications, this machine data in conjunction with a robust mechanical design and high precision gear units the KUKA KR5 arc is able to reach results in a repeatability of less than ±0.04 mm.

Fig. 22. - ACRO's KUKA KR5 ARC

3.1.1.1 Robot technical Data

Type	KR 5 arc
Number of Axes	6 axes
Volume of working envelope	8.4 m^3
Pose Repeatability	±0.04 mm
working envelope reference point	Intersection of axes 4 and 5
Weight	approx. 127 Kg
Protection classification of the robot	IP 54
Sound Level	<75 dB(A)
Mounting position	Floor, ceiling
Surface finish, paintwork	Base frame: black Moving parts: orange

Fig. 23. - Basic Robot Data

Operation	+10 °C to +55 °C (283 K to 328 K)
Operation with Safe RDC	+10 °C to +50 °C (283 K to 323 K)
Storage and transportation	-40 °C to +60 °C (233 K to 333 K)
Start-up	+10 °C to +15 °C (283 K to 288 K)
Humidity rating	Humidity class EN 60204/4.4.4 F

Fig. 24. - Temperature Rate

Axis	Range of motion, software-limited	Speed with rated payload
Nº1	±155˚	154°/s
Nº2	65˚-180˚	154°/s
Nº3	158˚ -15˚	228°/s
Nº4	±350˚	343°/s
Nº5	±130˚	384°/s
Nº6	±350˚	721°/s

Fig. 25. - Axis Data

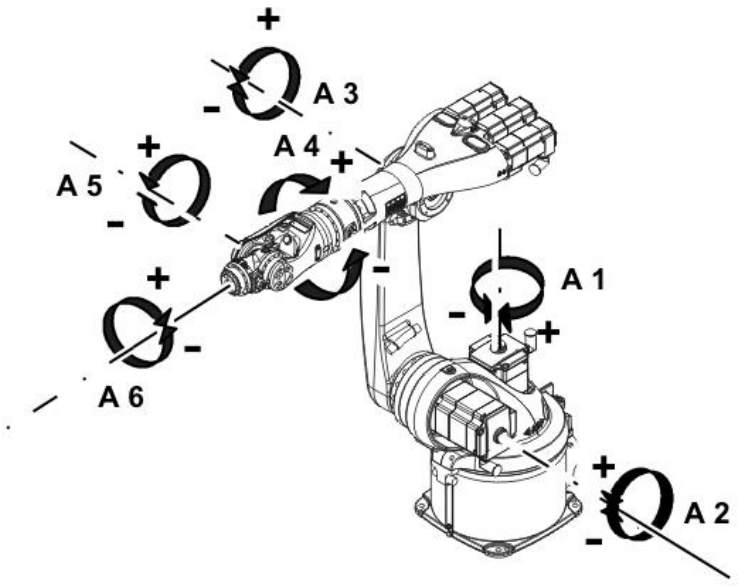


Fig. 26. - Rotation Direction of robot axes.

3.1.2. Robot Controller

The robot controller used in the project is selected in conjunction with the robot, KUKA KR C2. This robot controller takes flexibility into a high level, suitable for use with any KUKA robot, KUKA KR C2 has been designed to adapt.

The KUKA KR C2 controller features a plug-and-play functionality that enables rapid start-up. A low maintenance system, it uses service-proven technology and standard PC components to ensure maximum availability. The modular design of the KR C2 allows for a range of customized hardware and software expansions.

The control cabinet has a functional modular design with a weight of 185 Kg. Also this controller does not have the necessity of be placed near to the robot because the robot controller itself is a power switch.

The KUKA KR C2 consists in the following parts:

- Control PC: Performs all the function of the robot controller, such as: Windows user interface, sequence control and communication with external periphery.
- Power Unit: takes care of all the activities related with power supplement, some component of this unit are power supply unit, IPM motherboard and Mini I/O board.
- KCP Teach Pendant: The KCP (KUKA Control Panel) is the teach pendant for the robot system. The KCP has all the control and display functions required for operating and programming the robot system.
- Riser Cage: provides space for 5 PCI plug-in cards. The integrated network card forms the interface between the control unit and the power unit. Some of the compatible network card are KVGA, Profibus PCI and DeviceNet Card.
- Safety Logic: control and manage all the needed safety procedures in order to avoid any danger.
- Connection panel: consists in all the connection ports available to communicate the robot controller with the robot arm or any external device.

Fig. 27. - KR C2 in property of ACRO

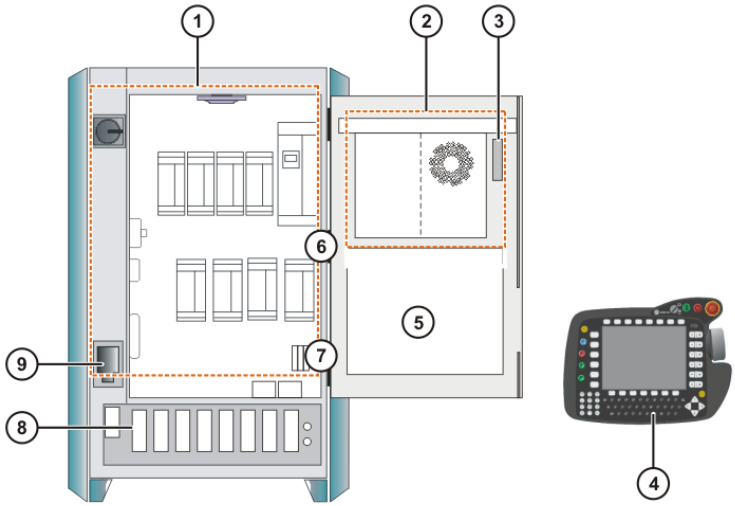


Fig. 28. - Overview of KUKA KR C2

1. Power Unit.
2. Control PC.
3. KCP coupler control and indicator elements (optional).
4. KCP.
5. Mounting plate for costumer components.
6. Safety Logic (ESC)
7. KCP coupler card (optional).
8. Connection Panel.
9. Service Socket (optional).

3.1.2.1 Robot Controller Technical Data

Control cabinet type	KR C2 edition 2005
Colour	See delivery note
Number of axes	Max. 8
Weight	See identification plate
Protection classification	IP 54
Sound level according to DIN 45635-1	Average: 67dB(A)
Installation with other cabinets (with/without cooling unit)	Side-by side, clearance 50mm
Load on cabinet roof with even distribution	1000 N

Fig. 29. - Basic KR C2 Data

Rated supply voltage	AC 3x400V… AC 3x415V
Permissible range of supply voltage	400 V -10%...415 V+10%
Mains frequency	49...61 Hz
System impedance up to the connection point of the robot controller	<= 300mΩ
Rated power input Standard	7.3 kVA, see rating plate
Rated power input Heavy-duty robot Palletizing robot Press-to-press robot	13.5 kVA, see rating plate
Mains-side fusing	Min. 3x25 A slow-blowing, Max 3x32 A slow-blowing, see rating plate
If an RCCB is used: trip current difference	300 mA per robot controller, universal-current sensitive
Equipotential bonding	The common neutral point for the equipotential bonding conductors is the reference bus of the power unit.

Fig. 30. - Power Supply Connection

Ambient temperature during operation without cooling unit	+5 °C to +45 °C (278 K to 318 K)
Ambient temperature during operation with cooling unit	+5 °C to +55°C (278 K to 328 K)
Ambien temperature during storage/transportation with batteries	-25 °C to +70°C (248K to 343 K)
Temperature change	Max 1.1 K/min
Humidity class	3k3 acc. to DIN EN 60721-3-3; 1995
Altitude	- Up to 1000 m above mean sea level with no reduction in power. -1000 to 4000 m above mean sea level with a reduction in power of 5/1000 m

Fig. 31. - Environmental conditions KR C2 Data

3.1.2.2 Teach Pendant KCP

The KCP (KUKA Control Panel) is the teach pendant of the industrial robot, it has all the control and display functions required in order to operate and programme the industrial robot.

Fig. 32. - Front view of the KCP

1. Mode selector switch.
2. Drives ON.
3. Drives OFF/ SSB GUI.
4. EMERGENCY STOP Button.
5. Space Mouse.
6. Right-hand status keys.
7. Enter key.
8. Arrow Key.
9. Keypad
10. Numeric keypad.
11. Softkeys.
12. Start backwards key.
13. Start key.
14. Stop key.
15. Windows selection key.
16. ESC key.
17. Left-hand status key.
18. Menu Keys.

Fig. 33. - Rear view of the KCP

1. Rating Plate.
2. Start key.
3. Enabling Switch
4. Enabling switch.
5. Enabling switch.

3.1.3 Software Controller.

For the KUKA KR 5 arc the software version deployed is KSS (KUKA System Software) 5.5, installed in the robot controller the KSS is responsible for all the basic operator control functions of the Industrial Robot. This control function are:

- Path planning.
- I/O management.
- Data and file management.
- Control of additional technology package (KUKA.Tech)

In order to present to the robot user all the possibilities available with KSS 5.5 KUKA offers its software user interface KUKA.HMI (KUKA Human Machine Interface), it present an easy and flexible environment ensuring a quick learning curve and friendly-programmable editor. Among its features are:

- User Management.
- Program Editor.
- KRL (KUKA Robot Language).
- Inline forms for programming.
- Message Display.
- Configuration window.
- Online help.

Furthermore, KUKA HMI present a great level of customization, ensuring the user satisfy all his necessities. Because of that, user interface may vary from the original standard interface.

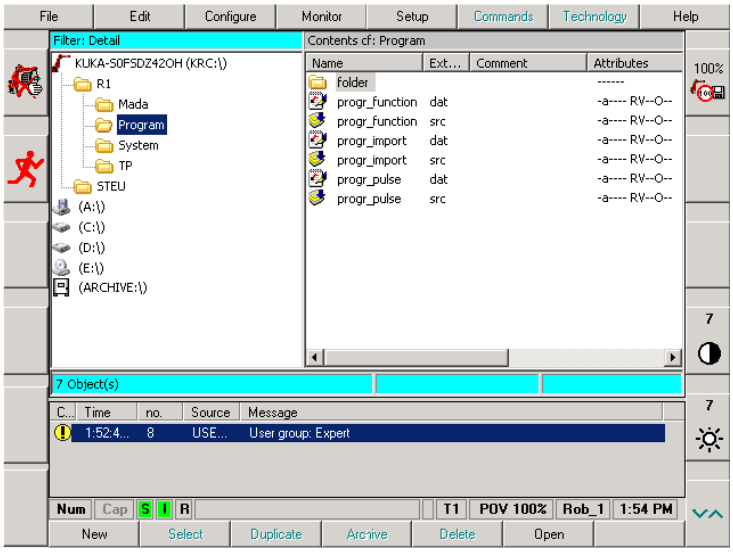


Fig. 34. - KUKA HMI user interface

3.1.4. KUKA. ArcTech Digital Software Package.

In order to control the robot and ensure efficient welds, we have chosen the software Arc Tech Digital of KUKA. This software makes possible not only control the robot, also it ensures operate with the Fronius TPS 4000 in the simplest way. The software package integrates perfectly with KUKA HMI. Additionally, this software have already been configured to recognize the inputs and outputs of welding power unit. Consequently, it is possible form an application with this software package and the welding power unit with almost any pre-set work.

The version of KUKA.ArcTech used in the project is A.20. It is an upgrade of the last version A.10 and present a series of new possibilities and features. The conventional ArcTech features are:

- Menu-guided creation of programs at the user level.
- Simple operation using application-specific softkeys and menus.
- Prepared programs and subroutines.
- Adaptation to the peripheral equipment and configurable options of the digital outputs.
- Simple setting of variables, entries in files and parameter lists.
- Use of the function generators for two--dimensional mechanical weaving as well as the possibility of configuring your own weave patterns.
- Adaptation to various welding controllers with program number control and their different coding systems.
- Various routines used for ignition faults and monitoring of the number of ignition attempts.
- Restart options in case of faults.

In conjunction with this features KUKA ArcTech Digital (A.20) present a range of options in addition with the basic configuration:

- Adaptation to various welding controllers with program number control and their different coding systems.
- Various routines used for ignition faults and monitoring of the number of ignition attempts.
- Re-ignition after faults.
- Restart options in case of faults in the seam.
- Configurable user--specific strategies and routines in case of faults.
- Selection of several defined patterns for mechanical weaving as well as the possibility of programming your own weave patterns.

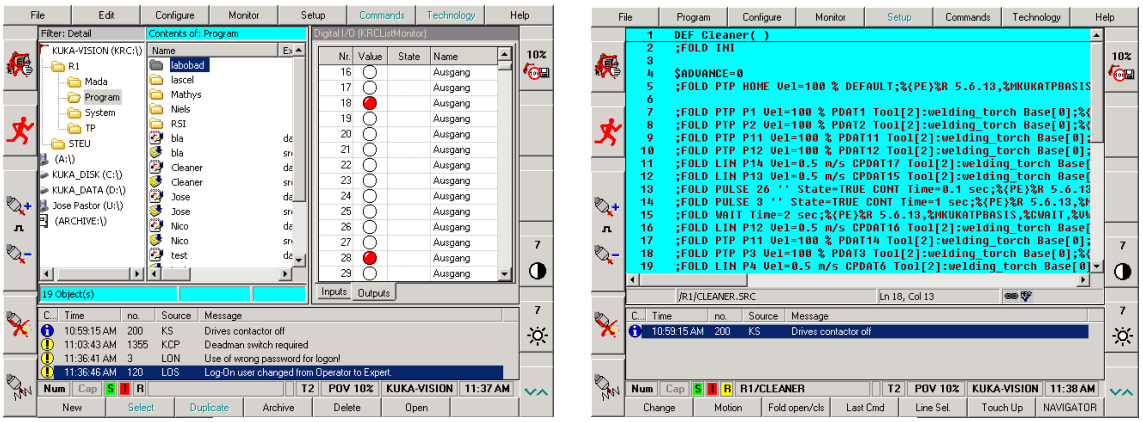


Fig. 35. a,b - KUKA.HMI user interface with KUKA.ArcTech integrated

3.1.4.1 KUKA.ArcTech Digital Interface

With a simple look over the previous images, it is possible to observe some difference between the HMI before and after the installation of the technology package, the new items appeared corresponds with the new possibilities added with the installation, they are explained as follows:

- controlling welding and dry run

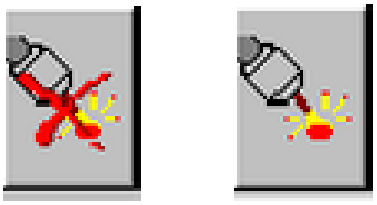


Fig. 36. a,b - HOT/COLD status key

The status key of the previous image is used for manually switching between “HOT” (welding) and “COLD” (robot motion only). This is also called “flying” on/off.

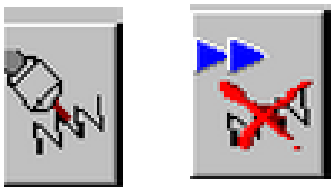


Fig. 37. a,b - Dry status key

The DRY status key is used for the selection of the speed of the robot during the welding action in addition with other parameters. When it is switched on, the robot moves at the default speed setting in Config.dat and the welding process is switched off.

- Wire-feeder +/-

Apart from the two status key items there are to additional items in order to manually adapt the stick-out parameter.

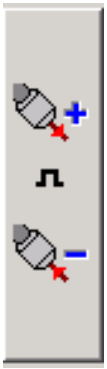


Fig. 38. - Wire-feeder +/- item

3.2. Welding System

The total welding system available in ACRO and implemented in the robotic welding cell is provided by the company FRONIUS, all the components are specially designed for its application in robotic solutions so in that way, we ensure a correct integration between the welding system and the robotic system.

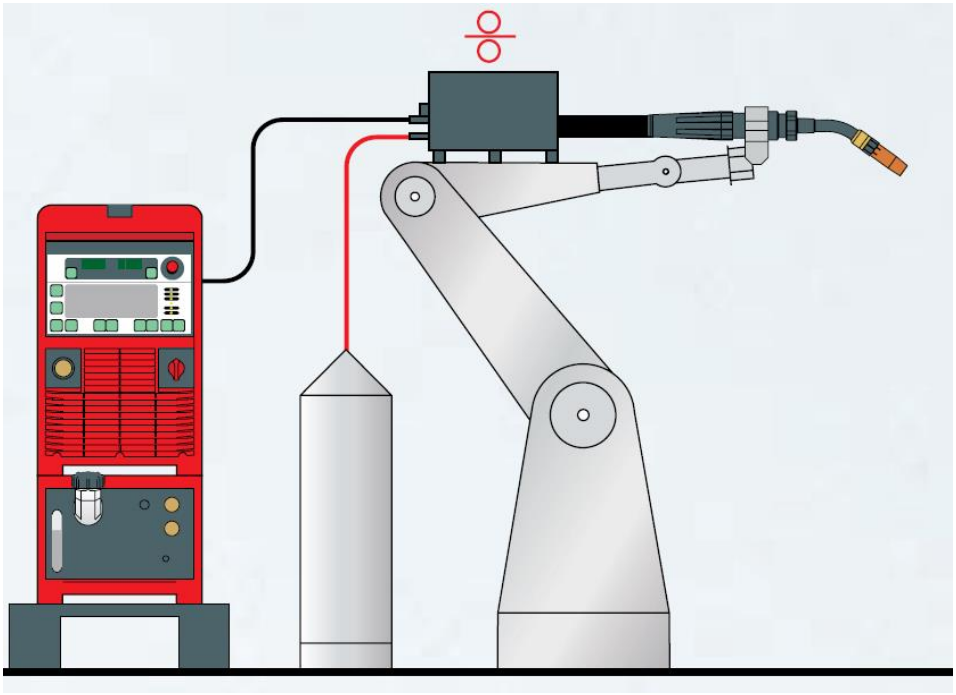


Fig. 39. - Welding System scheme

The system provided by Fronius is composed by this different components:

1. Power Source: Fronius TPS 4000.
2. Wire-feeder: Fronius VR 1500.
3. Cooling Unit: Fronius Fk 4000.
4. Welding Torch: Fronius Robacta Drive W.

Besides of the Fronius's elements, ACRO the protective active gas system and a complete and functional welding torch cleaner cell that complete the system.

3.2.1 Power Source

Fronius TransPlusSynergic 4000 is encompassed in the group of the first microprocessor controlled and digitalized regulated GMA inverted power sources. The most interesting option of this system is the possibility of select from a vast range of job program to develop the desired welding. Also, it is possible to create our job modes and save it in order to use in future applications.

The main principle of Fronius TPS4000 resides in the central control and regulation unit of the power sources. This unit is coupled with a digital signal processor. The central control and regulation unit and signal processor control the entire welding process. During the welding process the actual data is measured continuously and the machine responds immediately to any changes. The control algorithms developed by the manufacturer ensure that the specified target status is maintained. This system process ensures:

- A precise welding process.
- Exact reproducibility of all results.
- Excellent welding properties.

Fig. 40. - - Fronius TPS 4000 ACRO

The Fronius TransPlus Synergic 4000 is a multifaceted professional equipment. It satisfies the highest and more exigent applications in the welding industries. Because of that, this system is applied in many industry sectors such as:

- Car Industry.
- Train Industry.
- Chemist Industry.
- Shipyard Industry.

Also, this equipment ensures weld properly with almost any kind of metal, the project only contemplates steel welding, but Fronius TransPulsSynergic 4000 is adapted a vast range of materials such as aluminum, galvanized laps, tin and chromium/nickel.

The multifaceted characteristic of Fronius TPS 4000 lays in its multiprocessor capacity and knowledge pre-set of every welding process, they make possible to develop MIG/MAG, TIG Manual welding in the more accuracy and efficient option. All this potential is drived by the interface ROB 4000. The robot interface ROB 4000 is an automatic-welder interface and robot interface with analogue and digital outputs and inputs. The ROB 4000/5000 is designed to be installed in an automatic-welder or robot control cubicle (can also be retrofitted).

The advantages of this welder interface are large:

- Analogue inputs and outputs for transmitting process variables (independence from the bit –width of data processing in the existing robot control).
- Power source can be easily changed.
- Limited amount of wiring and cabling needed.
- Simple plug-in connections.
- High degree of interference immunity during data transmission.

For MIG/MAG welding the power source offers a total of 4 operating modes (2-step mode, 4-step mode, special 4-step mode and spot-welding). However, once the interface system has detected the connection with the robot, this operating mode is set to 2-step mode and it remains unchangeable.

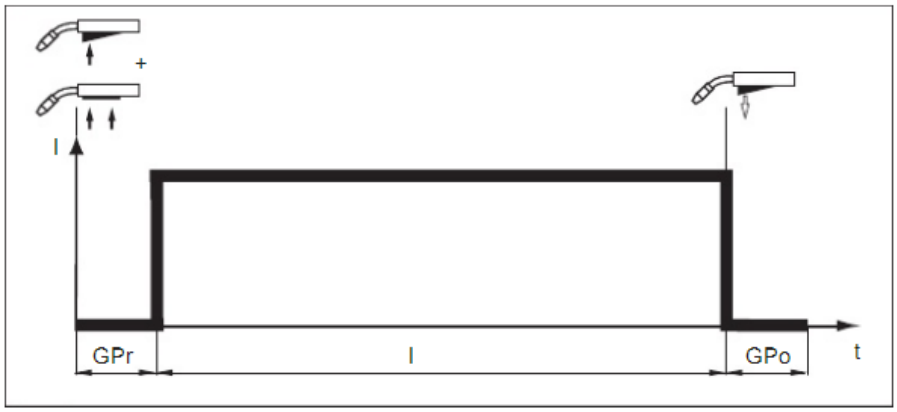


Fig. 41. - 2-step mode functionality

3.2.1.1 Description of the control panel

The functions on the control panels are all arranged in a logical way. The various parameters needed for welding are easy to select by pressing the appropriate button, and can easily be-altered adjusting the dial (“Comfort”), and-shown on the display during welding.

Thanks to the “Synergic” function, whenever you alter any one parameter, suitable adjustments will automatically be made to all the other parameters.

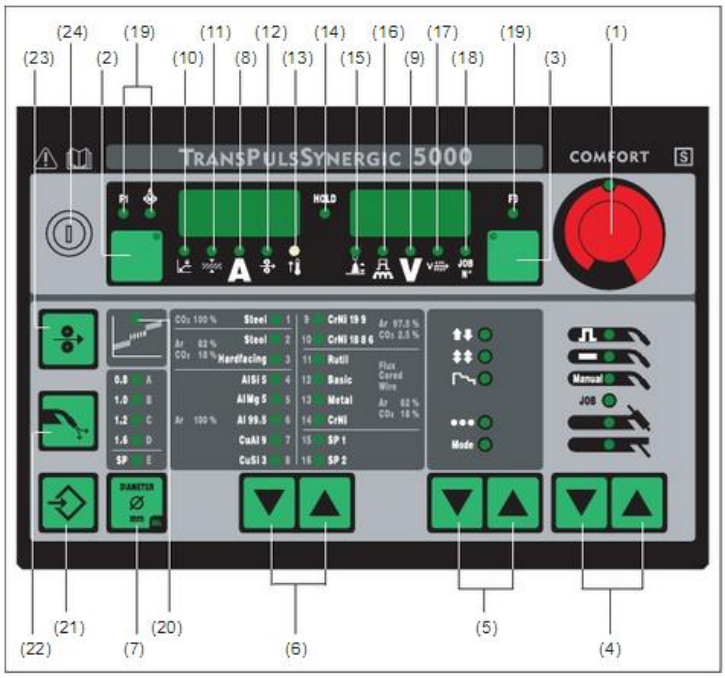


Fig. 42. - "Comfort" Control Panel

1. Adjusting dial: for changing parameters. If the indicator next to the adjusting dial is light up, then the selected parameter is one: that can be altered.
2. Parameter selection button: possibility of selection of the following parameters:
    - "a"-dimension-
    - Sheet thickness.
    - Welding current.
    - Wirefeed speed.
    - F1 indicator.
    - Wirefeed-drive current-input indicator.
3. Parameter selection button: selection of the following parameters:
    - Arc-length correction.
    - Droplet-detachment / arc.
    - Force (dynamic) correction.
    - Welding voltage.
    - Welding speed.
    - Job n°.
    - F3 indicator.
4. Process buttons: selection between the different welding process available
    - MIG/MAG pulse synergic welding.
    - MIG/MAG standard synergic welding.
    - MIG/MAG standard manual welding.
    - Job mode.
    - TIG welding with touch-down ignition.
    - Rod electrode (MMA) welding.

5. Process buttons: selection of the operating mode:
    - 2-step mode.
    - 4-step mode.
    - Special 4-step mode.
    - Spot welding mode.

6. Material buttons: for selecting the filler metal and the shielding gas used in the welding application.
7. Wire Diameter button: Selection of the wire diameter to be used.
8. Welding current parameter.
9. Welding voltage parameter.
10. "a" dimension parameter.
11. Sheet thickness parameter.
12. Wire-feed speed parameter.
13. Over-temperature indicator.
14. Hold indicator: memory of the last customization of parameters applied.
15. Arc-length correction parameter.
16. Arc force correction and Arc force parameter.
17. Welding speed parameter.
18. JOB Nº Parameter.
19. F1/wire-feed drive current input/F3 indicators.
20. Intermediate arc indicator.
21. "Store" button.
22. Gas-test button.
23. "Feeder inching" button.
24. Key-lock Switch.

3.2.1.2 Technical Data

Mains voltage			3 x 400 V
Mains voltage tolerance			± 15 %
Power frequency			50 / 60 Hz
Mains fuse protection			35 A slow-blow
Primary continuous current (100 % d.c.)			-
Primary continuous power			12.2 kVA
Cos phi			0,99
Degree of efficiency			88 %
Welding current range:			
	MIG/MAG		3 -400 A
	Rod electrode (MMA)		10 -400 A
	TIG		3 -400 A
Welding current at	10 min/40°C (104°F)	50 % d.c.	400 A
		60 % d.c.	365 A
		100 % d.c.	320 A
Welding voltage range according to standard characteristic			
	MIG/MAG		14.2 - 34.0 V
	Rod electrode (MMA)		20.4 - 36.0 V
	TIG		10.1 - 26.0 V
Max. welding voltage			-
Open-circuit voltage			70 V
Protection			IP 23
Type of cooling			AF
Insulation class			F
Marks of conformity			CE, CSA
Safety			S
Measurements l x b x h			625 x 290 x 475 mm 24.6 x 11.4 x 18.7 in.
Weight			35.2 kg 77.6 lb.

Fig. 43. - Technical Data Fronius TPS 4000.

3.2.2 Wire-feeder

The Fronius VR 1500 is a wire-feed system specially designed to be mounted in a robot arm. Small, lightweight, digitally controlled robot wire-feeder with 4-roller drive, ideal for mounting on the 3rd axis of the robot.

It can be used in conjunction with the TS 4000 / 5000, TPS 3200 / 4000 / 5000 power sources. Furthermore, it is suitable for all standard shielding gases and from a vast range of welding speeds:

- 11 m/min.
- 12 m/min.
- 22 m/min.
- 30 m/min.

Because of its modular design, the VR 1500 robot wire-feeder can be extended as needed (e.g. Robacta Drive, air-flush, torch support etc.). The Fronius VR 1500 is specially designed for filler wire applications but it also is suitable for different welding applications:

- Complex wire-feeding applications requiring a high degree of torque:
  - Solid flux core wires up to 2.3 mm on large spools.
  - Solid flux core wires up to 2.3 mm in welding wire drums.

- Flat wire applications.

Fig. 44. - Fronius VR 1500 implemented in ACRO Project

3.2.2.1 Technical Data

Supply voltage	55 V DC
Rated current	4 A
Wire diameter	0,8 - 1,6 mm .03 - .06 in.
Wire speed	0,5 - 22 m/min 19.69 - 866.14 ipm.
Torque	4 Nm
Degree of protection	IP 21
Dimensions L / W / H	405 / 208 / 205 mm 15.94 / 8.19 / 8.07 in.
Weight	7 kg 15.43 lb.
Drive	4-roller drive
Max. shielding-gas pressure	7 bar 101 psi.
Coolant	Original Fronius coolant
Max. coolant pressure	6 bar 87 psi.

Fig. 45. - Technical Data Fronius VR 1500

3.2.3. Cooling Unit

The welding system need a cooling unit in order to control the high temperatures produced in the power source during the welding action. The cooling unit used in the project is the Fronius Fk 4000 R. This cooling unit present a vast range of features that fits perfectly with the requirements in our application. It also present a great standardization that ensures use it with almost any power source from Fronius's catalogue. Despite that, there are others cooling units in Fronius FK 4000 series in order to satisfy a vast power source catalogue of Fronius, a briefly list of this cooling units of this series is presented below:

- FK 4000: with vibrate pump.
- FK 4000: with centrifugal pump.
- FK 4000 R FC: with centrifugal pump and flow watchdog.
- FK 4000 R US: with centrifugal pump and flow sensor.
- FK 4000 Rob: with magnetically-coupled centrifugal pump, flow sensor and thermostat.

All this cooling units have the same appearance, they only differ in terms of inside components and characteristics.

Fig. 46. - Fronius FK 4000 R

The Fronius Fk 4000 R is specially designed in order to be applied in the following welding process:

- TS and TPS 4000/5000 power sources.
- JobMaster welding torches.
- Push-Pull welding torches.
- Robot welding.
- Hosepacks over 5 m long.
- MIG/MAG pulsed-arc welding.
- Where welding is performed in higher power ranges.
- In conjunction with Robacta machine hosepacks.
- In conjunction with Robacta or Robacta Drive robot hosepacks.

Here are presented the parameter that has to be though in order to select the proper cooling unit:

- The type of pump used.
- The ambient temperature.
- The discharge head.
- The through-flow rate Q (l/min).
- The through-flow rate Q will depend upon the length of the interconnecting cable and on the diameter of the hose.

3.2.3.1 Technical Data

FK 4000 R, FK 4000 Rob and FK 4000 R FC	Supply voltage (from the power source)	3 x 400 V. 50 Hz
	Power consumption	0.5 A / 0.6 A
	Cooling capacity: Q = 1 l/min +25°C (77°F)	1360 W
	Q = 1 l/min +40°C (104°F)	870 W
	Max. delivery capacity	3.5 l/min
	Max. pump pressure	4.2 bar (60.89 psi.)
	Max. delivery height	~ 33 m (108 ft. 3.24 in.)
	Type of pump	Centrifugal pump
	Coolant volume	5.5 l
	Degree of protection	IP 23
	Dimensions (L x W x H)	725 x 290 x 230 mm (28.54 x 11.42 x 9.06 in.)
	Weight (without coolant)	13.3 kg (29.32 lbs.)

Fig. 47. - Fronius Fk 4000 R technical data.

3.2.4 Welding Torch

The welding torch Fronius Robacta Drive W guides on the 6th axis of the robot, this ensures adapt the torch to a vast range of different applications.

The contact tube plays an important role in the specifications, it is made of high-grade copper alloy ensuring a long service life. Its measure of 10x40 mm can dissipate the heat very effectively and stable contracting.

On the water-cooled welding torches, the forced wire contact makes sure that the welding wire is guided into the contact tube at an exactly defined angle. Dependable current transfer is the result, even when using a bulk pay-off packs where the curvature of the wire would otherwise impair the wire guidance. Another positive result of the wire contact, incidentally, is a boost of up to 100 % in the contact tube endurance time.

This welding torch has many number of features to make sure that the wire gets to where it is needed without the slightest margin of error. One of them is the digital actual-value pick-up, which causes exactly as much wire available as the system requires at any given moment. Another feature is the drive rollers, which are intermeshed so that the force is transferred to both rollers simultaneously. Their size of 23 mm guarantees optimum force transmission at all times with less risk of slippage. Also, the feed rollers provide 4 point contact support to the wire as this run through a specially shaped groove.

The standard features of the Fronius Robacta Drive are:

- Exact speed regulation assured by digital encoder.
- Gas-test button.
- Graphite inner liner diam. 2.5 mm (for Al and CrNi wires).
- Interchangeable torch bodies.
- Powerful gear-motor unit.
- Separate gas and blow-out lines.
- Superb torch cooling.
- Swirl-free gas-flow - no loss of gas.
- Toothed pressure and drive rollers.
- UV, temperature and ozone-resistant rubber fabric hoses.
- Wire-feed FWD/BACK button.

Fig. 48. - Robacta Drive W in ACRO

3.2.4.1 Technical Data

	Welding Amperage	Duty Cycle	Torch-Body Angle	Robacta Drive Hose Pack	Lenght
Robacta Drive W	700 A	100 %	45˚	Water Cooled	8.25 cm

Fig. 49. - Robacta Drive W Technical Data

3.2.5. Protective gas system.

MAG process are based in the inclusion of a protective active gas in order to ensure the correct mix of element in the welding environment. In our case, the gas used is a mix of Dioxide of Carbon and Argon. (85% Ar + 15% CO_2).

The gas cylinder is connected to the power source and it with a pre-customization of the parameters regulates and controls the adequate flow protective gas in every application.

3.2.6. Welding Torch Cleaning Cell

The welding cell in property of ACRO centre contains a complete cleaning station which ensures a proper cleaning and lubrication of the weld torch.

This cell is totally integrated in the communication system and it works under the orders of the robot controller. This control is done through the use of 2 outputs from the robot (2, 3). The output 2 activates the spinning of the cylinder and the piston that fastens the weld torch. The output 3 activates the rise of the cylinder and the wire cutting action.

The cleaning procedure with its respective KRL code will be explained in section 5 of this paper.

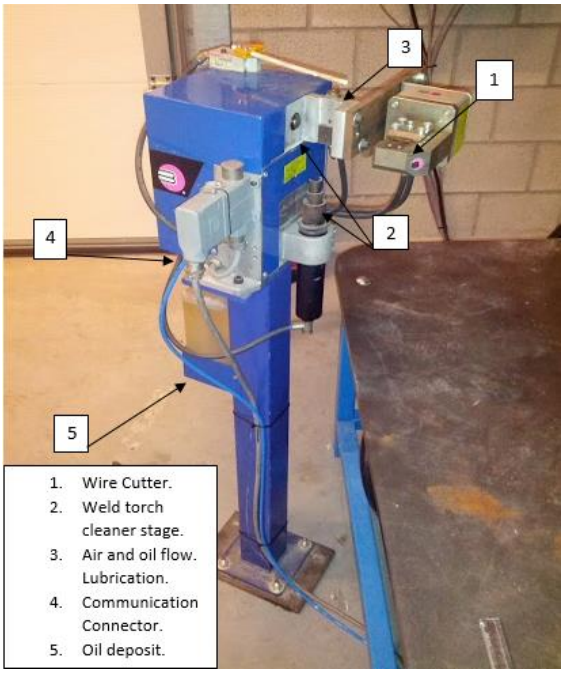


Fig. 50. - ACRO's cleaner cell

3.3. Vision System

Vision System encloses the software and all the hardware necessary in order to develop the machine vision task, in our case, our vision system in formed in the software HALCON with its programming environment HDevelop. The image of the work-piece are acquired by the camera Ueye UI 1465-LE by IDS.

3.3.1. HALCON Software

HALCON is one of the best exponents of the correct definition of the state of art in machine vision software. It provides an extensive vision library, many vision procedures in order to solve a vast range of task with the highest speed and accuracy possible.

HALCON is not only focus on solving image processing, it also present more features such as: process control, database access, control of hardware (image acquisition, illumination, mechanical components etc.). Despite this possibilities, HALCON is perfectly optimized to present an easy learning and use in vision system tasks.

HALCON takes cares of all the important aspects:

- The software development is supported by the interactive tool HDevelop, which enables a quick development of image processing tasks combined with an easy integration into standard development environments like Microsoft Visual Studio via the automatic code export.

- The problem-oriented documentation covers all levels from a quick access to important information up to a detailed discussion of advanced topics.
- These descriptions are combined with hundreds of examples for an intuitive understanding of the solutions, which can serve as templates to shorten the development time.
- Last but not least, HALCON provides open interfaces for efficient data exchange, to integrate own operators, or to access specialized hardware round off the system.

3.3.1.1 Develop task in HALCON

HALCON as a flexible and adaptive tool offers many ways for application development. However, in order to use all the potential of HALCON the architecture mode has to follow a proper form of development.

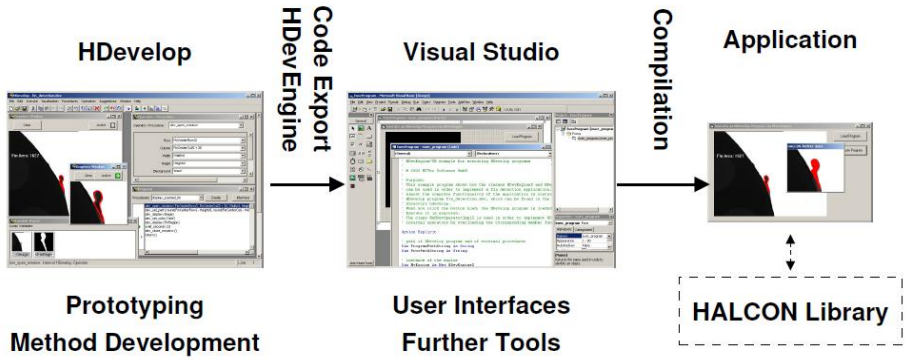


Fig. 51. - Three-step HALCON procedure

All the activities related with Image inspection, prototyping of the vision method or the final development of the vision method are done within HDevelop. In this programming environment the program is structured into different sub-system where the sub-task that conform the final task are done, once all the sub-task are developed all the procedure are called in the main program, the image are passed into this procedures and the final result are obtained. Afterwards, the program is exported to the desired programming language.

The complete application is developed in a programming environment such as Microsoft Visual Studio. The code from HDevelop is imported and the user interface and other necessary code is implemented using the normal mechanisms offered by the given language. Finally, the project is compiled and linked.

Together with the HALCON library, the total program represents a complete and functional application that can be used in an isolated way. The three-step approach has several advantages:

- Whenever needed the vision part can easily be optimized or extended because HDevelop offers much better inspection and debugging facilities for image data than the standard programming environments.
- A newly exported HDevelop program can be incorporated into the programming environment quite easily because the code is included and requires modifications in the general code only if the parameters have been changed or new procedures have been introduced. This closes the development cycle in a natural manner.

- Because the vision part is separated from the general code it can easily be executed in a standalone manner. Furthermore, it can be given to others without the need to pass the whole project. Especially in the case of support questions, the HDevelop program with one or more images can quickly be sent to the distributor.

3.3.1.2 Data Structure and Architecture

Architecture, data structure and all the procedures of HALCON were developed in order to satisfy the highest requirements in computational issues, here are outlined the premises of HALCON:

1. *Efficient*. Means that the execution time of each HALCON operator should be as short as possible. Furthermore, the operator design has been made such that combinations that are standard sequences or more complex tasks must still remain efficient.
2. *Open*. The open architecture is important in two respects: First, it must be possible to make use of HALCON from many different languages. Here, passing of external data to HALCON and accessing internal data of HALCON must also be supported. Finally, there must be transparent interfaces to integrate user-defined operators and non-standard image acquisition devices. This open architecture allows, e.g., a simple update to a new version of a frame grabber interface without changing the installation of HALCON.
3. *Standardized*. Standardized means that the signatures, naming, and usage of operators and data strict rules. This allows a quick learning combined with few possible errors.
4. *Self-describing*. HALCON provides detailed information about each operator and their parameters not only in the documentation but also online via specialized operators.

3.3.1.3 HALCON operators

All type of functionality is obtained from HALCON's library and it is done by a proper operator. There are more than 1100 operators in the current version of HALCON. Most of them conform multiples functionalities in one parameter and it are selected via the given parameters. The most important shared features between the different HALCON operators are described as follows:

- There is no hierarchy among operators. From the software architecture point of view, all operators are on the same level.
- There are logical groups of operators. This can directly be seen by the classes offered for C++ and COM, where operators processing the same data type are used as members of the corresponding classes.
- Operators have standardized rules for ordering input and output parameters.
- The design of operators follows the rules of the open architecture. Therefore, you can create your own operators and thus extend HALCON.
- Many operators can make transparent use of automatic parallelization, which allows an easy way of speeding up the program when using large images on a multi-CPU computer.

3.3.1.4 Parameters and Data Structures

Here are the most important characteristic of parameters and data structures:

- HALCON has two basic types of parameters: iconic data (images etc.) and control data (integers, handles, etc.).
- The parameters for each operator are arranged in a standardized order: input iconic, output iconic, input control, and output control. Not all of the groups might be needed for a given operator. However, the order remains being the same.
- Each operator has a self-describing interface. This description contains, besides the standard documentation, information about parameters like types or value lists, which can be accessed online.
- Input parameters of operators are never modified, which results in a very clear and simple semantics.
- The open architecture allows to access internal data and integrate external data.
- All necessary data structures for 2D and 3D image processing like (multichannel) images, region, contours, tuples, etc. are directly supported using an extremely efficient implementation.

Two of the most representative data structures in Halcon are Images and Control Tuples:

-Images
Images belong to the iconic data. The major part of an image are the channels, i.e., matrices containing the grey values of various pixel types.

For each image, the so-called *domain* specifies which part of the image is processed. It thus acts as a *region of interest* (ROI). The domain is a HALCON region and can therefore be defined very flexibly.

Almost arbitrary content is possible, from standard 8-bit grey values to floating-point numbers describing derivatives. For integer values one, two, and four byte versions (with and without sign) are available. Besides this, floating point and complex images are available.

An image channel corresponds to an image matrix. Each image can have an arbitrary number of channels, all of this channels have the same size. The most common channels are: single-channel grey value-image, colour image with three channel (RGB) and multichannel image from a multispectral sensor or as a texture filtering result. There are also special data types for describing edge direction or hue values are supported.

Finally, in order to define the coordinate system in an image, the origin of an image is the upper left corner with coordinates (0,0). The single pixels are accessed using row and column coordinates, like in a matrix. The coordinates range from (0,0) up to (height-1, width-1). A pixel has an extent of 1, whereas the centre of gravity of the first pixel of an image is (0,0). This has the effect that this pixel ranges from (-0.5, -0,5) to (0.5,0.5).

- Control Tuples
Tuples are the generic data type for integer and floating point values such as strings. A variable of type tuple can be formed by any of three simple variable kinds.

Besides single values, arrays of the basic types are supported. Therefore, one variable can contain none, one, or an arbitrary number of values, where the types of each element can be

different. In most cases, single values are treated in the same way as multiple values. If, e.g., a feature operator is called with a single region one feature value is returned. When the operator is called with multiple regions a tuple with the corresponding number of values is returned. The index of tuples range from 0 to the number of values minus 1.

3.3.1.5 Image acquisition

In last upgraded version of Halcon there are provided interfaces about 40 frame grabbers in the form of dynamically loadable libraries (Windows: DLLs; UNIX: shared libraries). These libraries are installed together with the Halcons libraries. Library names start with the prefix HFG; the libraries starting with par HFG are used by Parallel Halcon. The Halcon frame grabber interface libraries form the bridge between software provided by the frame grabber's manufacturer and Halcon. They form a common, generic interface that requires a small set of operators only.

If the frammegrabber is successfully installed, there is only has to access it from Halcon is to call the operator open framegrabber, specifying the name of the framegrabber and some additional information, regarding the connected camera. Then, images can be grabbed by calling the operator *grab_image* or *grab_image_async*.

3.3.1.6 Hdevelop

HDevelop is a powerful environment for both prototyping and method development. The easy-use Hdevelop's environment ensures its use with a few concepts of vision and programming.

Furthermore, there are a vast range of examples with almost all the operators available in HALCON's libraries.

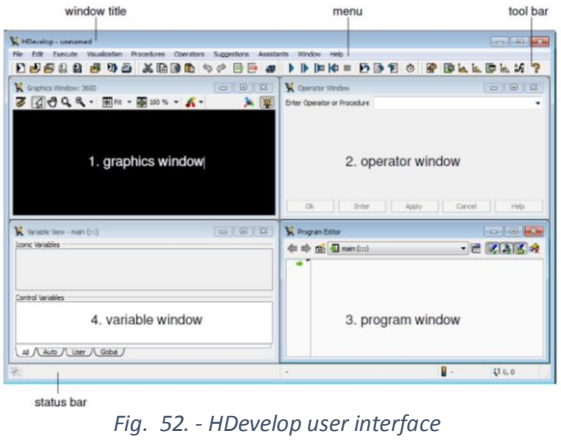


Fig. 52. - HDevelop user interface

After loading the file, the corresponding program code is displayed in the program window. The used variables - so far not instantiated - can be seen in the variable watch window. The program is now ready for execution.

Steps to run a program:

1. Press the Run button to execute the program. To continue at a stop statement, press Run again.

2. Besides the Run button, HDevelop provides a Step button, which executes only a single line and displays the results immediately afterwards. If the program contains procedures, it might be of interest to use the buttons Step Into and Step Out.
3. To run the complete program the Reset button can be used. To run specific parts only, simply click with the mouse to the left of the desired program line. This will reposition the program counter. When executing the program anew it will then start at the newly selected position.

A list of useful tips for HDevelop are described as follows:

- At the lower end of the main window, HDevelop provides a status bar. This displays useful information in many cases. Especially during the execution, when the program stops to visualize results or waits for a user interaction corresponding instructions are given.
- Many programs will automatically display relevant data in the graphics window. Manual visualization can easily be achieved by double clicking on the icons in the variable watch window.
- Some programs use frame grabbers for image acquisition. If the corresponding frame grabber type is not available, an error message will be raised. In this case, we recommend to either use another example or to modify the parameters to fit to the available hardware.

3.3.1.7. Using HALCON in different Programming languages

HALCON offers special exportation of language interfaces. They are libraries that enable you to call the operators and to use the data types of the HALCON library in an easy way. Two language interfaces are designed for specific languages. These are the C and the C++ interfaces. In contrast, the COM interface is independent of a given language. It can be used,
e.g., with Visual Basic, C#, or Delphi.

Independent of which programming language you choose, a suitable interface library
(HALCONc.*, HALCONcpp.*, HALCONx.*) together with the HALCON library
(HALCON.*) must be linked to the application. In addition for C and C++, the corresponding files such as images and acquisition drivers must be included.

3.3.2. Camera

The camera implemented in the project constitutes a cheap solution in order to detect the 3D position and orientation information of simple pieces. The camera is developed by IDS and the model is the UEye UI1465-LE. It is a 2D camera used in our application as an area scan device.

Furthermore, the camera is mounted in conjunction with the objective Fujinon 1:1.4/25 mm HF25HA 1B characterised by its 25 mm of focus length.

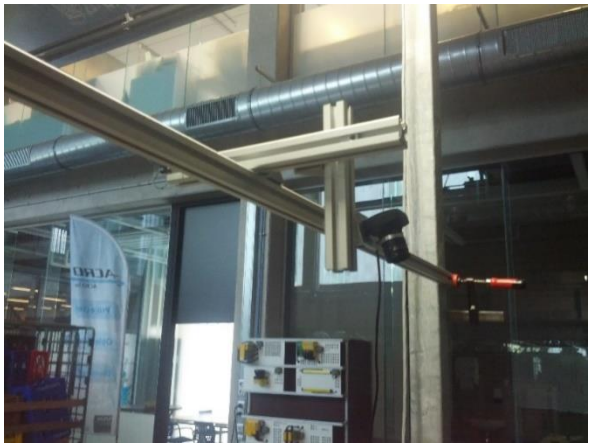
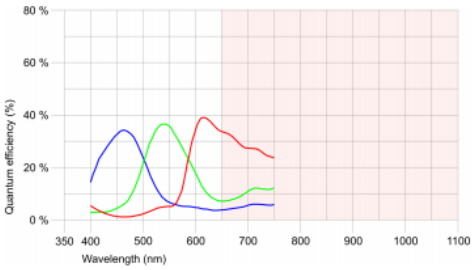


Fig. 53. - Camera and objective used

3.3.2.1 Specifications

Sensor	
Sensor Technology	CMOS Color
Manufacturer	Aptina
Resolution (h x v)	2048 x 1536
Color depth (sensor)	10 bit
Color depth (camera)	8 bit
Pixel Class	3 MP
Sensor Size	1/2"
Shutter	Rolling shutter
max. fps in Freerun Mode	11.2
Binning Modes	Color
Subsampling Modes	Color
Sensor Model	MT9T031
Pixel size	3.2 µm
Optical Size	6.554 mm x 4.915 mm



Design	
Interface	USB 2
Lens Mount	CS- / C-Mount
I/O In	-
I/O Out	-
I/O RS-232	-
I/O GPIO	-
I/O I2C	-
Protection Class	IP30
Dimensions H/W/L	48.6 mm x 44.0 mm x 25.6 mm
Mass	41 g
Power supply	USB Cable

Fig. 54. - UEye UI 1465-LE Specifications

3.3.3. Illumination

In order to avoid the formation of shadows that affect negatively to the matching process of the work-piece it is necessarily to introduce a proper illumination system.

The illumination system is formed by two fluorescent tubes situated above the camera. In that way the work-zone is correctly illuminated avoiding any shadows and possible reflections of the light that could interfere in the recognition process.

Chapter 4

4. Installation and Pre-set Work

In this section information about communication between the Robotic System and the Welding System are going to be outlined.

Here is explained also all the setting tasks performed in order to achieve the correct installation and function of both systems.

4.1. Communication System

Once the Robot and the power source have been installed, it is necessarily establish a communication system between them in order to ensure the correct transmission of the I/O of both systems which base the decision of the robot controller.

The communication is based in a DeviceNet application with CanBus, this technology has been selected because of its flexibility and its open structure.

4.1.1. DeviceNet

DeviceNet is a low-cost communications technology designed to connect industrial devices (such as limit switches, photoelectric sensors, valve manifolds, motor starters, process sensors, bar code readers, variable frequency drives, panel displays and operator interfaces) to a network and eliminate expensive hardwiring.

The direct connectivity provides improved communication between devices as well as important device-level diagnostics not easily accessible or available through hardwired I/O interfaces.

DeviceNet is an open network standard. The specification and protocol are open – vendors are not required to purchase hardware, software or licensing rights to connect devices to a system. Anyone may obtain the DeviceNet Specification from the Open DeviceNet Vendor Association, Inc. (ODVA)

Any company that use and develop DeviceNet products may join ODVA and participate in technical working groups that are developing enhancements to the DeviceNet Specification.

Buyers of the DeviceNet Specification receive an unlimited, royalty-free license to develop DeviceNet products. Companies looking for assistance can purchase sample code that eases their implementations, development toolkits, and development services from many sources.

The key hardware components are available from the largest worldwide suppliers of semiconductors.

4.1.1.1 DeciveNet Specifications

The DeviceNet Specification defines a network communication system for moving data between elements of an industrial control system. The specification is divided into two volumes and defines the following elements:

- Volume 1
  - DeviceNet Communication Protocol and Application (Layer 7 - Application Layer)
  - CAN and its use in DeviceNet (Layer 2 - Data Link Layer)
  - DeviceNet Physical Layer and Media (Layer 1 - Physical Layer)

- Volume 2
  - Device Profiles to obtain interoperability and interchangeability among like products.

DeviceNet incorporates CAN (Controller Area Network). CAN defines the syntax or form of the data movement. The DeviceNet application layer defines the semantics or meaning of the data moved.

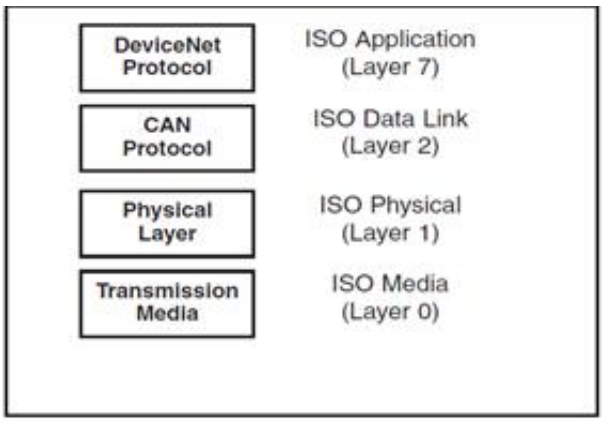


Fig. 55 - Layer Protocol (ISO Layer 7)

4.1.1.2 Communication Protocol Features

- Peer-to-Peer data exchange in which any DeviceNet product can produce and consume messages.
- Master/Slave operation defined as a proper subset of Peer-to-Peer.
- A DeviceNet product may behave as a Client or a Server or both.
- A DeviceNet network may have up to 64 Media Access Control Identifiers or MAC IDs (node addresses). Each node can support an infinite number of I/O. Typical I/O counts for pneumatic valve actuators are 16 or 32.

A DeviceNet node is modelled as a collection of Objects. An object provides an abstract representation of a particular component within a product. The realization of this abstract object model with a product is implementation dependent.

4.1.2. Communication Application for the Welding Cell Project

4.1.2.1 Overview of the application

The communication between the robotic and the welding systems in our project has been created following the instances declared by DeviceNet, adaptable, open and flexible, in that way, this communication ensures future expansions of the system without almost any re-adaptation.

All the connexions between the different devices are deployed in a communication cabinet, there the different outputs of the power source are transformed as input for the robot controller and vice versa. In this cabinet also the signals that regulate the cleaner cell are integrated. In the following image there are represented a simple diagram of who the communication system are developed.

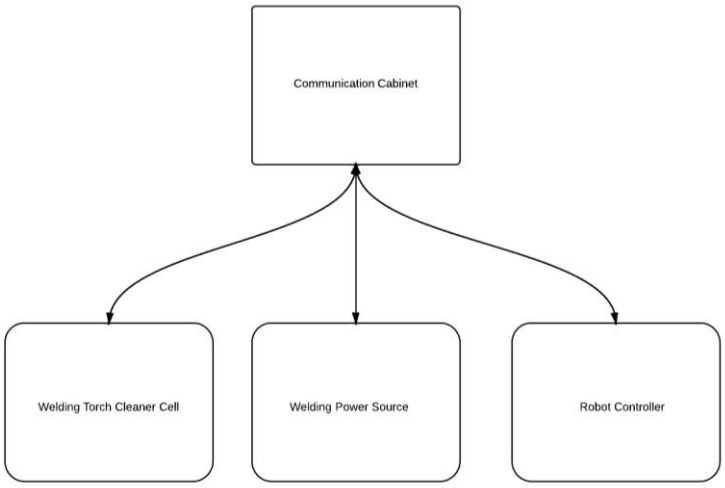


Fig. 56. - Simplified Communication Diagram

Apart from creating the connexions between the devices, some software actions has to be developed in KUKA KSS 5.5 in order to delimit the signals and give some information about who the information is received and sent. This actions will be explained in the following sections.

4.1.2.2 Communication Cabinet

As seen in the previous point, all the communication connexions between the power source, robot controller and welding torch cleaner cell are encompassed in a cabinet.

This cabinet apart from respective connexions has the needed safety elements of the communication and also two buttons with two led that ensure reset the power source through the activation of the outputs of the robot.

The cabinet is built in metal and has a dimensions of 60x40x20 cm and also it has a frontal door in order let a quick access of connectors and fuses.

Fig. 57. - a,b Communication Cabinet

The electrical schematics of the connections in the control cabinet are in the *A1.1* section of the appendix.

As we can see in the schematics and in the pictures of the cabinet there are a voltage source of 48 voltages (Mark 48) implemented in the communication system. This power source is connected directly to an input of the robot controller and the welding wire. This power source was implemented in order to receive a direct signal in the robot controller every time the welding action is producing a shortcut between ground and the filler metal.

4.1.2.3 Communication Software Settings

In the robot controller we took a file already created in the company SIRRIS, in it there are deployed all the needed settings in order to create the DeviceNet application, this application is stored in the file *"iosys.ini".* The only customization that we have to do was select the correct range of the I/O signals because in our communication application, the order of the signal is different than in SIRRIS's application. The data that we changed was the definition of the range of the different input and output that define the DeviceNet application.

We are not going no comment all the definition because is not the purpose of this paper. However, we have attached the file of the I/O configuration to the appendix section *A1.2***.**

One of the most important definition carried out was the differentiation between the signals that connect the FRONIUS power source and the robot controller and the signals that only are used by the robot controller. In this aspect two group were formed:

1. Connected only with the robot controller: Input and Output from 1 to 8 (1 byte)
2. Connected with the robot controller and power source: Input and Output from 17 to 112 (12 bytes).

In the first group are also encompassed the signals that regulate the welding torch cleaning output 2 and 3 of the robot.

The lines of code of the application were this differentiation is deployed are outlined as follows:

```
[DEVNET]
;Devicenet MACID 1 (kast)
INB0=1,0,x1  ;$in[1-8]
OUTB0=1,0,x1 ;$out[1-8]
;Devicenet MACID 21 (lasbron FRONIUS) 12 bytes in en uitgangen
INB2=21,0,x1 ;$in[17-24]
INB3=21,1,x1 ;$in[25-32]
INB4=21,2,x1 ;$in[33-40]
INB5=21,3,x1 ;$in[41-48]
INB6=21,4,x1 ;$in[49-56]
INB7=21,5,x1 ;$in[57-64]
INB8=21,6,x1 ;$in[65-72]
INB9=21,7,x1 ;$in[73-80]
INB10=21,8,x1 ;$in[81-88]
INB11=21,9,x1 ;$in[89-96]
INB12=21,10,x1 ;$in[97-104]
INB13=21,11,x1 ;$in[105-112]
```

Following the schematics of the communication cabinet attached in the appendix, it is remarkable outline the necessity of establish a list of output of the robot in high value to authorize the communication between both system. Also this outputs have to be high when a fault in the welding process has happened and a reset of the system is needed. The list of the outputs are showed below:

- Output 1: provokes the activation of input 5 from the power source
- Output 4: Provokes the activation of the input 3 from the power source
- Output 8: Activates contactors 1K7 and 1K8 to authorize the welding action.
- Output 18: send the information to the power source that the robot is ready
- Output 28 reset the power source after a fault in the welding process.

4.1.2.4 I/O Signals Definition

In the following tables there are outlined the definition of the signals used that connect welding system with the robotic system for MIG/MAG process.

Input	Remarks
17	Welding Start
18	Robot Ready
19	Bit 0 operating modes
20	Bit 1 operating modes
21	Bit 2 operating modes
22	Master selection twin
23	Not in use
24	Not in use
25	Gas Test
26	Wire inching
27	Wire retract
28	Source Error Reset
29	Touch Sensing
30	Torch blow trough
31	Not in use
32	Not in use
33-40	Job Number
41-47	Program Number
48	Welding simulation

Fig. 58. - Power source input/ Robot controller output process control

Operating Mode	19	20	21
Program Standard	0	0	0
Program Pulsed-arc	0	0	1
Job Mode	0	1	0
Parameter selection internally	0	1	1
Manual	1	0	0
CC/CV	1	0	1
TIG	1	1	0
CMT/Special Process	1	1	1

Fig. 59. - Power source input/ Robot controller output operating modes

Output	Remarks
17	Arc Stable
28	Limit Signal
19	Process Active
20	Main Current Signal
21	Torch Collision Protection
22	Power Source Ready
23	Communication Ready
24	Spare
25-32	Error Number
33-40	Not in use
41	Wire stick control
42	Not in use
43	Robot access
44	Wire available
45	Timeout short circuit
46	Data documentation ready
47	Not in use
48	Power outside range

Fig. 60. - Power source output/ Robot controller input process control

4.1.2.5 Signal Interface in KUKA KSS 5.5

KUKA KSS 5.5 offers in its KMI an I/O signals interface in order to control of the signal and a correct visualization of the process state in real-time. Also in this interface it is possible to control the output from the robot sent to the power source and the welding torch cleaning cell with only one click or press one button in the KCP.

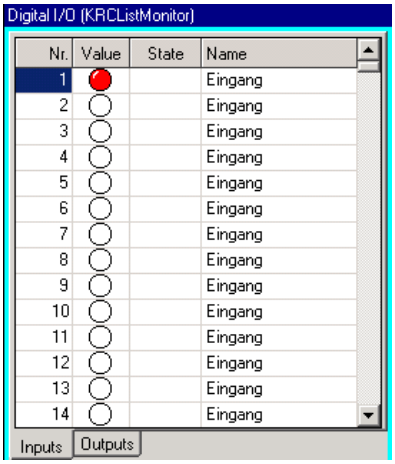


Fig. 61. - I/O KUKA.KSS 5.5 interface

4.2. Robotic System Settings

Once the communication between the robotic and welding system has been developed, two setting processes have to be achieved in order to obtain the proper function of the global system, this two setting tasks are:

1. Tool Calibration.
2. Definition of I/O signals in KUKA.ArcTech.

Tool calibration is the process of definition of the tool coordinate system, this coordinate system ensures a correct definition and measurement of the points in the program in relation with the tool used.

The I/O signals definition consists in the link between the signals originating from the welding power source and the control software, this definition is extremely needed in order to achieve a proper welding control of the process.

4.2.1. Tool Calibration

4.2.1.1 Coordinate Systems in KUKA KSS 5.5

The following Cartesian coordinate systems are defined in the robot system:

- WORLD (WCS)
The WORLD coordinate system is a permanently defined Cartesian coordinate system. It is the root coordinate system for the ROBROOT and BASE coordinate systems.

By default, the WORLD coordinate system is located at the robot base.

- ROBROOT (RCS)
The ROBROOT coordinate system is a Cartesian coordinate system, which is always located at the robot base. It defines the position of the robot relative to the WORLD coordinate system.

By default, the ROBROOT coordinate system is identical to the WORLD coordinate system. $ROBROOT allows the definition of an offset of the robot relative to the WORLD coordinate system.

- BASE (BCS)
The BASE coordinate system is a Cartesian coordinate system that defines the position of the work-piece. It is relative to the WORLD coordinate system.

By default, the BASE coordinate system is identical to the WORLD coordinate system. It is offset to the work-piece by the user.

- TOOL (TCS)

The TOOL coordinate system is a Cartesian coordinate system which is located at the tool centre point.

By default, the origin of the TOOL coordinate system is located at the flange centre point. (In this case it is called the FLANGE coordinate system) the TOOL coordinate system is offset to the tool centre point by the user.

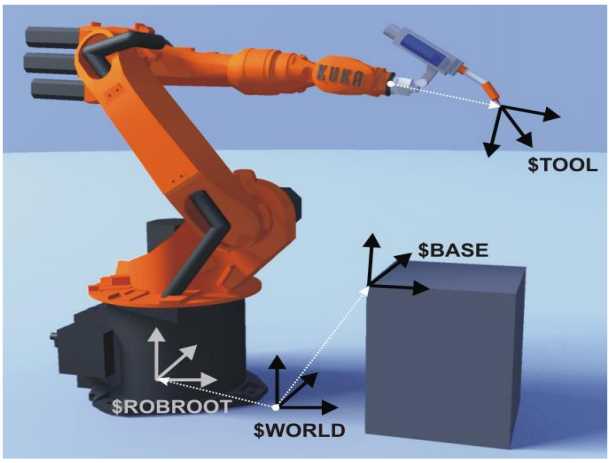


Fig. 62. - Overview of coordinate systems

4.2.1.2 Tool Calibration information

During tool calibration, the user assigns a Cartesian coordinate system (TOOL coordinate system) to the tool mounted on the mounting flange. The TOOL coordinate system has its origin at a user-defined point. This is called the TCP (Tool Centre Point). The TCP is generally situated at the working point of the tool.

Advantages of the tool calibration:

- The tool can be moved in a straight line in the tool direction.
- The tool can be rotated about the TCP without changing the position of the TCP.
- In program mode: The programmed velocity is maintained at the TCP along the path.

A maximum of 16 TOOL coordinate systems can be saved. Variable: TOOL_DATA[1…16].

The following data are saved:

- X, Y, Z: Origin of the TOOL coordinate system relative to the FLANGE coordinate system.
- A, B, C: Orientation of the TOOL coordinate system relative to the FLANGE coordinate system.

Tool calibration consists of 2 steps:

1. Definition of the origin of the Tool coordinate system. The following methods are available:

- XYZ 4-Point method: the TCP of the tool to be calibrated is moved to a reference point from 4 different directions. The reference point can be freely selected. The robot controller calculates the TCP from the different flange positions.
- XYZ Reference method: in the case of the XYZ Reference method, a new tool is calibrated with a tool that has already been calibrated. The robot controller compares the flange positions and calculates the TCP of the new tool.

2. Definition of the orientation of the Tool coordinate system. The following methods are available:

    - ABC World method**:** the axes of the TOOL coordinate system are aligned parallel to the axes of the WORLD coordinate system. This communicates the orientation of the TOOL coordinate system to the robot controller.
    - ABC 2-Point method: The axes of the TOOL coordinate system are communicated to the robot controller by moving to a point on the X axis and a point in the XY plane. This method is used if it is necessary to define the axis directions with particular precision.

The tool data can be entered manually from different sources:

- CAD.
- Externally calibrated tool.
- Tool manufacturer specifications.

4.2.2.3 Tool Calibration Development

In our application we have selected the XYZ 4-Point method for position and ABC 2-Point method for orientation.

The XYZ-4 Point method consists in the following steps:

1. Move the TCP to a reference point and measure it.
2. Move the TCP to the reference point from a different direction and measure it again, and repeat this step 4 times.
3. Save the information and select between the two available orientation methods.

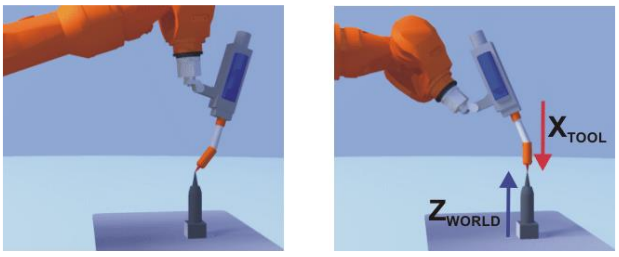


Fig. 63. - a,b Examples of movement to the reference point.

The ABC-2 Point method is formed by the following steps:

1. Move the TCP to any reference point and measure it.
2. Move the tool so that the reference point on the X axis has a negative X value (i.e. move against the tool direction) and measure it.
3. Move the tool so that the reference point in the XY plane has a negative Y value and measure this point.
4. Save the information in one of the 16 tool coordinate system available.

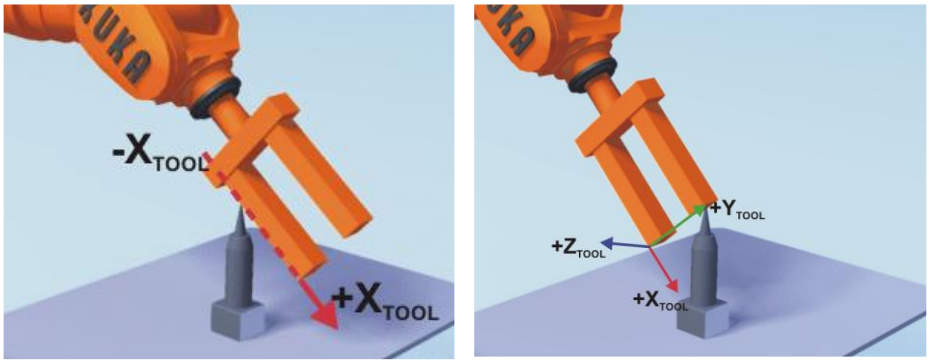


Fig. 64. - a,b Example of ABC 2-Point method

After applying this two method we have obtained a tool coordinate system, it is saved in the *config.dat* file of the robot controller with the name *welding_torch2* in position 3 of the 16 available.

The information obtained is the following:

```
TOOL_NAME[3]="welding_torch2"

TOOL_DATA[3]= {X -1.39300001,Y -2.78500009,Z 321.808014,A 61.0186996,B -36.0579987,C -
179.041}
```

4.2.2. I/O signals Setting of KUKA.ArcTech

4.2.2.1 Config.dat

The *config.dat* file is encompassed inside the system folder of the robot controller, it has all the information concerning the robot: coordinate system information, communication channel information, home points of the robot, limits of the six axes, maximum speed and acceleration of the six axes, socket information for the different ports of the robot, user defined global variables and, naturally, all the information of the different technology package that are installed in the robot.

The complete file is included in the section *A1.3* of the appendix.

4.2.2.2 ArcTech I/O declaration.

The declaration of the signals of ArcTech is deployed in the section ArcTech Digital globals of the *config.dat*. It is in this part where the connection with the respective Fronius signals has to be deployed. The lines of code that declare the variables and its respective connections are outlined as follows:

```
;FOLD ArcTech Outputs
DECL CTRL_OUT_T O_WELD_CTRL[15]
O_WELD_CTRL[1]={OUT_NR 0,INI FALSE,NAME_NAT[] "Start Welding       "}
O_WELD_CTRL[2]={OUT_NR 0,INI TRUE,NAME_NAT[] "Robot in Position   "}
O_WELD_CTRL[3]={OUT_NR 0,INI FALSE,NAME_NAT[] "Trigger Program Nr  "}
O_WELD_CTRL[4]={OUT_NR 0,INI FALSE,NAME_NAT[] "Acknowledge Errors  "}
O_WELD_CTRL[5]={OUT_NR 0,INI FALSE,NAME_NAT[] "Sneezer             "}
O_WELD_CTRL[6]={OUT_NR 0,INI FALSE,NAME_NAT[] "Cleaner Motor       "}
O_WELD_CTRL[7]={OUT_NR 0,INI FALSE,NAME_NAT[] "Sprayer             "}
O_WELD_CTRL[8]={OUT_NR 0,INI FALSE,NAME_NAT[] "                    "}
O_WELD_CTRL[9]={OUT_NR 0,INI FALSE,NAME_NAT[] "Gas purge          "}
O_WELD_CTRL[10]={OUT_NR 0,INI FALSE,NAME_NAT[] "Weld Error Signal  "}
O_WELD_CTRL[11]={OUT_NR 0,INI FALSE,NAME_NAT[] "                    "}
O_WELD_CTRL[12]={OUT_NR 0,INI FALSE,NAME_NAT[] "Target Point reached"}
O_WELD_CTRL[13]={OUT_NR 0,INI FALSE,NAME_NAT[] "Ignition ErrorSignal"}
O_WELD_CTRL[14]={OUT_NR 0,INI FALSE,NAME_NAT[] "WireFeed +          "}
O_WELD_CTRL[15]={OUT_NR 0,INI FALSE,NAME_NAT[] "WireFeed -          "}
;ENDFOLD (ArcTech Outputs)

;FOLD ArcTech Inputs
DECL CTRL_IN_T I_WELD_CTRL[15]
I_WELD_CTRL[1]={IN_NR 0,NAME_NAT[] "Power Supply Ready  "}
I_WELD_CTRL[2]={IN_NR 0,NAME_NAT[] "Current available   "}
I_WELD_CTRL[3]={IN_NR 0,NAME_NAT[] "Current over        "}
I_WELD_CTRL[4]={IN_NR 0,NAME_NAT[] "Wire available      "}
I_WELD_CTRL[5]={IN_NR 0,NAME_NAT[] "Water flow okay     "}
I_WELD_CTRL[6]={IN_NR 0,NAME_NAT[] "Key switch welding  "}
I_WELD_CTRL[7]={IN_NR 0,NAME_NAT[] "                    "}
I_WELD_CTRL[8]={IN_NR 0,NAME_NAT[] "                    "}
I_WELD_CTRL[9]={IN_NR 0,NAME_NAT[] "                    "}
I_WELD_CTRL[10]={IN_NR 0,NAME_NAT[] "                    "}
```

```
I_WELD_CTRL[11]={IN_NR 0,NAME_NAT[] "                    "}
I_WELD_CTRL[12]={IN_NR 0,NAME_NAT[] "                    "}
I_WELD_CTRL[13]={IN_NR 0,NAME_NAT[] "                    "}
I_WELD_CTRL[14]={IN_NR 0,NAME_NAT[] "                    "}
I_WELD_CTRL[15]={IN_NR 0,NAME_NAT[] "                    "}
;ENDFOLD (ArcTech Inputs)
```

As we can see, not all of the 15 control input and output are used or declared, this is because the ArcTech presents a great flexibility and capacity of extension so there are more than enough control variable in order to not restrict future expansions.

Taking a look on the code presented, is in the part *Out_Nr* (for output) and *IN_Nr* (for input) the place where the connection with the Fronius's signals must take place.

4.2.2.3 Final ArcTech I/O definition

The link process between robotic and welding signal system is centred in the correct connection of the signals, this seemed an easy step at the beginning but, because of the quantity inputs defined in the Fronius TPS 4000 and many of them in ambiguous ways; the process of the I/O customization has been very laborious.

After a long try-error process we could develop the correct customization of the signal in the *config.dat*, this final setting is presented in the following code lines.

```
;FOLD ArcTech Outputs
DECL CTRL_OUT_T O_WELD_CTRL[15]
O_WELD_CTRL[1]={OUT_NR 17,INI FALSE,NAME_NAT[] "Start Welding       "}
O_WELD_CTRL[2]={OUT_NR 18,INI TRUE,NAME_NAT[] "Robot in Position   "}
O_WELD_CTRL[3]={OUT_NR 0,INI FALSE,NAME_NAT[] "Trigger Program Nr  "}
O_WELD_CTRL[4]={OUT_NR 28,INI FALSE,NAME_NAT[] "Acknowledge Errors  "}
O_WELD_CTRL[5]={OUT_NR 30,INI FALSE,NAME_NAT[] "Sneezer             "}
O_WELD_CTRL[6]={OUT_NR 2,INI FALSE,NAME_NAT[] "Cleaner Motor       "}
O_WELD_CTRL[7]={OUT_NR 0,INI FALSE,NAME_NAT[] "Sprayer             "}
O_WELD_CTRL[8]={OUT_NR 0,INI FALSE,NAME_NAT[] "                    "}
O_WELD_CTRL[9]={OUT_NR 25,INI FALSE,NAME_NAT[] "Gas purge          "}
O_WELD_CTRL[10]={OUT_NR 0,INI FALSE,NAME_NAT[] "Weld Error Signal  "}
O_WELD_CTRL[11]={OUT_NR 0,INI FALSE,NAME_NAT[] "                    "}
O_WELD_CTRL[12]={OUT_NR 0,INI FALSE,NAME_NAT[] "Target Point reached"}
O_WELD_CTRL[13]={OUT_NR 0,INI FALSE,NAME_NAT[] "Ignition ErrorSignal"}
```

```
O_WELD_CTRL[14]={OUT_NR 26,INI FALSE,NAME_NAT[] "WireFeed +          "}
O_WELD_CTRL[15]={OUT_NR 27,INI FALSE,NAME_NAT[] "WireFeed -          "}
;ENDFOLD (ArcTech Outputs)

;FOLD ArcTech Inputs
DECL CTRL_IN_T I_WELD_CTRL[15]
I_WELD_CTRL[1]={IN_NR 23,NAME_NAT[] "Power Supply Ready  "}
I_WELD_CTRL[2]={IN_NR 20,NAME_NAT[] "Current available   "}
I_WELD_CTRL[3]={IN_NR 48,NAME_NAT[] "Current over        "}
I_WELD_CTRL[4]={IN_NR 0,NAME_NAT[] "Wire available      "}
I_WELD_CTRL[5]={IN_NR 0,NAME_NAT[] "Water flow okay     "}
I_WELD_CTRL[6]={IN_NR 19,NAME_NAT[] "Key switch welding  "}
I_WELD_CTRL[7]={IN_NR 0,NAME_NAT[] "                     "}
I_WELD_CTRL[8]={IN_NR 0,NAME_NAT[] "                     "}
I_WELD_CTRL[9]={IN_NR 0,NAME_NAT[] "                     "}
I_WELD_CTRL[10]={IN_NR 0,NAME_NAT[] "                     "}
I_WELD_CTRL[11]={IN_NR 0,NAME_NAT[] "                     "}
I_WELD_CTRL[12]={IN_NR 0,NAME_NAT[] "                     "}
I_WELD_CTRL[13]={IN_NR 0,NAME_NAT[] "                     "}
I_WELD_CTRL[14]={IN_NR 0,NAME_NAT[] "                     "}
I_WELD_CTRL[15]={IN_NR 0,NAME_NAT[] "                     "}
;ENDFOLD (ArcTech Inputs)
```

The numbers of the inputs are outputs corresponds with the Fronius's signal definition (exposed in section 4.2.2.4).

Following the established in the previous section, There are many signals that are not used in our application, this is because our goal is not designed in order to create a complete a complex welding machine, in fact we have presented briefly the welding system and focused on the vision and robotic system, in a future if new goals about welding are defined the process of signal customization will have to be boarded again.

4.3. Welding System settings

As well as the robotic system, some setting work has to be developed in the welding system in order to achieve the desired results in welding characteristic and accuracy. First, the installation of a ground connection for welding and after, an experiment to determine the proper welding parameters have been performed.

4.3.1. Welding Ground Connection

The welding ground connection has been deployed with a cable of 25 mm of section to ensure the correct circulation of the current avoiding the undesired high temperatures in the cable.

The connection to the ground has been fixed in the platform of the robot in the same point where the power source ground had been connected previously.

Fig. 65. - a,b Welding Ground Connection

4.3.2 Power Source Parameter Customization

An experiment have been deployed in the project in order to enclose the proper welding parameter values that have to be selected in the power source.

In that experiment we have created a total of 9 jobs changing the thickness parameter available in the Fronius TPS 4000 in order to determine the quality of the welds attending to this parameter. The Job Nº1 starts with a material thickness of 1mm and gradually this parameter is increased in the different jobs until it achieve its maximum value of 5mm in JOB Nº9. Some of the parameters changes automatically in conjunction with the thickness parameter ensuring in that way a correct customization for every thickness selected.

However, there are a list of parameters that have remained without changes in all the jobs:

- Diameter of the welding wire: 0.8mm.
- Type of the welding Gas: G3/4 Si 1 (Ar-82%, CO_2-18%).
- Gas pre-flow time: 0.1 s.
- Gas post-flow: 0.5 s.
- Feeder creep: Aut.

- Feeder inching: 10 m/min.
- Burn-back time correction: -20s.
- Arc length correction: 0.
- Arc-force (dynamic) correction: 0.

The experiment has been deployed on a work-piece of 5 mm thick.

4.3.2.1 JOB Nº1

Parameters of JOB Nº1:

- A dimension parameter: 2.2º.
- Thickness: 1 mm.
- Welding Current: 51 A.
- Wire-feed speed:2.8 m/min.
- Welding Voltage: 15.8 V.
- Welding speed: 30 cm/min.

Fig. 66. - JOB Nº 1

4.3.2.2 JOB Nº2

Parameters of JOB Nº2:

- A dimension parameter: 3.1º.
- Thickness: 1.5 mm.
- Welding Current: 99 A.
- Wire-feed speed: 5.9 m/min.
- Welding Voltage: 17.1 V.
- Welding speed: 31 cm/min.

Fig. 67. - JOB Nº 2

4.3.2.3 JOB Nº3

Parameters of JOB Nº3:

- A dimension parameter: 3.2º.
- Thickness: 2mm.
- Welding Current: 110 A
- Wire-feed speed: 6.5 m/min.
- Welding Voltage: 17.9 V.
- Welding speed: 32 cm/min

Fig. 68. - JOB Nº 3

4.3.2.4 JOB Nº4

Parameters of JOB Nº4:

- A dimension parameter: 3.7º.
- Thickness: 2.5 mm.
- Welding Current: 135 A.
- Wire-feed speed: 8.7 m/min.

- Welding Voltage: 19 V.
- Welding speed: 32 cm/min.

Fig. 69. - JOB Nº 4

4.3.2.5 JOB Nº5

Parameters of JOB Nº5:

- A dimension parameter: 4.1º.
- Thickness: 3 mm.
- Welding Current: 161 A.
- Wire-feed speed: 11 m/min.
- Welding Voltage: 20.3 V.
- Welding speed: 33 cm/min.

Fig. 70. - JOB Nº 5

4.3.2.6 JOB Nº6

Parameters of JOB Nº6:

- A dimension parameter: 4.3º.
- Thickness: 3.5 mm.
- Welding Current: 168 A.
- Wire-feed speed: 13.4 m/min.
- Welding Voltage: 21.7 V.

- Welding speed: 33 cm/min.

Fig. 71. - JOB Nº6

4.3.2.7 JOB Nº7

Parameters of JOB Nº7:

- A dimension parameter: 4.5º.
- Thickness: 4 mm.
- Welding Current: 175 A.
- Wire-feed speed: 14 m/min.
- Welding Voltage: 23.4 V.
- Welding speed: 34 cm/min.

Fig. 72. - JOB Nº 7

4.3.2.8 JOB Nº8

Parameters of JOB Nº8:

- A dimension parameter: 4.8º
- Thickness: 4.5 mm.
- Welding Current: 181 A.
- Wire-feed speed: 15.4 m/min.
- Welding Voltage: 25.1 V.
- Welding speed: 34 cm/min.

Fig. 73. - JOB Nº 8

4.3.2.9 JOB Nº9

Parameters of JOB Nº9:

- A dimension parameter: 4.9º
- Thickness: 5 mm.
- Welding Current: 187 A.
- Wirefeed speed: 16.7 m/min.
- Welding Voltage: 26.9 V.
- Welding speed: 35 cm/min.

Fig. 74. - JOB Nº 9

As a result, after a quick observation of the welding result obtained in every job, the most suitable thickness for this work-piece is the selected in the JOB Nº8 4.5 mm. This result could present some ambiguity because the work-piece is 5mm thick instead of 4.5 mm. This difference could be attributed to the some difference between the material of the work-piece selected in the power source and the real material used. And also because of the section of the welding wire which was wider than the recommended specificarions

The KRL code developed to perform the experiment is attached in the appendix section *A.3.1*

Chapter 5.

5. Initial Welding Applications

Once the robotic and welding system are correct set-up and the communication has been properly developed it is possible to start the creation of some example programs of welding in order to present the initial possibilities of the system. Besides, before outline this examples some information about how the program are created and deployed must be included.

5.1. KRL basic program structure

```
1  DEF my_program( )
2  INI
3
4  PTP HOME  Vel= 100 % DEFAULT
   ...
8  LIN point_5 CONT Vel= 2 m/s CPDAT1 Tool[3] Base[4]
   ...
14 PTP point_1 CONT Vel= 100 % PDAT1 Tool[3] Base[4]
   ...
20 PTP HOME  Vel= 100 % DEFAULT
21
22 END
```

Fig. 75. - Basic KRL program

In the previous figure an example of a KRL program is present, a description of every code line of it is outlined as follows:

1. *Def line*. The Def line indicates the name of the program. If the program is a function, the Def line begins with 'Def' and contains additional information.

2. *Ini line*. The Ini line contains initializations for internal variables and parameters.
This line must not be deleted.

4. *Home position*. The Home position is not program-specific. It is generally used as the first and last position in the program as it is uniquely defined and uncritical. The Home position is stored by default in the robot controller.

22. *End line.* The End line is the last line in any program. If the program is a function, the wording of the End line is 'End'.

5.2. Base Coordinate System

Following the statements established in the Coordinate System information (exposed in section 4.3.1.1) a base coordinate system is defined coincident with world coordinate system by default. In order to change or to attach it to any work-piece a new base calibration has to be done.

During base calibration, the user assigns a Cartesian coordinate system
(BASE coordinate system) to a work surface or the work-piece. The BASE coordinate system has its origin at a user-defined point. The advantages of base calibration are:

- The TCP can be jogged along the edges of the work surface or work-piece.
- Points can be taught relative to the base. If it is necessary to offset the base, e.g. because the work surface has been offset, the points move with it and do not need to be retaught.

A maximum of 32 BASE coordinate systems can be saved. Variable:
BASE_DATA[1…32].

There are 3 ways of calibrating a base:

- 3-point method.
- Indirect method.
- Introduce numerical data.

In our application base coordinate system plays an important role. In fact all the points taught are in respect a base coordinate system, this base is given by the vision system every time it matches the orientation and position of the cylinder. In that way, we ensure a correct positioning of the points whatever the position and orientation of the work-piece is.

To sum up the process, the vision system determine the coordinate system of the piece, this data is homogenously transformed and sent to the robot controller that introduce it as a base coordinate system numerically and then, the pre-taught points do not need to be re-programmed every time.

5.3. Available motion programming in ArcTech

ArcTech digital package supports three different motion in order to achieve welding between two points, this three movements are PTP, LIN and CIRC.

5.3.1. PTP

The robot guides the TCP along the fastest path to the end point. The fastest path is generally not the shortest path and is thus not a straight line. As the motions of the robot axes are rotational, curved paths can be executed faster than straight paths. The exact path of the motion cannot be predicted.

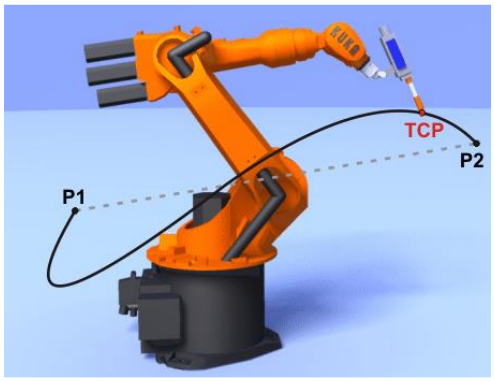


Fig. 76. - PTP motion

5.3.2. LIN

The robot guides the TCP at a defined velocity along the shortest path to the end point. The shortest path is always a straight line.

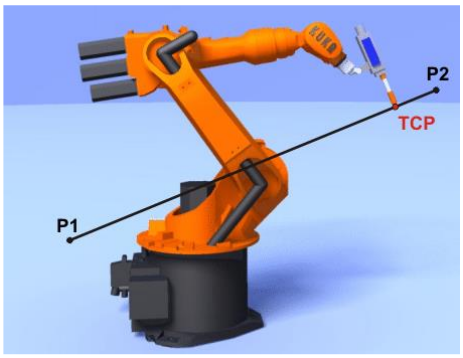


Fig. 77. - LIN motion

5.3.3. CIRC

The robot guides the TCP at a defined velocity along a circular path to the end point. The circular path is defined by a start point, auxiliary point and end point. It uses an additional point apart from the start or the end (P aux) in order to give information of the robot controller the arc that the TCP has to realise.

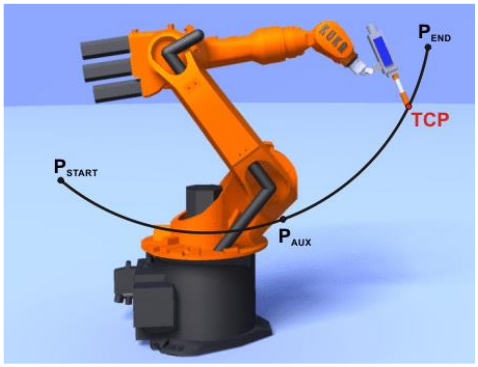


Fig. 78. - CIRC motion

5.4. KUKA.ArcTech Digital programming features

The main features used by the KUKA.ArcTech are INI, ARC ON, ARC OFF and ARC SWITCH. This three terms ensures the creation of almost any welding application.

The flowcharts of the three terms are encompassed in the section *A2.1*, A2.2 and *A2.3* of the Appendix

5.4.1. INI

This command is automatically added at the beginning of every new program created. This command sets the basic parameters for the robot and the application and initialises the welding and motion commands and approximate positioning.

5.4.2. ARC ON

The command "*ARC ON*" contains the parameters for moving the welding torch (type of motion, velocity, etc.) from the home position to the start point of the seam, the start parameters (start delay) and the program number.

While the *"ARC ON"* program phase is being executed, the system scans the peripheral signal *"I_WELD_COND"* to check whether the welding controller is ready. When the welding torch reaches the ignition position, arc ignition is enabled by means of the signal *"O_WELD_START[ ]".*

When the arc has been struck, the welding power source supplies the signal *"I_START_MOVE[]*", as a result of which the robot starts to move in accordance with the programmed path and velocity. The signal "*O_ACK_START[ ]"* informs the welding controller that the robot is moving.

The signals previously commented are directly linked with signals from the Fronius power source.

The movement from the home position to the start point of the seam can be executed as a "PTP", "LIN" or "CIRC" motions.

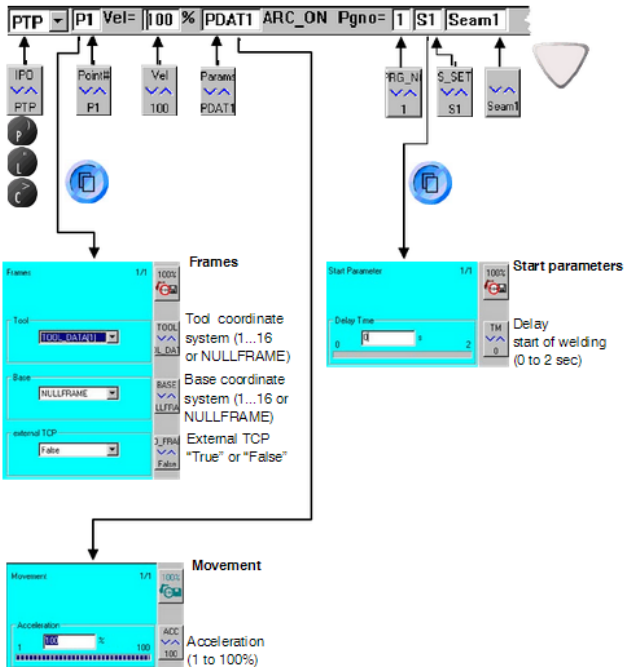


Fig. 79. - Inline form for ARC ON in PTP motion

5.4.3. ARC OFF

The welding command “ARC OFF” contains the program number for the power source, the motion parameters and, if relevant, the mechanical weaving parameters used for a single seam from a weld start (ARC ON) to the end of the seam, and also parameters for crater filling. A single seam therefore requires at least two commands, namely “ARC ON” and “ARC SWITCH” or “ARC OFF”.

If a seam consists of several seam sections with different motion and/or weld parameters, the command “ARC OFF” is used for the last seam section.

Motions from the ignition point (ARC ON), or in case of several seam sections from the target point of the last section of an “ARC SWITCH” command to the end point of the seam, can be “LIN” or “CIRC” motions.

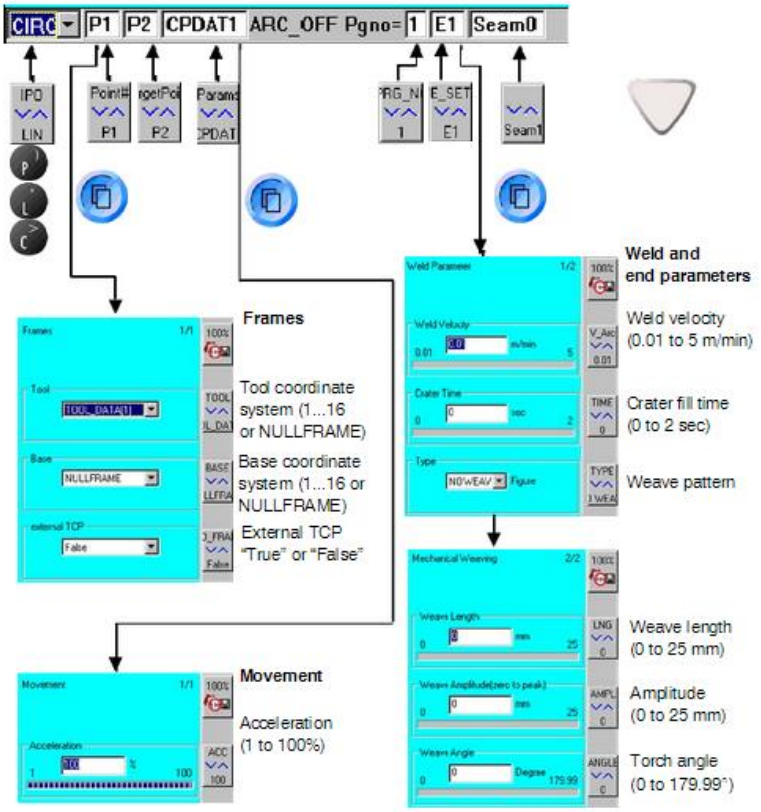


Fig. 80. - Inline form for ARC OFF in CIRC motion

5.4.4. ARC SWITCH

The command “ARC” (shown as “ARC SWITCH” in the menu) is used between the commands “ARC ON” and “ARC OFF” when the seam is divided into several sections with different motion and/or weld parameters.

ARC SWITCH contains the program number, the motion parameters for the current section of the seam, and also the parameters for the weld velocity and the mechanical weaving for the current section of the seam.

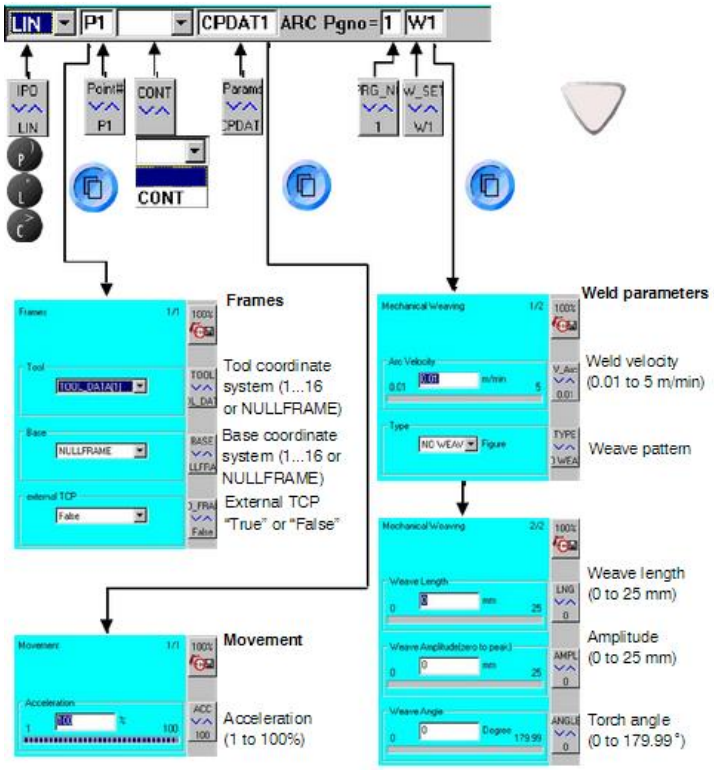


Fig. 81. - Inline form for ARC SWITCH in LIN motion

5.5. ArcTech Program Structure

Although ArcTech Digital is a flexible tool in order to create almost any welding application reachable in industrial robotics environments, the programmes created with this technology has to follow a determined structure. This is caused because of some parameters must be set with specific values at the beginning and end of the program, also when ARC SWITCH is programmed re-adjusting of parameters must be taken. The structure is outlined in the next figure:

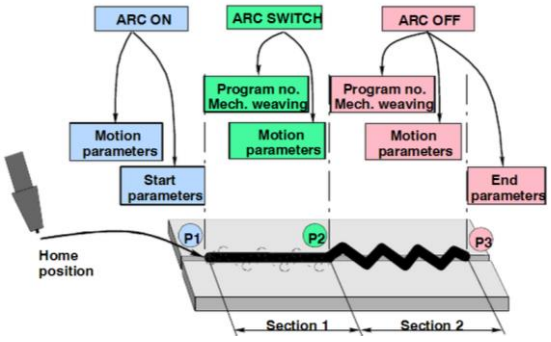


Fig. 82. - Program structure of ArcTechDigital

5.6. Initial programmes

Once all the necessary information has been outlined it is possible to present two examples of welding applications with determined position of the pieces. This two examples follow a similar patron of welding. The pieces used are a cylinder and a special piece formed by different shapes.

5.6.1. Cylinder

Following the statements outlined in the abstract of this paper the objective of the project is going to be achieved through the recognition, position detection and welding of a cylinder, in this section the robot program developed are outlined in order to weld the base of the cylinder into a flat square piece. In this program, all the points are respect a specific base coordinate system situated in the centre base of the cylinder.

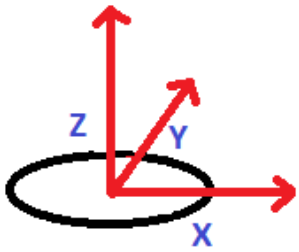


Fig. 83. - Base Reference System defined

As we have established before, this base will be sent by the vision system every time a new cylinder matching is achieved.

The dimension of the cylinder are:

- 48 mm of diameter.
- 48 mm of height.
- 3 mm of thickness.

The flat piece has the dimensions 100x100x5 mm.

5.6.1.1 Process

The welding process is divided into four parts in order to avoid the thermal fluctuations that appeared if the circular weld would be performed continuously. All the four parts starts from the same point (P1) in the centre of the cylinder but with bigger Z coordinate value. Besides, at beginning and at the end of the program the weld torch moves from or to the HOME position where all the applications have been initialised.

The process of welding is presented in the following diagram.

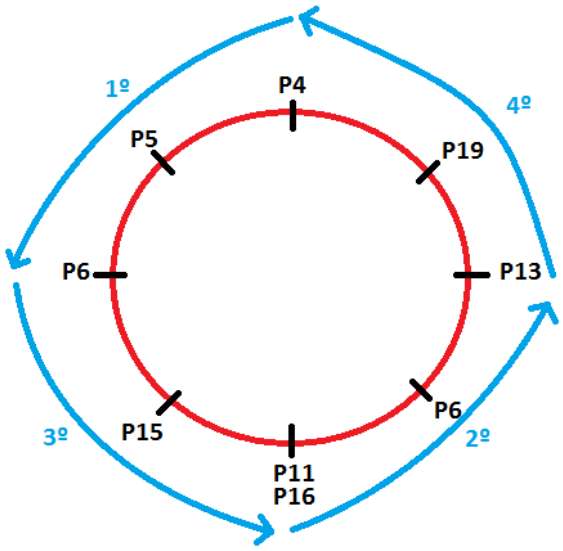


Fig. 84. - Diagram of the cylinder welding process

As we can see, the process is quite simple, we have only to care about of three issues:

1. In order to maintain the orientation of the weld torch, the option “HAND PTP” in the CIRC command inline has to be selected, this option causes the set of the variable *$ORI_TYPE* to *#CONSTANT*, that makes constant the orientation of the TCP during the circular movement.

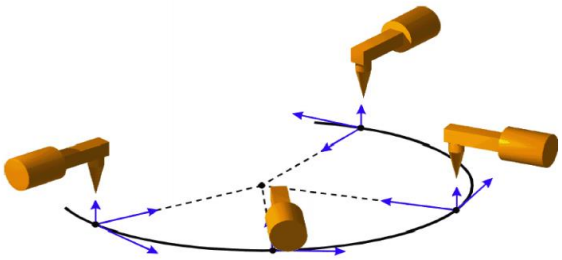


Fig. 85. - Constant orientation of the TCP during a CIRC command.

2. The position of the axis of the robot has to be took into account. In order to achieve a proper constant orientation during the circular movement. To ensure that, the 6th axis of the robot has to be parallel aligned with the base of the cylinder.

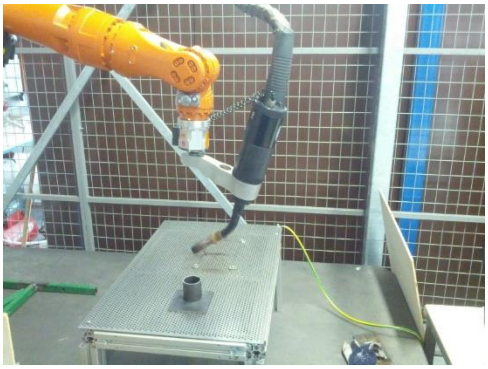


Fig. 86. - Parallel alignment between the base of the cylinder and the 6º axis in P1

3. All the installation of the weld system in the robot makes not possible to realise a complete circle continuously, the weld torch limits the movement of the robot. Because of that in the nearest point of the robot to weld, the weld torch has to approximate it in two different orientations (P11 and P16).

Fig. 87. - Weld torch in points P11 (a) and P16 (b)

5.6.1.2 Result obtained

The job used in the welding power source has been JOB Nº7 because of the cylinder is thinner than the lap piece used in the welding parameters experiment explained in the point 4.4.2.

Fig. 88. - Welding Result in the circular joint

As we can see in the previous image, the weld result is quite improvable, but following the statements done in the introduction of the project, this work is not focus on obtain high quality welds.

The KRL code deployed for this application is attached in the appendix section *A3.2***.**

5.6.2. Special piece.

The second example is focused on the achievement of realise four different types of welds to demonstrate an example of the potential of the welding cell. The piece is formed by:

- 1 Cylinder (same dimensions that the used in the first example).
- 2 L Profiles (height 50mm, length 50 mm, thickness 5 mm) .
- 2 Square flat pieces (100 mm, thickness 5 mm).

5.6.2.1 Process

The process of this example is divided into four different sections:

1. Straight lap joint that fixes the square pieces.
2. Circular joint between the cylinder and the square flat piece.
3. Square joint between the two L profiles and the square piece.
4. Vertical joints between the two L profiles.

- Straight joint.

The first weld is the easiest and simplest, it is carried out by only a linear movement between two points (P4 and P5). The only issue is the approximation to the joint, because of the cylinder and the two L profiles limit the approximation envelope available to the robot.

The Job used in this welding is the JOB Nº8.

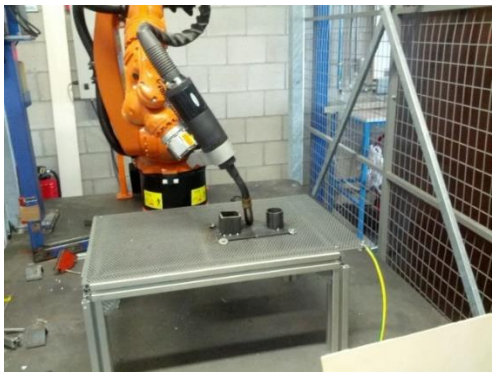


Fig. 89. - Central straight joint (simulation of welding)

- Circular Joint

The circular joint that attaches the cylinder and the square piece follows precisely the same process explained in the first welding application, the only difference the way of approximation to the joint to avoid crashes with other parts of the piece.

- Square Joint

The square welding are also simple, the only issue as well as in the circular joint is the obligation of perform the weld in different sections in order to avoid thermal diffusions of the material.

The job used in the welding power source is the JOB Nº8.

In the following diagram there are outlined the order of the sectors to weld.

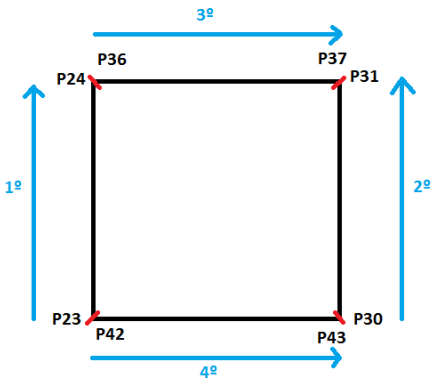


Fig. 91. - Diagram of the square welding process

Fig. 90. - TCP in point P36 in welding simulation

- Verticals Joints

The two vertical joints are the most difficult joint to weld, there are two problem that have to be solved:

1. Because of the L profiles have rounded edges, there is hardly material in the joints.
2. Vertical joint are one of the most difficult kind of weld, because of the filler material flow and it difficulties the creation a constant and homogeneous welding.

The solution adopted is based on a weld with low speed of the filler metal from the highest part of the joint to the bottom and with a big triangular wave in the weld torch to achieve the adhesion of the filler metal in the extremes of the beam. This wave movement is done through the selection of wave and wave type in the inline form of the ARC OFF command. The job mode used in the power source has been JOB Nº6.

Fig. 93. - Diagram of one vertical weld

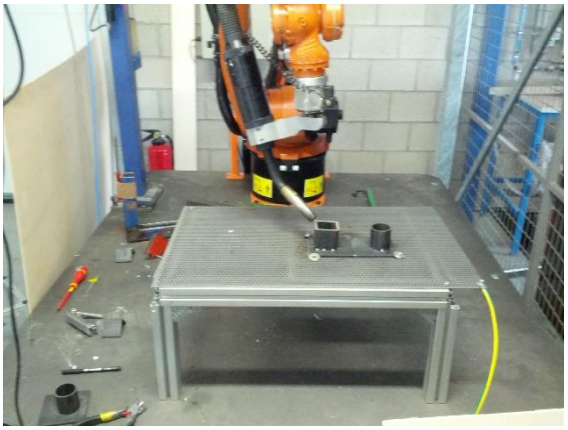


Fig. 92. - TCP in point P49 of one of the vertical joints (welding simulation mode)

5.6.2.2. Results obtained

The result obtained is quite acceptable to the purpose of the experiment, the piece does not need high specification in mechanic characteristics and it is going to be used only as a demonstration.

Fig. 94. - Final Result of special piece

The KRL code is attached to the section *A3.3* of the appendix.

In order to improve the welding obtained, some welding techniques would be applied, but as well as in the cylinder high welding features are not the objective so there is no need of wasting a lot of time in obtaining high quality welds.

5.7. Welding Torch Cleaning Application

As seen in section 3 of this paper, inside this project there is a complete welding torch cleaning station. Here we are going to comment the work develop in order to create a routine application to clean the welding gun after finishing welding applications.

The process of the application is divided into three parts:

1. Wire cutting.
2. Cleaning the inside of the weld torch.
3. Lubrication.

During this procedure, the advance of the robot is set to 0 in order to achieve a correct synchronization of the I/O pulse and the robot motions. If the advance is not set to 0 the I/O pulses are executed before how it were sequentially programmed, this happens because the robot controller computes the code line before the execution of the program has reached it.

The KRL code is outlined in the appendix section *A3.4.*

5.7.1. Wire Cutting

The cutting action is the first stage of the process. It is developed through the activation of the output 3 and teaching previously the positioning of the weld torch in order to get a correct stick out parameter. We have to outline that the output 26 “wire +” is activated by a pulse just before the activation of the output 3. This is done to ensure that always is wire to cut.

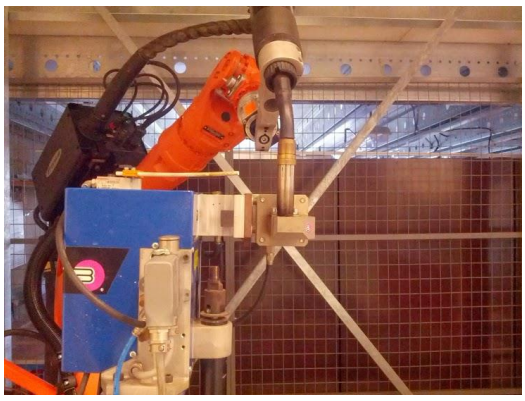


Fig. 95. - Stage 1.Wire cutting

5.7.2. Cleaning the inside of the weld torch

The cleaning action is done though a spin action of a cylinder. The process in this stage is actually simple.

First the weld torch approximates to the clean point, afterwards the piston fastens it and the cylinder starts to spin (through activation of output 2). Then the cylinder raises and contacts with the inside of the weld torch (through activation of output 3). Once the cylinder has remained spinning in this position during 3 seconds, it descents to its initial point (deactivating output 3). Finally, the piston releases the weld torch, the spinning action is finished (deactivating output 2) and the clean stage is finish.

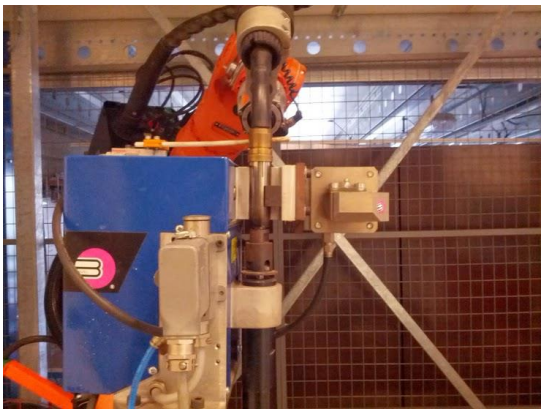


Fig. 96. - Stage 2. Inside cleaning

5.7.3 Lubrication

This stage is the simplest one, the weld torch approximates to the lubrication point and in this approximation the weld torch push a bar that activates the flow of air and oil. The weld torch remains straighten with the orifice where the lubricant flows during two seconds, afterwards it goes back to its initial position in this stage.

Fig. 97. - Stage 3. Lubrication

5.7.4 Motion from the welding-zone to the cleaning-zone and vice versa

Apart from this three stages, there are two additional motion steps in the application: the approximation from the work zone to the clean zone at the beginning of the application and the inverse movement after the end of the three processes. This motions are achieved with the movement of the welding tool through two points (P1 and P2) and the home position stored for the welding applications.

Chapter 6.

6. Vision System

In this section there are described all the processes developed in order to achieve a correct matching of the cylinder in different positions and orientations in the field view of the camera.

At first, the different methods available in HALCON to 3D recognition are briefly described, afterwards, there are explained the fundamentals of camera calibration.

Finally, after the correct description of the 3D Shape Based Matching and the communication between the vision computer and the robot controller, the final tool obtained is outlined and some final results are showed up.

6.1. Different methods of 3D object position recognition

Estimating the 3D pose of an object is an important task in many application areas, for example, during completeness checks or for 3D alignment in robot vision applications. In HALCON there are available multiple methods in order to obtain the total 3D position or poses of known objects. This methods are: Pose Estimation from Points, 3D Matching, 3D Primitive Fitting, Perspective Matching and Circular or Rectangular Pose.

A 3D pose is nothing more than an easier-to-understand representation of a rigid transformation: Instead of the 12 elements of the homogeneous transformation matrix, a pose describes the rigid transformation with 6 parameters, 3 for the rotation and 3 for the translation: (TransX, TransY, TransZ, RotX, RotY, RotZ). The main principle behind poses is that even a rotation around an arbitrary axis can always be represented by a sequence of three rotations around the axes of a coordinate system.

6.1.1. Pose estimation from points

The most general approach, that does not use a CAD model, determines the pose of a known 3D object using at least three corresponding points such as points with known 3D object coordinates for which the corresponding image coordinates are extracted. The approach is also known as “mono 3D”.

6.1.2. 3D Matching

If a model of a known 3D object is available, 3D matching can be applied to locate the object. The available 3D matching approaches perform a full 3D object recognition, they not only estimate a pose but first locate the object in the respective search data:

- Shape-based 3D matching can be used to locate a complex 3D object in a single 2D image. The model of the 3D object must be available as a Computer Aided Design (CAD) model in DXF, OFF, or STL format and the object needs "hard geometric edges" to be recognized. This method has been the selected in order to apply in our project, this election is based on the great properties of 3D shape based being the best method using a 2D camera to detect position and orientation in 3D objects. Further information about it will be given in the following sections.
- Surface-based 3D matching can be used to quickly locate a complex 3D object in a 3D scene, i.e., in a set of 3D points that is available as a so-called 3D object model. The model of the 3D object must be available also as a 3D object model and can be obtained either from a CAD model in DXF, OFF, or PLY format or from a reference 3D scene that is obtained by a 3D reconstruction approach, e.g., stereo or sheet of light. Here, the object may also consist of a "smooth surface". Note that this approach is also known as "volume matching".

6.1.3. 3D primitive fitting

If the poses of simple 3D shapes like cylinders, spheres, or planes, which are called "3D primitives", are searched in a 3D scene that is available as a 3D object model, the 3D primitives fitting can be used. There, the 3D scene is segmented into sub-parts so that into each sub-part a primitive of a selected type can be fitted. For each sub-part the fitting returns the parameters of the best fitting primitive, for example, the pose for a fitted plane.

6.1.4. Perspective matching

Sometimes, a full 3D object recognition or 3D matching is not necessary because you can estimate the pose of the object with simpler means. For example, if the object contains a characteristic planar part, you can estimate its 3D pose from a single image using perspective matching. Similar to the 3D matching approaches, they locate the object before they estimate its pose. Two approaches are available:

- The calibrated perspective deformable matching determines the 3D pose of a planar object that was defined by a template object by using automatically derived contours of the object.
- The calibrated descriptor-based matching determines the 3D pose of a planar object that was defined by a template object by using automatically derived distinctive points of the object, which are called "interest points".

6.1.5. Circular or rectangle pose

If a circle or rectangle is contained in the plane for which the 3D pose is needed, the pose estimation can be applied also by a simple circle pose or rectangle pose estimation. There, the circle or rectangle must be extracted from the image and the internal camera parameters as well as the dimensions of the circle or rectangle must be known.

6.2. Camera Calibration

In HALCON it is easy to obtain undistorted measurements in world coordinates from images. In general, this can only be done if two or more images of the same object are taken at the same time with cameras at different spatial positions, this is the so-called stereo approach.

However, in industrial inspection, there often possibilities to have one camera available and time constraints to use stereo system. But HALCON offers a list of techniques that allow to relate the camera coordinate system with the world coordinate system. This is done thanks to the camera calibration, it obtains the external camera parameters (relationship between both coordinate system) and the internal camera parameters (characteristic of every camera).

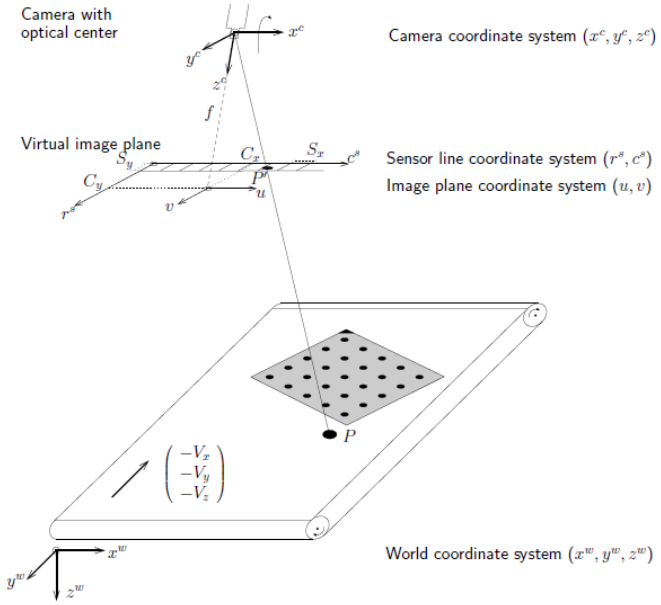


Fig. 98. - Example of a Calibration in a Line Scan Camera

6.2.1. Calibration Plate

The simplest method to determine the camera parameters of a CCD camera is to use the standardised calibration plates. In this case, the whole process of finding the calibration plate, extracting the calibration marks, and determining the correspondences between the extracted

calibration marks and the respective 3D world coordinates can be carried out automatically with the HDevelop Calibration Assistant,process described in next sections of this paper.

The calibration plates are available in different materials (ceramics for front light and glass for back light applications) and sizes (e.g., 0.65 × 0.65mm2, 10 × 10mm2, 200 × 200mm2). Each calibration plate comes with a description file with the extension "*.*descr*," in this file the information about the marks, rows, columns and size of the calibration plate are outlined. Once the description file has been given to Halcon, then, it is possible to use the name directly in the operator *caltab_points*.

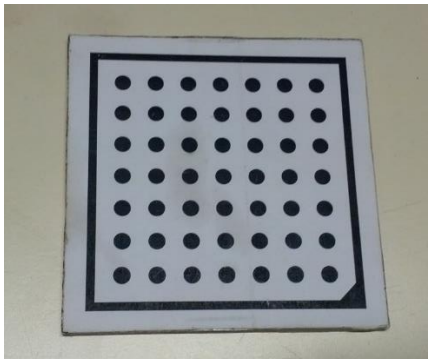


Fig. 99. - Calibration Plate Example (70 mm)

6.2.1.1 Recommendations for Acquiring Calibration Images

Some recommendations are attached in order to achieve an accurate calibration:

- A calibration plate should cover at very least the ninth part of the image area. To improve the calibration quality, it recommended a plate that covers at least a quarter of the image area.
- Use a clean calibration plate.
- Cover the whole field of view with multiple images, for example, place the calibration plate in all areas of the field of view at least once.
- Vary the orientations of the calibration plate. This includes rotations around the x- and y-axis of the calibration plate, such that the perspective distortions of the calibration pattern are clearly visible. Without some tilted calibration plates the focal length cannot be calculated properly (a tilt angle of approximately 45 degrees is recommended).
- Use at least 10 – 15 images.
- Use an illumination where the background is darker than the calibration plate.
- The bright parts of the calibration plate should have a grey value of at least 100.
- The contrast between the bright and the dark parts of the calibration plate should be more than 100 grey values.
- Use an illumination where the calibration plate is homogeneous.
- The images should not be overexposed (the grey values of the bright parts of the image should be strictly below 255).
- The diameter of a circle should be at least 10 pixels.
- The calibration plate should be completely visible inside the image.

- The images should contain as little noise as possible.
- The images should be sharply focused, i.e., transitions between objects should be clearly delimited.

6.2.2. Perform the Calibration

The calibration process in HALCON consists of three steps:

- Preparation.
- Calibration.
- Accessing the results.

After accessing the results, they can be stored and then release the memory of the calibration data model.

6.2.2.1 Pre-Calibration Preparation

Before the calibration takes place there a list of parameters and tasks have to be determined and given to the calibrator operators, they are:

- Create the model and specify basic information such as the number of cameras to calibrate.
- Specify initial values for the internal camera parameters.
- Describe the calibration object.
- Observe the calibration object in multiple poses (images) and store the extracted information, and optionally restrict the calibration to certain parameters, keeping the others fixed.

6.2.2.2 Performing the Actual Calibration

The Calibration is carried out by the operator *calibrate_camera* that receive the information of the calibration parameters determined in the previous phase.

6.2.2.3 Accessing the Results of the Calibration

Once the calibration has been done its result can be stored, in particular, the most interesting data to be stored are the calibrated camera parameters and poses of the calibration. There is also the possibility to access of this data after the calibration using the operator *get_calib_data*.

6.2.2.4 Example of Camera Calibration

In order to clarify some concepts of camera calibration a little example of this application has been deployed, it has been done in a Ueye camera and with a 70mm calibration plate.

The code of the program can be sub-divided into three stages. In the first one all the required initialization parameters are obtained. The second stage is formed by a *for* loop with 10 iterations which in every of them an image of the calibration plate is acquired and the calibration data model is found. Finally in the third stage, the calibration is carried out and stored, besides the acquirement device connection is finished and the application is finish.

The code deployed in HDevelop is outlined in the section *A4.1* of the appendix**.**

- First Stage

The first thing to do is to disconnect any acquirement device that could remain connected to the HDevelop, afterwards the initial parameters are obtained, the information about the calibration plate is read from the file *"calta_70mm.descr"*, the initial camera parameters obtained from the reference manual of the Ueye camera are assigned to the array *StartCamPar*. After that, an initial calibration data model is created and the parameters from the calibration plate and the camera are set to it.

Afterwards, and as a final process of this stage the acquirement device connection is established and with the camera information about the height and width of the camera the result window is created.

- Second Stage

The second stage is delimited by a *for* loop with 10 repetitions. In every iteration an image of the calibration plate is acquired and with the operator *find_calib_object* the calibration data model is searched in the image. Once the object has been founded the contour of the found object and the marks of it are superimposed in the image in order to check if the result obtained is correct.

Afterwards, the application is paused in order to let the user change the position of the calibration respect the camera to acquire a new image and start again the process.

- Third Stage

Inn the third stage, the calibration result is carried out with the previously obtained calibration data model. This is done through the use of the operator *calibrate_cameras*. Afterwards, the camera parameters are obtained from the calibration data model with the operator *get_calib_data* and it are stored in the file *"Ueye param.dat"*.

Finally the data calibration model is detached and the acquirement device connection is ended.

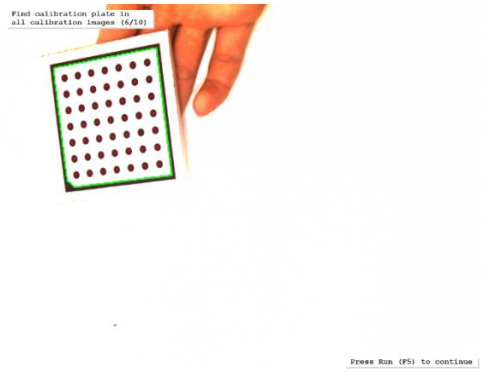


Fig. 100. – Calibration plate recognition

6.2.3 Hdevelop Calibration Assistant

Halcon offers many assistants in order to make easier some typical procedures that are carried out in all the vision solutions developed. Calibration is one of this procedures, with the Calibration Assistant the process of Calibration is summed up in only three simple steps all done in it.

- Introduction of the camera and calibration plate information.
- Acquirement of the images.
- Obtain and save the calibration results.

The real potential of this assistant is the capacity of seen and detect the calibration in real-time. With this capacity, the process has to do an iteration of the code every time a new image is acquired. In that way, the process raises high levels of speed and simplicity.

Another great advantage is the elimination of the code development, the assistant can perform the calibration without any code, and even it also has an option of generation the code of the calibration performed in the calibration.

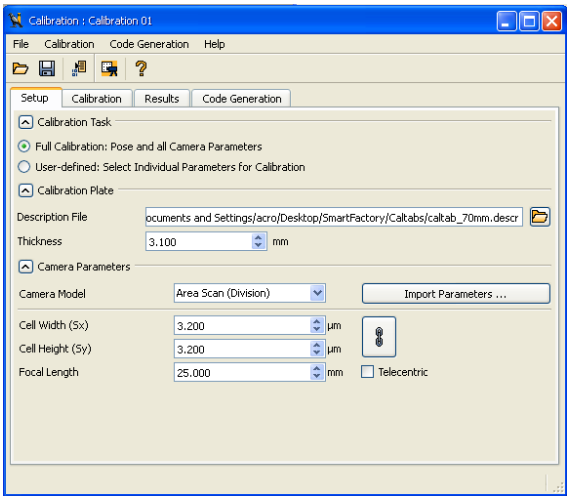


Fig. 101. - HDevelop Calibration Assistant Window

6.2.4. Calibration Procedure

The camera calibration in this project has been done in the calibration assistant, the steps deployed are quite simple.

6.2.4.1 Camera fixer Structure and Position

The camera has been fixed in a small structure just below of the light font to avoid any shadows in the field of view.

This structure also avoids any relation movement between the camera and the field of work, in that way the camera calibration has to be done only once.

6.2.4.2 Camera and calibration plate data information.

The first step is to introduce all the relative information about the camera and the calibration plate the actions done are the followings:

- Select the file “caltab_70mm.descr” that contains all the information about the calibration marks.
- Select the thickness of the calibration plate. In our case, the plate has a thickness of 3.13 mm.
- Select the camera model, in our case Area Scan Camera.
- Select the Cell Height (Sy) and Width (Sx) following the specification outlined in the data sheet of the camera (3.2 µm)
- Select the adequate focal length, determined by the objective (25 mm).

6.2.4.3 Image Acquirement

Following the possibility of use live acquirement in the calibration assistant, this steps has been done in a reduced period of time. The process is quite simple, the calibration plate is moved over all places in the scene and the image obtained is saved. Once all the space has been cover a sequence of image with tilted positions of the calibration plate are acquired and saved.

Although, the recommendations indicate a total of 15 images we have took a total of 30 images in order to ensure a high accuracy in the calibration process

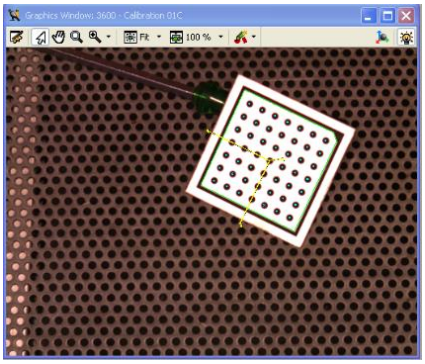


Fig. 102. - Calibration image acquired

6.2.4.4 Obtaining and saving the calibration results

Once all the position requirements of the calibration plate have been satisfied it is possible to perform the calibration. The process is as easy as a click in the button calibration situated in the calibration assistant.

Afterwards, the result obtained can be seen in the calibration assistant and be saved for further process in the vision system. In our case the camera parameters has been saved in the file “camparam.cal” and the camera pose in “cam_pose.dat” both files are attached in the sections A5.1 and 5.2 of the appendix.

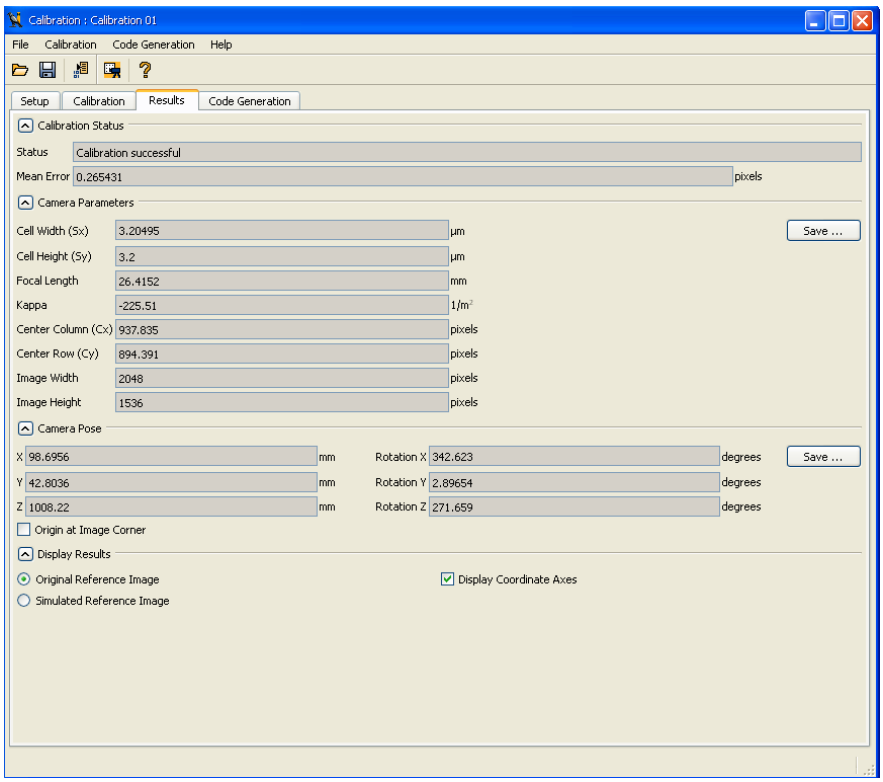


Fig. 103. - Calibration Results obtained

As we can see in the image the mean error obtained is 0.2654 pixels, with this result we can establish that the calibration process has been successfully carried out.

6.3. Shape-Based Matching.

One of the most powerful options available in HALCON for 3D matching using a 2D camera is Shape-Based 3D matching. The 3D shape model is generated from a 3D computer aided design (CAD) model.

The 3D shape model consists of 2D projections of the 3D object seen from different views. To restrain the needed memory and runtime for the shape-based 3D matching, it is necessary to restrict the allowed pose range of the shape model and thus, minimize the number of 2D projections that have to be computed and stored in the 3D shape model.

Analogously to the shape-based matching of 2D structures the 3D shape model is used to recognize instances of the object in the image. But here, instead of a 2D position, orientation, and scaling, the 3D pose of each instance is returned.

6.3.1. General Proceeding of Shape-Based 3D Matching

Shape-based 3D matching consists of the following five steps:

1. The 3D object model is accessed from file.
2. The 3D shape model is created from it.
3. The 3D object model is destroyed and memory released.
4. The 3D shape model is used to search the object in search images.
5. The 3D shape model is destroyed and memory released.

6.3.1.1 Read the 3D object model

The 3D object model describing the object of interest is accessed in HALCON with the operator *read_object_model_3d*. It must be available as a CAD model in DXF, STL, OFF, or PLY or binary format.

6.3.1.2 Create the 3D shape model

The 3D shape model is created with the operator *create_shape_model_3d*. It needs the 3D object model and camera parameters as input. Additionally, a set of parameters has to be adjusted. The camera parameters can be obtained by a camera calibration.

Before creating the 3D shape model, it is recommended to prepare the 3D object model for the shape-based 3D matching using *prepare_object_model_3d*. Otherwise, the preparation is applied internally within *create_shape_model_3d*, which may slow down the application if the same 3D object model is used several times.

Also with *inspect_object_model_3d* a pop up window appear and let the user select the desired pose range that conforms the 2d projections of the piece.

6.3.1.3 Destroy the 3D object model

After creating the 3D shape model, the 3D object model is not needed anymore and it can be destroyed for memory reasons using the operator *clear_object_model_3d*. If the 3d object model is still needed for visualization purposes for example, this step must be moved at the end of the code.

6.3.1.4 Find the 3D shape model in search images

With the 3D shape model that was created by *create_shape_model_3d* or read from file by *read_shape_model_3d*, the object can be searched for in images through the use of the operator *find_shape_model_3d*. Several parameters can be set to control the search process.

The operator returns the pose of the matching model, the standard deviation of the pose, and the score of the found instances of the 3D shape model that describes how much of the model is visible in the image.

6.3.1.5 Destroy the 3D shape model

When the 3D shape model is not needed anymore, it is destroyed with the operator *clear_shape_model_3d*.

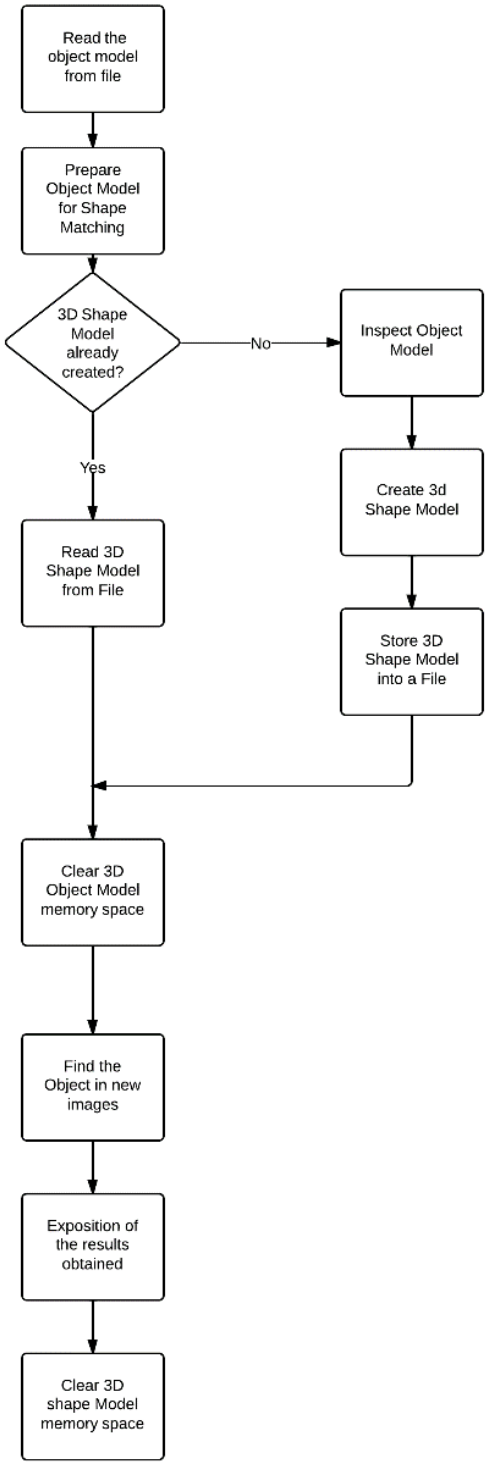


Fig. 104. – 3D Shape-Based Marching Flowchart

6.3.2. Application Example

We have developed an example of 3D Shape-Based Matching in order to interiorize all the needed concepts of the process and start with some training for the future development of the 3D matching in the work-piece that will be welded. The application has been named as *"3D Shape-Based Example.hdev".* The camera used was Ueye 1465LE-C with a COMPUTAR objective.

The CAD model of the example has been developed in SOLIDWORKS and it has been saved as a STL format in order to let HALCON to read it with the name *"Test1.stl"*.

The calibration of the Ueye has been done in the application *"Ueye calibration.hdev"* (explained in section 6.2.2.4 of this paper). After the calibration, the result window is created with the information obtained in the calibration and the acquirement device is initialized.

Once the reading of the CAD model and its preparation for shape-matching have been performed, the object model is prepared with 3D shape-based matching with the operator *prepare_object_model.* Afterwards, the application looks for a previously created shape-model and if it doesn't find it, the model is created with the operator *create_shape_model_3d*. Before the shape model is created the proper parameters given to the operator are selected through a try-error experiment in order to obtain the best results.

After the creation of the shape model, it is stored with the name "*Test1_shape_based_model.sm3" a*nd the memory space of the 3D object model is detached.

Straightaway, the finding action is deployed inside of a for and in every execution of the loop the pose obtained is save in the file "Test1_Pose.dat" and the coordinate system and shape of the finding are plot in the image previously acquired.

Finally, once the loop has ended the shape-model memory space is cleared and the application is finish.

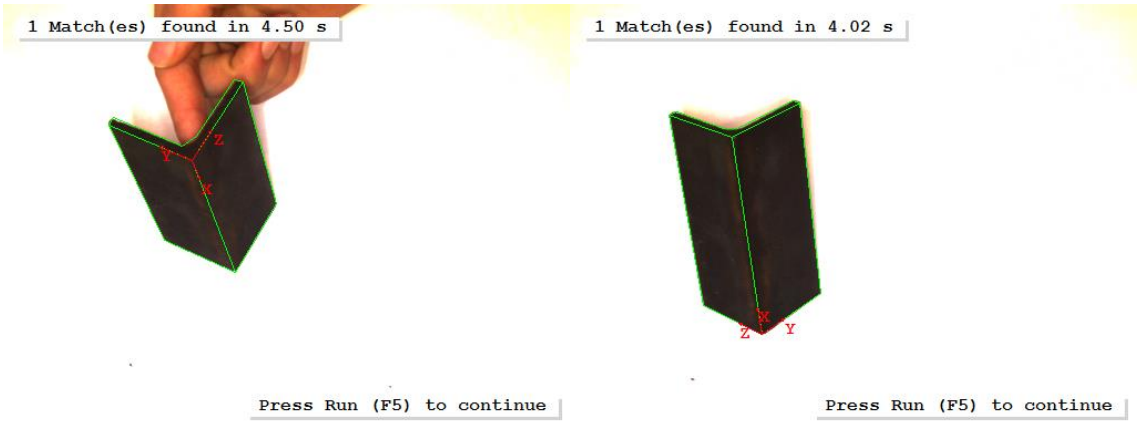


Fig. 105. - a,b Test 1 Final Results

The code implemented is outlined in the appendix section *A.4.2.*

6.3.3. Cylinder 3D Shape-Based Matching

Following the statements outlined in the abstract of this paper the objective of the project is going to be achieved through the recognition, position detection and welding of a cylinder of 48mm of diameter, 48 mm of height and 3 mm of thickness. In this section, there is described the final vision tool developed in order to achieve the recognition and position detection of the cylinder.

The process deployed is very similar to the previously outlined in the matching example, the only divergences are created in order to fit the code to the cylinder, select the proper camera and its calibration and achieve good results of matching with the cylinder in almost every position and orientation.

The CAD model is created following the same procedure than in the example, using SolidWorks and saved as a STL file, the name of the model is *"Cylinder.STL".*

The new 3D shape model searched or created has the name "Cylinder.sm3". If it is not found, before the process of creation is processed a new operator of HDevelop is introduced, *inspect_object_model_3d*. With this new operator, a group of windows is popped up in HDevelop, this window lets select the range of the imaginary pose where the 2D projections of the cylinder will be took.

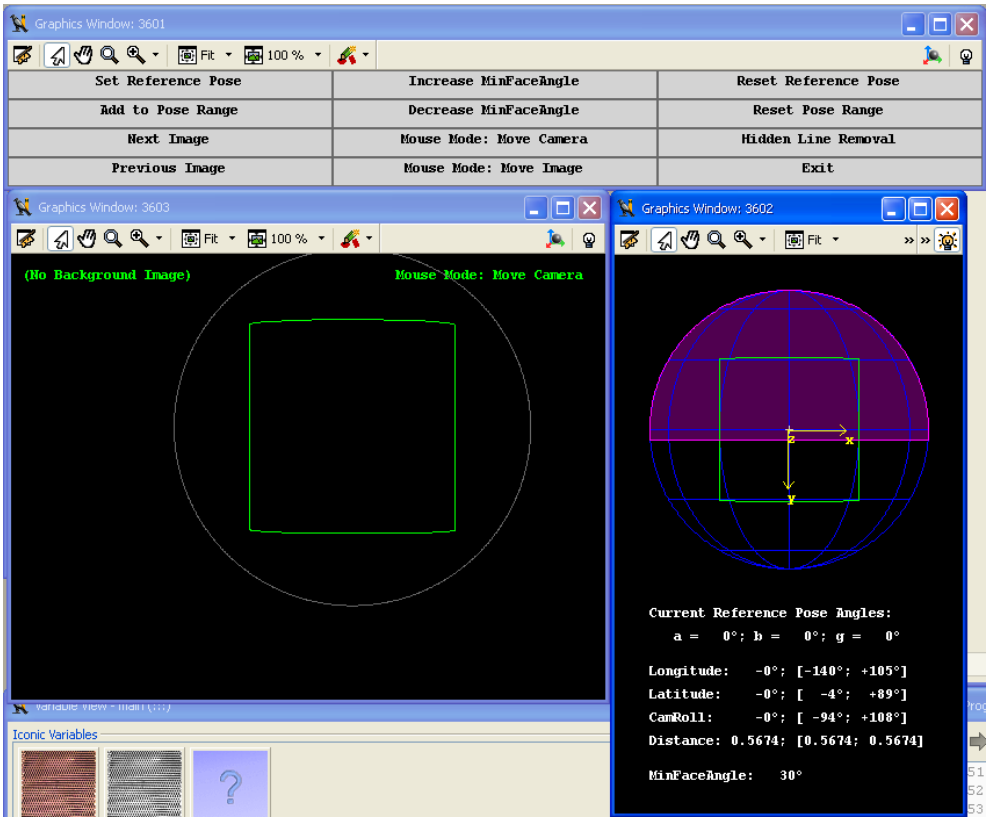


Fig. 106. - Windows view of inspect process

Once the range of the pose has been selected a list of parameters is given as an output by the inspect operator and this parameters will be given to the finding operator in order to determine the characteristics of the search. The parameters that are selected manually in this operator are

the minimum and maximum distance between the possible position of the cylinder and the camera, this distances are 0.95 m and 1.10 m. The rest of the parameters given by the inspect operator are:

- RefRotX: Rotation in X of the imaginary pose.
- RefRotY: Rotation in Y of the imaginary pose.
- RefRotZ: Rotation in Z of the imaginary pose.
- 'gba': order of rotation of the imaginary pose.
- LongitudeMin: minimum longitude of the imaginary pose origin.
- LongitudeMax: maximum longitude of the imaginary pose origin.
- LatitudeMin: minimum latitude of the imaginary pose origin.
- LatitudeMax: maximum latitude of the imaginary pose origin.
- CamRollMin: minimum camera roll during searching.
- CamRollMax: maximum camera roll during searching.

Once the cylinder has been found, before its pose is stored some homogeneous transformations has to be applied in order to move the coordinate system to the centre of the cylinder base and orient it with the axis Z normal to the base. First the pose is transformed to a 3D homogeneous matrix and then a translation and a rotation are implemented:

1. Translation of 24 mm in axis X and Y and 48 mm in Z.
2. Rotation of 180 degrees in axis X.

Once the pose has been successfully transformed it is saved with the name "*Cylinder_in_CamPose.dat",* a stored result is attached in the appendix section *A5.3.*

Some images of the matching results are outlined as follows:

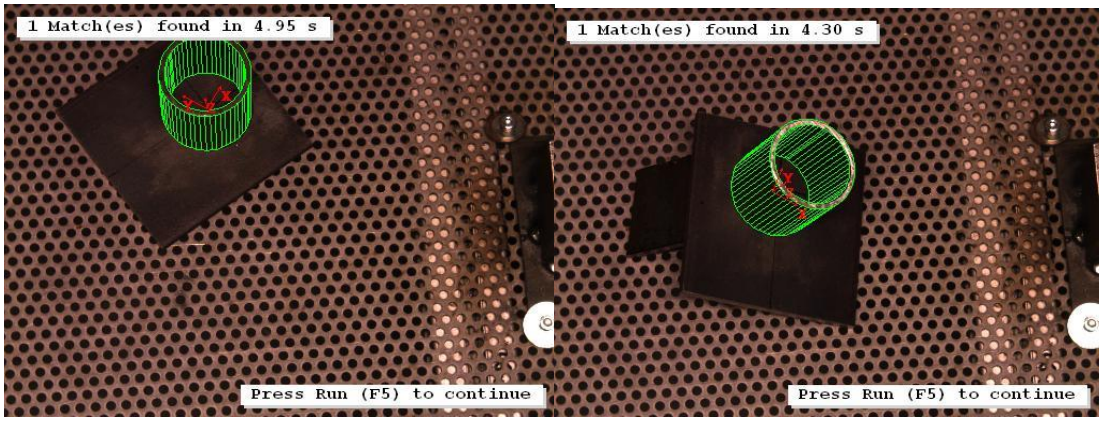


Fig. 107. - a,b Cylinder 3D Shape-Based Matching Results

The Hdevelop code is attached in section *A4.2* of the appendix.

6.4. Inclusion of the Vision Tool in the Robotic Welding Cell

After the proper development of the vision solution and obtained good results of matching the information obtained has to be correctly sent to the robot controller. To do that, it is necessarily to overcome two problems:

1. Create a proper communication link between the robot controller and the vision remote computer.
2. Determined a correct relation between the world coordinate system of the robot (situated in the base of the robot) and the camera coordinate system.

In this section the process and procedures developed to solve successfully both problems are outlined.

6.4.1. Communication between the robot controller and the vision computer

The communication link achieved has used Ethernet technology because of its facility to develop a new link and there is no need a very robust communication the information sent is no large so there is no problem of losing information packets.

The process itself is it quite simple, an Ethernet cable is connected between both system and a new network is customized in every system.

6.4.1.1 Network characteristics customization

The procedure implemented is the following:

1. In the vision computer there is only to select a correct IP address in the network characteristics, the network selected has been 10.10.107.20.
2. In the robot controller the process is the same because it is based in Windows Embed, in this case the IP selected is 10.10.107.25.

Once the IP selection has been performed it is possible to check if there is communication between both systems, this is done through the use of the command *ping* in the CMD of both systems.

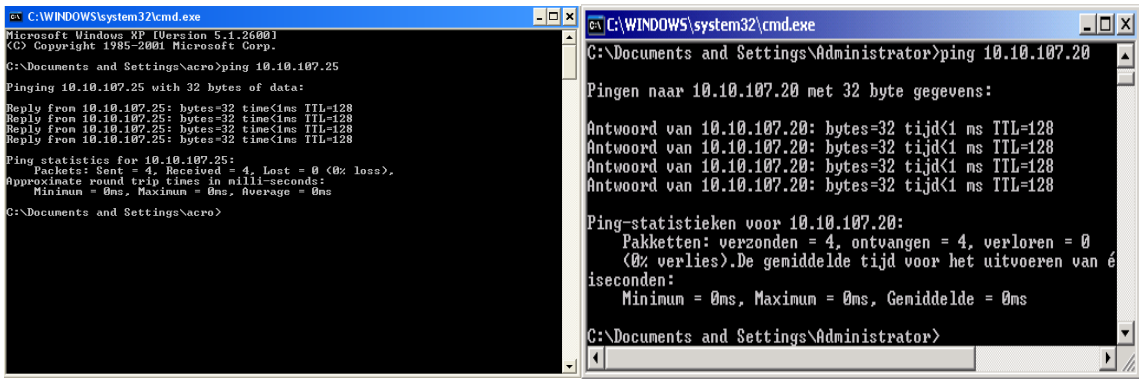


Fig. 108. - a,b Ping results obtained in both systems.

6.4.1.2 Robot Socket Interface customization

After getting a proper communication of the network, in the robot controller is it necessarily to select the correct options in the robot socket interface.

1. The proper network card has to be selected, the index of the network card selected has to be the index with the IP introduced before. Furthermore, the protocol mode have to be selected (TCP in this project).

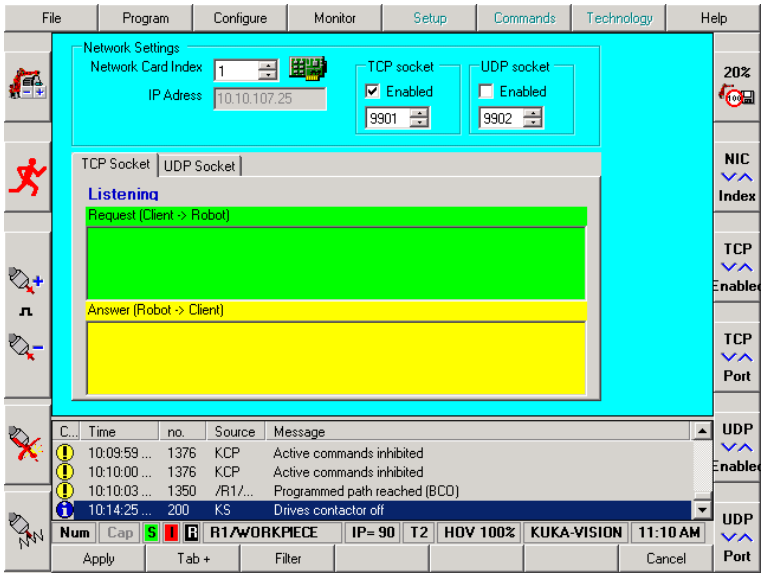


Fig. 109. - Robot Socket Interface

2. Also, the IP address of the vision computer has to be added in the FILTER file as a host IP address, also in this file the variables of the robot controller that want to be read or written has to be selected. In our case, $POS_ACT (actual position of the robot has to be read to the Hand-Eye calibration) and BASE_DATA[3] (base coordinate system defined by the vision system) has to be written.

3. After all the customizations have been done, the bottom Apply is clicked and the socket configuration is updated.

6.4.1.3 Communication Procedure in Halcon

As we have established before Halcon is not only a vision tool, a part of its vision features it also ensure to realise in it communication links, send and receive information of a specific network. In our project, this process is done through the adding of a few lines in every application where the communication has to be performed.

The process itself follows a specifics steps:

1. First, the IP address of the robot controller and the vision computer are selected and saved it in the variables: *local_IP_addres*s (vision computer) and *remote_IP_address* (robot controller). Furthermore, the local and remote port number, the maximum time of waiting (timeout) and the kind of port are selected. Here are the lines of this selection:

```
local_IP_address:='10.10.107.20'
remote_IP_address:='10.10.107.25'
Local_port_number:=9901
Remote_port_number:=9901
TCP_or_UDP:='TCP'
timeout:=200
```

2. The second step in the subroutine is open a new socket through which the communication is achieved. This is done with the HDevelop operator *open_socket_conect*. Also, the information that are going to be read or sent has to be in a xml file, this file are saved in two variables, this two variable will be modified to send/receive the correct value and information to the robot controller:

```
*opening the socket
open_socket_connect (remote_IP_adres, 9901, ['protocol','timeout'],
['TCP',timeout], Socket)
*preparing data to send
data_send_write:= '<?xml version="1.0" encoding="UTF-8"?><SetVar VarValue=""
VarName="$OV_PRO"/>'
data_send_read:='<?xml version="1.0" encoding="UTF-8"?><ShowVar
VarName="$POS_ACT"/>'
```

3. Once the message has been correctly modified it is sent and receive through the socket, this actions are performed with the operators *send_data/receive_data:*

```
send_data (Socket, 'z', data_send_read, [])
receive_data (Socket, 'z', DataReceived, From)
```

4. After receive the information, in order to could use it, it is necessary to extract the interesting information from the xml file received. For example, if we want to read the actual position of the weld torch respect the robot base (this will be used in the Hand-

Eye Calibration) the pose information has to be extracted from the complete message, this is done through the subtraction of a part of the message where the pose information is contained. This operation is carried out in HDevelop with the operators: tuple_ strstr (find the key word afterward is situated the information searched) and tuple_substr (extraction of the information, with the length of the extraction indicated manually)

```
*Sending data request to the robot
send_data (Socket, 'z', data_send_read, [])
*Recieve data
receive_data (Socket, 'z', DataReceived, From)
*Find the position of the information in the data recieved
tuple_strstr (DataReceived, 'X', Position)
tuple_substr (DataReceived, Position+2, Position+8, Rob_X)
tuple_number(Rob_X,Rob_X)
tuple_strstr (DataReceived, 'Y', Position)
tuple_substr (DataReceived, Position+2, Position+7, Rob_Y)
tuple_number(Rob_Y,Rob_Y)
tuple_strstr (DataReceived, 'Z', Position)
tuple_substr (DataReceived, Position+2, Position+8, Rob_Z)
tuple_number(Rob_Z,Rob_Z)
tuple_strstr (DataReceived, ',A ', Position)
tuple_substr (DataReceived, Position+2, Position+8, Rob_A)
tuple_number(Rob_A,Rob_A)
tuple_strstr (DataReceived, ',B ', Position)
tuple_substr (DataReceived, Position+2, Position+8, Rob_B)
tuple_number(Rob_B,Rob_B)
tuple_strstr (DataReceived, ',C ', Position)
tuple_substr (DataReceived, Position+2, Position+8, Rob_C)
tuple_number(Rob_C,Rob_C)
```

5. Finally, after all the messages have been sent and received properly the socket connection is closed with the operator *close_socket*.

```
*closing socket
close_socket(Socket)
```

We have to take into account that, in order to receive data, first there is needed to send an initial message to the robot controller outlining the information of the variables that we want to receive. Here is the process we have to follow:

```
send_data (Socket, 'z', data_send_read, [])
* To receive data we have to send data first to tell the robot what data we want to
*receive
receive_data (Socket, 'z', DataReceived, From)
```

6.4.2. Hand-Eye Calibration

A typical application area for 3D vision is robot vision. Such systems are also called “hand-eye systems” because the robotic “hand” is guided by mechanical “eyes”.

In order to use the information extracted by the camera, it must be transformed into the coordinate system of the robot. Thus, besides camera calibration it is needed also calibrate the hand-eye system (determine the pose transformation between camera and robot coordinates).

There are two possible scenarios of Hand-Eye applications. The camera can either be mounted on the robot and is moved to different positions by it, or it can be stationary.

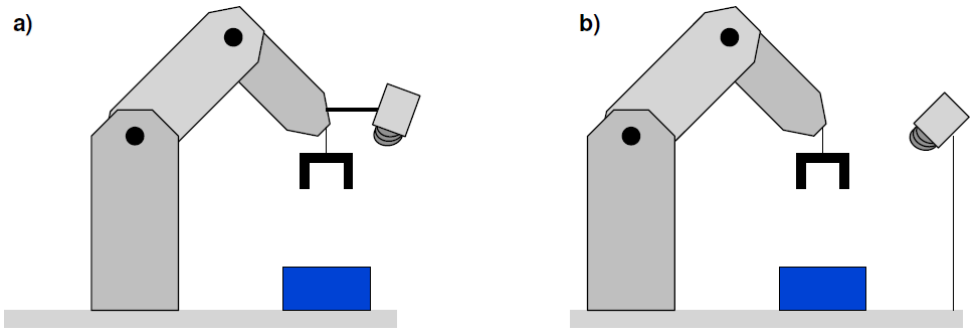


Fig. 110. - Robot vision scenarios: a) Moving Camera, b) Stationary Camera

We have selected a stationary camera scenario in order to not limit the reachability of the weld torch in the work-zone and to avoid possible damages in the camera during the welding action.

6.4.2.1. Hand-Eye Calibration Fundamentals

As well as the camera calibration, the hand-eye calibration is based on providing multiple images of a known calibration object. But in contrast to the camera calibration, here the calibration object is not moved manually but by the robot, which moves either the calibration object in front of a stationary camera and in every new image the robot pose information is provided to the calibration method by the robot controller.

The chain of transformation in order to obtain the coordinates acquired by the camera in the tool coordinate system of the robot in a stationary system follows this form:

$$H_{Cal}^{Cam} = H_{Base}^{Cam} \cdot H_{Tool}^{Base} \cdot H_{Cal}^{Tool} \quad (2)$$

In the chain of transformations, H_{Tool}^{Base} corresponds with the transformation pose given by the robot controller, H_{Base}^{Cam} and H_{Cal}^{Tool} with the poses obtained in the Hand-Eye calibration.

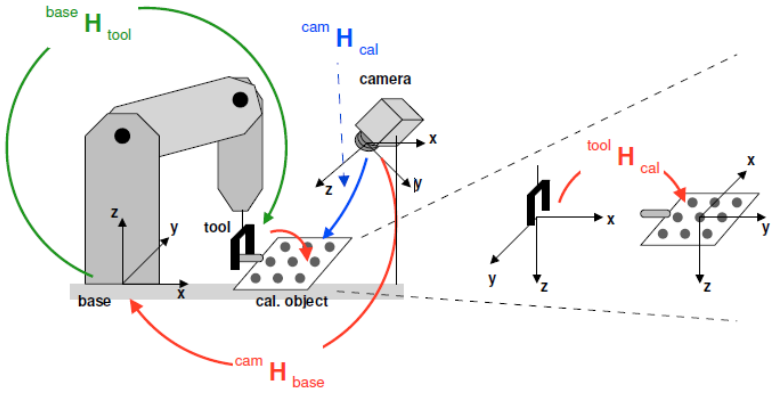


Fig. 111. - Chain of Transformation for a stationary camera system

The process of Hand-Eye calibration is divided into two different stages:

1. Acquire the images and the robot poses information.
2. Perform the Hand-Eye calibration and store the results.

As we have commented before, our application is centred in obtaining a coordinate system from the matching and then use this information to create a base coordinate system in the robot. Because of that, we only need to know the pose H_{Base}^{Cam} (obtained in Hand-Eye Calibration) and the H_{Cam}^{Cylin} (obtained in the matching process) in order to perform the pertinent homogeneous transformations and obtain the new base coordinate system. In this situation, we do not need to determine the manually the H_{Cal}^{Tool} neither use the H_{Tool}^{Cam} obtained in the Hand-Eye calibration.

The calibration plate has to be as best as possible fixed to the robot in order to avoid any position distortion between images that would worsen the calibration results. Following this statement, a little structure has been created in order to achieve a proper fastening.

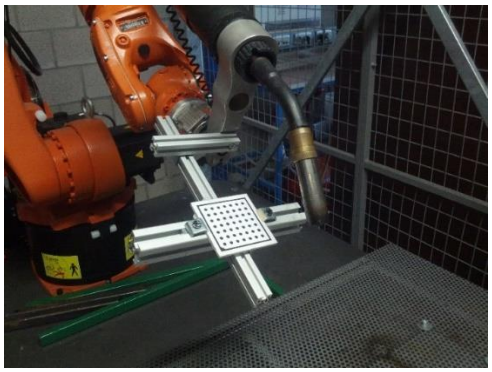


Fig. 112. - View of the calibration plate and its fixer structure mounted in the robot

As seen in the previous image, the calibration plate is the same used in the camera calibration process because it satisfies perfectly the recommendations to obtain a proper calibration commented in the point 6.2.1.1.

6.4.2.2 Acquirement of images and robot pose information.

The process of acquirement and store all the information needed to perform the Hand-Eye calibration has been gathered in the Halcon file "*Grabbing calibration images and robot info.hdev"*.

This application can be divided into the image acquirement part and the robot position acquirement section, but this two parts are inside of a for loop to corresponds every image with the correct robot position.

The process of images acquirement is very simple, the connection between the camera and the computer is performed as a first instance outside of the loop. Then every time the robot has been moved into a new position, the exposure of the camera is adjusted in order to ensure the correct recognition of the calibration plate in the Hand-Eye calibration. Finally the image are stored with a name "*Hand_EyeCal0I.png"* where the I corresponds with the times of iterations of the for loop. Finally once the loop has over the connection with the camera is finished.

The process of grabbing the robot position starts outside the loop with the introduction of all the communication information needed to create the communication link, this part is exactly the same as the presented in section 6.4.1.3. After that, in every iteration of the loop, a new xml file is sent from the robot controller, the information X, Y, Z, A, B and C is extracted from the file and converted into literal. After that a new pose is created with this information of the robot and stored with the name "*Robot_Pose0I.dat".* Once outside of the loop, the socket is close and the link communication is finished.

We have to take into account the kind of pose that the robot controller gives in order to create the robot pose properly, it uses the order of rotation 'abg' (RotZ(A) ·RotY(B)·RotX(C)). Following that, the code line that create the pose has this form:

```
create_pose(Rob_X,Rob_Y,Rob_Z,Rob_C,Rob_B,Rob_A,'Rp+T', 'abg', 'point' ,Robot_Pose)
```

Although, the hand-eye calibration only needs three different images with the robot in three different positions, we have acquired a total of fifteen images in order to ensure an acceptable results in the relation of the robot coordinate system and the camera coordinate system.

The code deployed in Hdevelop to perform this acquirement is outlined in the appendix section *A4.4.*

6.4.2.3. Hand-Eye Stationary Calibration Procedure

The Hand-Eye calibration always follows a specific sequential process, this process is outlined in the following flowchart.

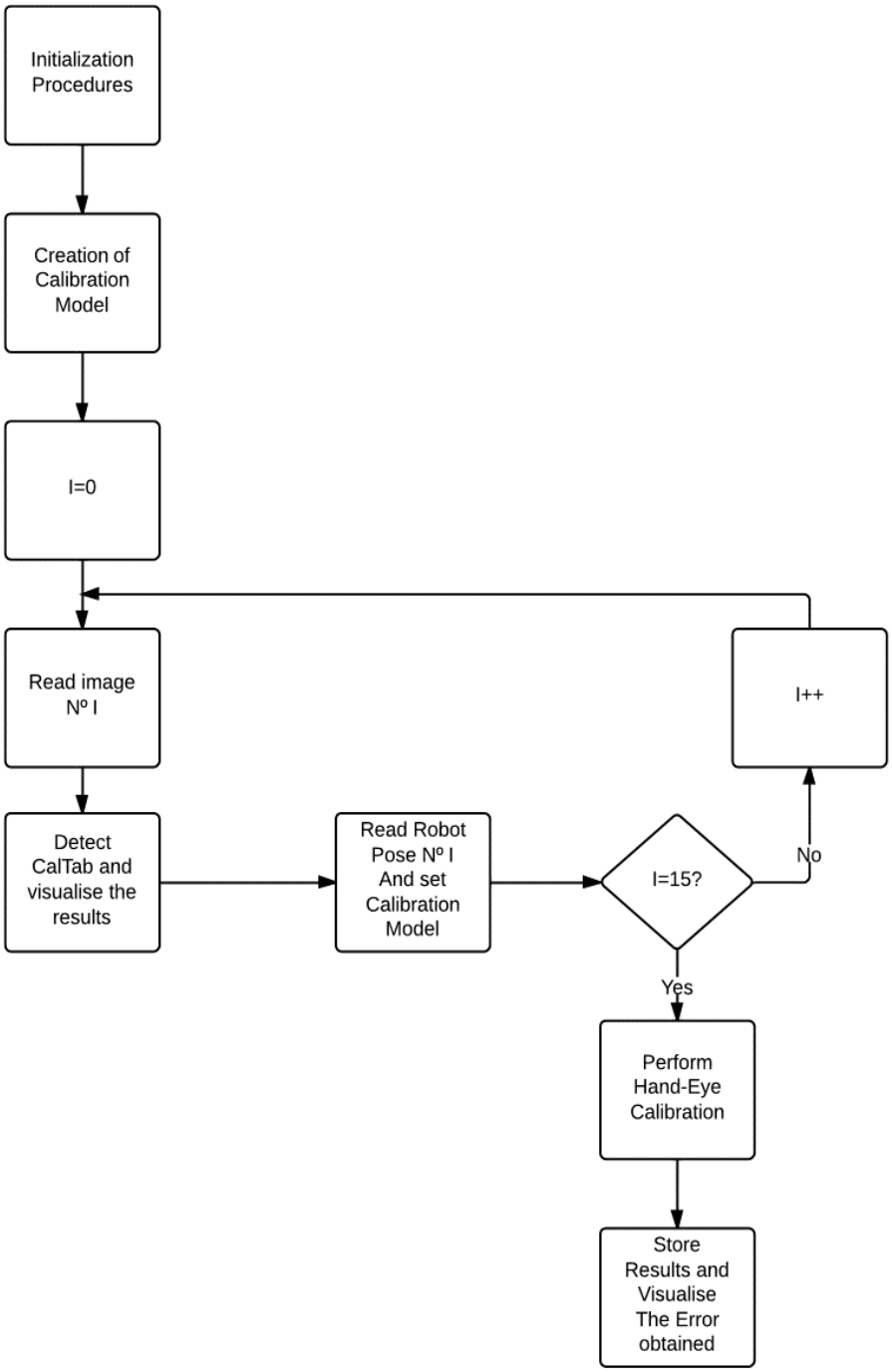


Fig. 113. - Hand-Eye Calibration process flowchart

- Initialization procedure

In this section the process performed are:

- The directories that contains the images and the robot poses are charged into a two variables.

- The number of images used is set.
- Creation of the windows results with the height and width information of one image.
- The description file of the calibration plate is charged, "*caltab_70mm.descr".*
- Camera parameters obtained during the camera calibration are charged into a variable.

- Creation of the calibration model

In this section the hand-eye calibration model is created, and customized with the camera parameters and the calibration plate used. The creation code created is:

```
create_calib_data ('hand_eye_stationary_cam', 1, 1, CalibDataID)
set_calib_data_cam_param (CalibDataID, 0, 'area_scan_division', StartCamParam)
set_calib_data_calib_object (CalibDataID, 0, CalTabFile)
```

-For Loop

Inside the loop created with 15 iterations, the image NºI is read and then the calibration object is searched in it with the operator *find_calib_object,* in this operator the smoothing parameter is changed in every iteration to ensure the correct detection of the calibration plate. After the correct detection, this detection is showed in the result window.

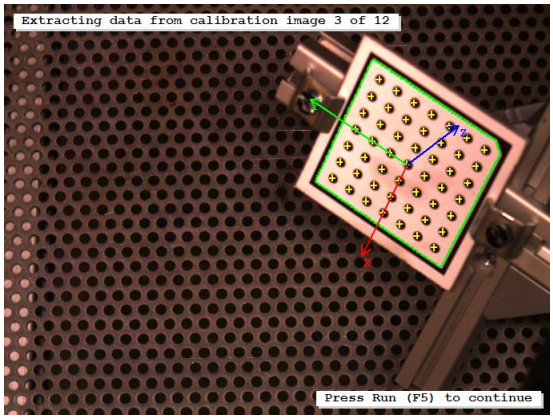


Fig. 114. - Example of Calibration plate detection in Hand-Eye calibration

Sequentially, the robot pose NºI is read and then the position parameters of the pose (X, Y and Z) are transformed from millimetres to metres because Halcon works in metres but the robot controller gives the information in millimetres. Afterwards, the calibration model is set with the robot information through the operator *set_calib_data*.

-Perform the calibration and store the results.

Once the fifteen position are processed the hand-eye calibration is performed by the operator *calibrate_hand_eye.* In this operator the errors of the process are saved and showed in the window results. The pose of the transformation between the robot coordinate system and the camera is query to the calibration model with the operator *get_calib_data* and saved. Finally,

the result obtained are saved in file "*Robot_Posefinal_pose_cam_base.dat*", the calibration model is cleared and the application is finished.

Quality of the results:	root mean square	maximum
Translation part in m:	0.0008	0.0016
Rotation part in degree:	0.3869	0.6651

Fig. 115. - Error obtained in the Hand-Eye Calibration

As we can see, the calibration is achieved with a maximum errors of 1.6mm and 0.6651˚ of position and orientation respectively. With this results we can establish that the accuracy obtained is acceptable to perform welding applications.

As well as the code of the acquirement, the code of this process is showed up in the appendix, in section *A4.5*.

6.4.3. Transform and send base coordinate system to robot system

Once the cylinder is properly matched and the hand-eye is successfully performed the only steps remaining are to transform the match pose and send it to the robot.

The code deployed that performs both steps is outlined in the appendix *A4.6.*

6.4.3.1 Transform the final pose

The needed homogeneous transformation that ensure the correct transformation of the position and orientation to the robot coordinate system (WCS) is very simple, there is only one matrix composition as we can see in the mathematical description of the transformation:

$$H_{Cylin}^{RobBase} = H_{Cam}^{RobBase} \cdot H_{Cylin}^{Cam} \quad (3)$$

Looking the previous expression H_{Cylin}^{Cam} is obtained from the matching process. However, during Hand-Eye calibration process, the pose obtained is the robot coordinate system in camera coordinate system ($H_{RobBase}^{Cam}$). Making a simple transformation the expression changes to:

$$H_{Cylin}^{RobBase} = \left(H_{RobBase}^{Cam}\right)^{-1} \cdot H_{Cylin}^{Cam} \quad (4)$$

This transformation in HDevelop are contained the following code lines:

```
hom_mat3d_invert(Cam_H_RobBase,RobBase_H_Cam)
hom_mat3d_compose(RobBase_H_Cam,Cam_H_Cylinder,RobBase_H_Cylinder)
```

```
hom_mat3d_to_pose(RobBase_H_Cylinder,Pose_Cylinder_in_RobBase)
```

Although the processes have been performed in the highest possible options of accuracy, during them, some little errors have appeared. Furthermore, the cylinders used do not have the highest dimension accuracy. Because of that some distortions have appeared in the final results. In order to avoid that, after an average process of the total variations in every axis, the manually offsets added have been:

- -0.0045 m in X
- -0.0025 m in Y
- 0.0041 m in Z

Once this offsets have been added, the pose is changed into mm and changed its type into the type that the robot uses. Finally the pose, is stored with the name "*CylinderinRobBase_Pose.dat*", an example of the final pose is outlined as follows:

```
# Used representation type:
f 2
# Rotation angles [deg] or Rodriguez vector:
r 223.704674222797 313.247499092894 223.704674222797
# Translation vector (x y z [mm]):
t 1161.7886418786 -173.815938824972 360.515939419688
```

6.4.3.2 Send Final Pose

The send process follows the form previously commented in the section 6.4.1.3. First the socket is open, and then the xml file with all the base coordinate system is sent through the operator "*send_data*".

As well as in the receiving message the xml message is a string for HDevelop, because of that, before be sent, the final pose has to be converted into a string and then the different parts of the pose must be properly introduced in the xml file. The xml file is stored in the variable data_write:

```
data_send_write:= '<?xml version="1.0" encoding="UTF-8"?><SetVar VarValue="{FRAME: X
'+Cylinder_RobBase_String[0]+',Y '+Cylinder_RobBase_String[1]+',Z
'+Cylinder_RobBase_String[2]+',A 0,B 0,C 0}" VarName="BASE_DATA[3]"/>'
```

As we can see in the xml file the information about the orientation (A, B and C) is not sent, this is caused by the restrictions that the robot has to realise the circular weld (commented in section 5.6.1.1). Furthermore, the cylinder is not going to be tilted in axis X or Y because it needs to be perfectly attached to the welding platform. In that way, the rotation the cylinder in axis Z does not matter, the robot will follow always the pre-taught pattern.

6.5. Final Vision Solution Obtained

Following the development steps described in section 3.3.1.1, the next process to achieve is encapsulate all the HDevelop code in a Visual Basic application in order to present a close application to the user without any possibility of operating errors.

The first point to satisfy is to export the different features of the code deployed in HDevelop into a Visual Basic Language. As we have outlined before in this paper, Halcon has the possibility of this exportations, the only process we have to realise is segment the different HDevelop codes to encapsulate all the features of the final application in different buttons and windows in the visual basic application.

Once all the features have been properly isolated and exported to Visual Basic, a template of Visual Basic 2005 of Halcon is used and customized to obtain the final tool with all the desired characteristics.

6.5.1. Segmentation of the vision solution

Every segmentation realised from the Hdevelop codes are stored in ".vb" files that posteriorly be added them as a link in the visual basic project. In total, 5 segmentation has been performed and stored into 5 different files:

- "*Acquirement.vb"*
- *"ShapeModelCreation.vb"*
- *"CylinderMatching.vb"*
- *"ConvertPose.vb"*
- *"Communication.vb"*

The code obtained after the five exportations is outlined in the section *A6* of the appendix.

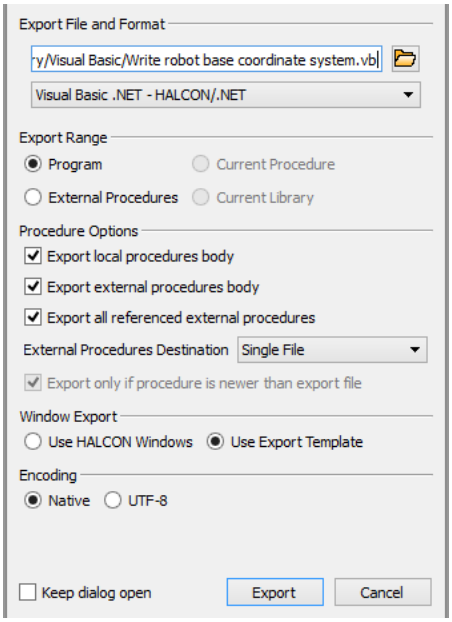


Fig. 116. - Export Window assistant in Hdevelop

6.5.1.1 Acquirement.vb

The first code exported is a simple feature, the process closes any possible framegrabber that could remain open, opens a new framegrabber (AcqHAndle) with the UEye camera, acquires and display an image. Finally, it closes the framegrabber opened.

The main process of this file is performed inside a visual basic procedure called *ActionAcq*. As well as the main process *ActionAcq* is called by another visual basic procedure that is the used in the windows form code (explained in the following sections), this procedure is called *RunHalconAcq*. This procedure has as a parameter a Window identification passed by reference, in this window is where the image acquired will be showed up.

6.5.1.2 ShapeModelCreation.vb

This section contains all the process of the 3D Shape Model creation. The steps followed are the same as the Hdevelop application outlined in section 6.3.3. However, in this file there is only performed the creation feature, the steps contained are the followings:

1. Read a 3D object model and camera parameters.
2. Prepare 3D object model.
3. Inspect 3D object model to obtain the shape model creation parameters.
4. Create the shape model.

As well as in the acquirement file, this action are encompassed in a procedure called *ActionCreateShapeModel* and posteriorly, it will be called by a procedure called

"*RunHalconCreateShapeModel*". This procedure have as a parameters four windows passed by reference, all are needed for the 3d model inspection. This windows are situated in the Shape Creation Form (described posteriorly).

Apart from all the features commented before, this file also contains the action of saving the 3D Shape Model, but this process is encompassed in a different procedure without any input parameter called *"RunHalconSaveShapeModel"*.

In this exported file some problem appeared because of the Halcon template is only prepared to show up one HWindow and to inspect the 3d object model there are necessarily four HWindows. The problem was solved through the modification of the default procedure including the four HWindows passed as references and through the customization of the exported file to show the correct information in each HWindow not all in one.

6.5.1.3 CylinderMatching.vb

In contrast with the previous file, this archive contains all the process to match the cylinder in the image using the 3d shape model created before. The steps deployed are the followings:

1. Open a framegrabber.
2. Read 3D Shape Model and camera parameters.
3. Grab a new image.
4. Match the shape model.
5. Show shape model contour in window and stored the pose obtained in a global variable.
6. Close the framegrabber.

All this steps are encompassed first in a procedure called ActionMatch and it is called by the procedure "*RunHalconMatch"* that has a Halcon window as a parameter passed by reference.

In this file, it is also the procedure *"FinishHalcon"* that destroy the 3D Shape model if it has been previously read.

6.5.1.4 ConverPose.vb

This file performs all the necessarily transformation that have to be done in order to obtain the pose in the robot base coordinate system (WCS) as well as the addition of the error offset commented before. This processes are encompassed first in a procedure called ActionConvert. It is called by the procedure "*RunHalconConvert".*

In this file, there is also gathered the action of save posed called by the procedure "*RunHalconSavePose".*

6.5.1.5 Communication.vb

The last file contains all the communication code commented in section 6.4.3.2. The only difference is that the different process are encompassed in different procedures to facility the programming of the Communication Form.

The action of opening and closing the socket are inside in the procedures *ActionConnect* and *ActionDiconnect* called by *"RunHalconConnect"* and *"RunHalconDisconnect"* respectively.

The process of send is encompassed in *ActionSend* called by "*RunHalconSend"*. In this procedure, the actions performed are:

1. Convert the pose to the form used by the robot controller: In millimetres, A being RotZ and C RotX.
2. Creation of the proper xml file with the pose information in it.
3. Send of the xml file to the robot controller with the new value of the base frame.

6.5.2. Encapsulation of the final vision solution in a Visual Basic Environment

Once all the features have been correctly sequenced and exported, the next step is to design and program the windows forms that will contain all the information and show the results and the states of the process.

In total, three windows forms are created: *VisionToolForm*, with all the initial information and the Halcon window result. ShapeModelCreationForm, which contains all the process of the 3d shape model creation. And, finally, CommunicationForm, that encompassed all the communication processes with the robot controller and the pose conversion to the robot base coordinate system.

6.5.2.1 Visual Basic brief description

Visual Basic is a high level programming language evolved from the earlier DOS version called Basic. Basic (Beginners All-purpose Symbolic instruction Code). It is a fairly easy programming language to learn. The codes look a bit like English Language.

Visual Basic is a visual and events driven Programming Language. These are the main divergence from the old Basic. In Basic, programming is done in a text-only environment and the program is executed sequentially. In Visual Basic, programming is done in a graphical environment. Because users may click on a certain object randomly, so each object has to be programmed independently to be able to response to those actions (events). Therefore, a Visual Basic Program is made up of many subprograms, each has its own program codes, and each can be executed independently and at the same time each can be linked together in one way or another.

It is possible to choose to start a new project, open an existing project or select a list of recently opened programs. A project is a collection of files that make up your application. There are various types of applications we could create; however, we shall concentrate on creating Standard EXE programs (EXE means executable program).

The Visual Basic Environment consists of the:

- A Blank Form to design the application's interface.
- The Project window which displays the files that are created in your application.
- The Properties window which displays the properties of various controls and objects that are created in your applications.

It also includes a Toolbox that consists of all the controls essential for developing a VB Application. Controls are tools such as text box, command button, label, combo box, picture box, image box, timer and other objects that can be dragged and drawn on a form to perform certain tasks according to the events associated with them. Additional objects can be added by clicking on the project item on the menu and click on components on the drop-down list.

6.5.2.2 Add Halcon features to Visual Studio

Once the procedures of HDevelop are exported, some features of Halcon has to be added to Visual Studio in order to could work with them.

HALCON/.NET assembly provides not only a class library but also one control: HWindowControl, which contains a HALCON graphics window for visualizing images and results. It is possible to add this control to Visual Studio's toolbox by performing the following steps:

- Right-click on the toolbox and select Customize Toolbox. This will open a dialog displaying all available .NET Framework components in a tab.
- Click on Browse, navigate to the directory %HALCONROOT%\bin\dotnet20 (Visual Studio 2005) or %HALCONROOT%\bin\dotnet35 (Visual Studio 2008) and select halcondotnet.dll.
- Then, the icon of HWindowControl appears in the toolbox.
- When developing an application with HALCON XL, you must select halcondotnetxl.dll instead of halcondotnet.dll.

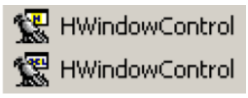


Fig. 117. - HWindow Controll for Default and XL Halcon applications

To be able to use the HALCON/.NET classes without prefixing them with their namespace, it is possible to specify this namespace at the beginning of each source file.

```
Imports HalconDotNet
```

The exported code can be integrated into the so-called template project (available for C# and Visual Basic .NET) in the subdirectory HDevelopTemplate (or HDevelopTemplateWPF, depending on your preferred GUI platform) as follows:

- Move or copy the exported source code file into subdirectory source of the template application.
- Open the solution file, right-click in the Solution Explorer, and select the menu item Add Existing Item. Navigate to the source code file, but don't click Open but on the arrow on the right side of this button and select Link File.
- When the application is running, if the button run is clicked the exported HDevelop program starts.
- If the exported program is not added correctly in Visual Basic .NET error messages appears.

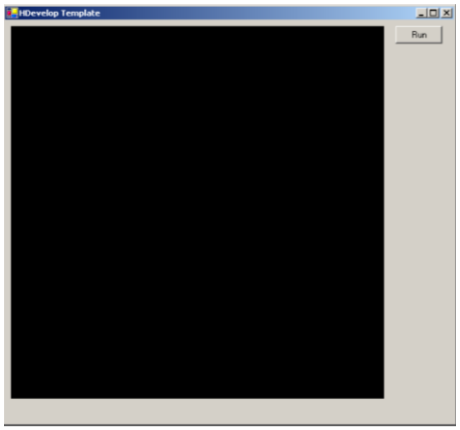


Fig. 118. - Halcon Template for Visual Basic applications

6.5.2.3 Vision Tool Form

The first window show once the EXE file is clicked is the principal window. In it is possible to realise this different process:

1. Open Camera parameters file or 3D shape model.
2. Acquire a new image calling the procedure *RunHalconAcq*.
3. Open the ShapeCreationForm through the clicking in Create Shape Model button.
4. Match the cylinder calling the function *RunHalconMatch.*
5. Open the CommunicationForm through the clicking in Send Poste to robot button
6. Visualise the number of pieces detected and its poses in camera coordinate system.
7. Close the program and finish the application.

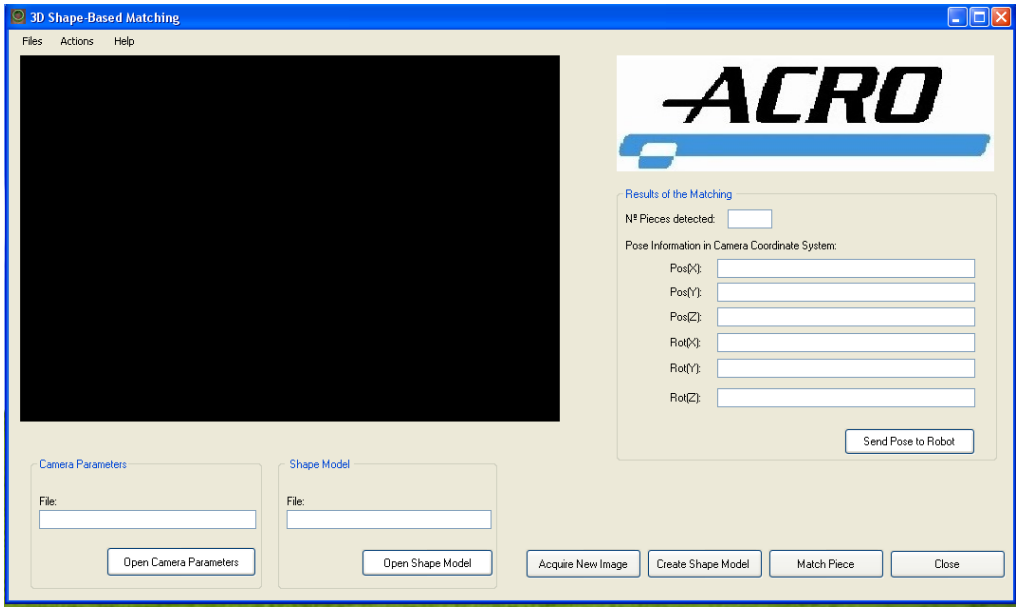


Fig. 119. - VisionToolForm

In order to avoid errors during the running of Halcon procedures some conditions are established in order to start the process of create shape model, match piece or send pose to robot. This requirement are the followings:

1. The camera parameters and the 3d shape model opened have to present the correct file extension (.cal and .sm3) to start the matching process. The directory where the file are stored is obtained with an Open File Dialog. This function also offers many solutions to check the correct file extension. This condition is checked for camera parameters as follows (for the shape model the condition structure is the same, only changes the file extension):

```
If Len(ofdCamera.FileName) = 0 Then

    'Do nothing
Else
    CamParamName = Split(ofdCamera.SafeFileName, ".")
    If CamParamName(CamParamName.Length - 1) = "cal" Then
        txbCamParam.Text = ofdCamera.SafeFileName()
        CamParamDirectory = (ofdCamera.FileName())

    Else
        txbCamParam.Text = "File extension invalid, charge again"
    End If
End If
```

2. To run the shape model creation first the camera parameters file has to be opened first.

```
Private Sub btnCreateShapeModel_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnCreateShapeModel.Click
 If Len(ofdCamera.FileName) = 0 Or CamParamName(CamParamName.Length - 1) <>
 "cal" Then
      Call MessageBox.Show("Camera Parameters has to be properly selected",
      Me.Text, MessageBoxButtons.OK)
 Else
```

```
        ShapeCreationForm.Show()
    End If

End Sub
```

3. The last requirement is to check that a cylinder pose has been created before to let pop up the CommunicationForm when the button Send Pose to Robot is clicked. This is achieved through the global Boolean variable `CylinPoseMatch`.

```
Private Sub btnSend_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSend.Click

        If CylinPoseMatch = True Then
            CommunicationForm.Show()
        Else
            Call MessageBox.Show("There is no pose to send", Me.Text,
            MessageBoxButtons.OK)
        End If


End Sub
```

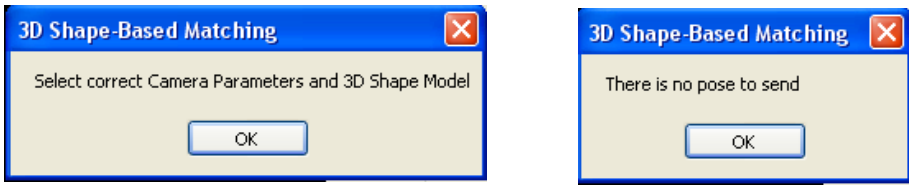


Fig. 120. - a,b Error messages in VisionToolForm

The total code of the VisionToolForm is outlined in the section *A6.7* of the appendix.

6.5.2.4 Shape Creation Form

Once the camera parameters have been properly opened in the Vision Tool Form, it is possible to pop up the Shape Creation Form. In this window a total of four HWindow are showed (in order to ensure the correct functionality of the 3d object model inspection). The features added in this windows form are:

1. Open a 3D object Model with an Open File Dialog.
2. Start the shape model creation calling the procedure *RunHalconCreateShapeModel.*
3. Save the 3d shape model obtained through the calling of the procedure *RunHalconSaveShapeModel.*
4. Close the window and back to The Vision Tool Form.

As well as in The Vision Tool Form some conditions have to be satisfied before the beginning of the shape model creation process:

- A 3D object model has to be charged with its correct file extension (.stl) using the Open File Dialog possibilities. The code to achieve that is similar to the check realised with the camera parameters only changing the file extension.

- A 3d object model has to be charged before the star up of the shape creation process:

```
If Len(ofdSTLModel.FileName) = 0 Or StlModelName(StlModelName.Length -
1) <> "STL" Then
	Call MessageBox.Show("A 3D Model has to be selected", Me.Text,
	MessageBoxButtons.OK)
Else

	Call RunHalconCreateShapeModel(WindowIDMenu, WindowIDSelect,
	WindowIDView, WindowIDBuffer)
	ShapeCreated = True
	ShapeRead = False
	Call MessageBox.Show("3D Shape Model created succesfully",
	Me.Text, MessageBoxButtons.OK)

End If
```

- In order to save the 3d shape model first it has to be created, this is controlled with the global variable *ShapeCreated*:

```
If ShapeCreated = True Then
	sfdShapeModel.Filter = "sm3 Files (*.sm3*)|*.sm3"
	sfdShapeModel.InitialDirectory = "C:\Documents and
	Settings\acro\Desktop\SmartFactory\matching\Cylinder"
	sfdShapeModel.ShowDialog()

	If sfdShapeModel.FileName <> "" Then
		ShapeModelDirectory = (sfdShapeModel.FileName())
		Call RunHalconSaveShapeModel()
		Call MessageBox.Show("3D Shape Model saved
		succesfully", Me.Text, MessageBoxButtons.OK)
	End If

Else
	Call MessageBox.Show("3D Shape Model has to be created first",
	Me.Text, MessageBoxButtons.OK)
End If
```

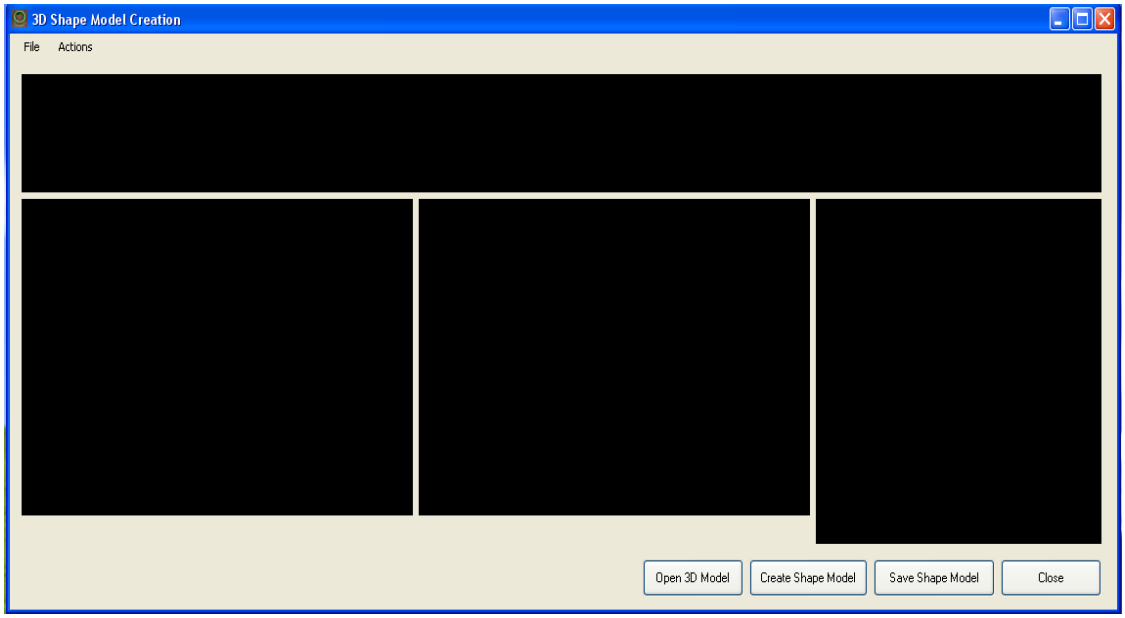


Fig. 121. - Shape Creation Form

The complete code of the Shape Creation Form is attached in the section *A6.8* of the appendix.

6.5.2.5 Communication Form

After the correct matching of the cylinder, through the clicking in the button "Send Pose to Robot" the Communication Form pops up. The features added in this form are the followings:

1. Manually introduction of the remote IP and the Port.
2. Open and Close socket with the robot controller through calling to *RunHalconConnect* and *RunHalconDisconnect.*
3. Using an Open File Dialog, charge the pose that relates camera coordinate system and robot base coordinate system (obtained during the Hand-Eye Calibration).
4. Convert the pose to the type used by the robot controller through calling the procedure *RunHalconConvert.*
5. Save the pose converted with the procedure *RunHalconSavePose* and selecting the store folder with a Save File Dialog.
6. Send Pose to the robot controller with the procedure *RunHalconSendPose*.
7. Close the windows form and back to The Vision Tool Form.

The complete code of the form is attached in the section A6.9 of the appendix.

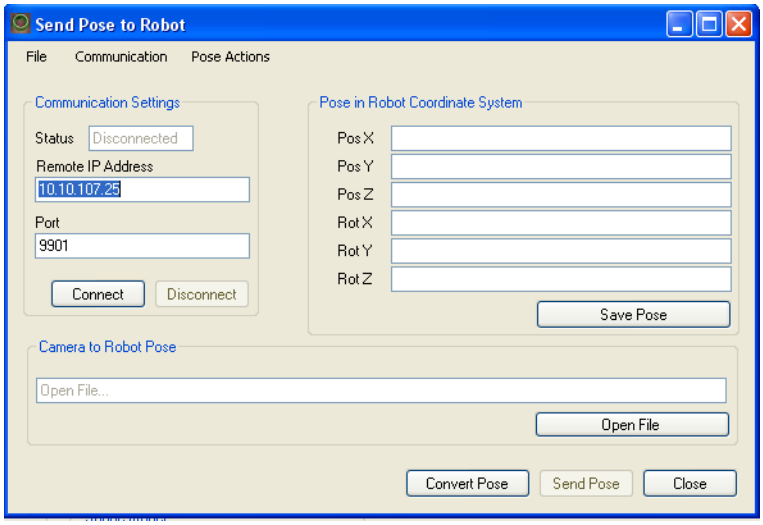


Fig. 122. - Communication Form

As well as in the two previous windows form there are some conditions that have to be achieved to run the Halcon procedures of pose conversion, connection, disconnection and send:

- The relation Pose has to present the correct file extension (.dat) to can convert the matched pose. The check structure it the same as the presented in the similar requirements in the other two windows forms.
- In order to ensure the correct open and close of the socket, the IP and Port manually introduced have to present a specific structure. For the IP, the string introduced have to present four different parts separated by the character "." And in every part the number introduced have to be in a range of 0<IPpart<255. For the port, it has to present only number in the string introduced in the text box.

```
If Len(txbPort.Text) <> 0 And Len(txbIP.Text) <> 0 Then
    Dim IPstring As String
    IPstring = txbIP.Text
    ip = txbIP.Text
    Port = Convert.ToInt32(txbPort.Text)
    Dim IPParts() As String

    IPParts = IPstring.Split(".")
    If IPParts.Length = 4 Then
        If IsNumeric(CInt(IPParts(0)) > 0 And CInt(IPParts(0)) < 255) Then
           If IsNumeric(CInt(IPParts(1)) > 0 And CInt(IPParts(1)) < 255) Then
              If IsNumeric(CInt(IPParts(2)) > 0 And CInt(IPParts(2)) < 255) Then
                If IsNumeric(CInt(IPParts(3)) > 0 And CInt(IPParts(3)) < 255)
                Then
                  'Run the procedure of connection
                  Call RunHalconConnect()
                  txbIP.Enabled = False
                  txbPort.Enabled = False
                  btnDisconnect.Enabled = True
                  btnConnect.Enabled = False
                  btnSend.Enabled = True
                  txbStatus.Text = "Connected"
                  IpMemo = txbIP.Text
                  PortMemo = txbPort.Text

                Else
```

```
                    Call MessageBox.Show("IP is not correct", Me.Text,
                    MessageBoxButtons.OK)
                End If
            Else
                Call MessageBox.Show("IP is not correct", Me.Text,
                MessageBoxButtons.OK)
            End If
        Else
            Call MessageBox.Show("IP is not correct", Me.Text,
            MessageBoxButtons.OK)
        End If
    Else
        Call MessageBox.Show("IP is not correct", Me.Text,
        MessageBoxButtons.OK)
    End If
Else
    Call MessageBox.Show("Introduce correct IP and Port", Me.Text,
    MessageBoxButtons.OK)
End If
```

- The last requirement uses a global variable *PoseConverted* to check that the matching pose has been converted before to send to the robot controller.

```
If Poseconverted = True Then
    Call RunHalconSend()
    Call MessageBox.Show("Pose sent succesfully", Me.Text,
    MessageBoxButtons.OK)
Else
    Call MessageBox.Show("Pose has to be converted", Me.Text,
    MessageBoxButtons.OK)
End If
```

6.5.2.6 Global Variables

Because of the procedures need to relation each other and work on the same variables (pose, shape model, etc.) a list of global variables has been created in another item called *"Module1.vb".*

The list of variable declared is the following:

- the directories selected in the different dialogs:
  1. *ShapeModelDirectory.*
  2. *CamParamDirectory.*
  3. *CamRobDirectory.*
  4. *ModelDirectory.*
  5. *FinalPoseDirectory*

- In order to control the creation of the 3d shape model:
  1. *ShapeRead.*
  2. *hv_ShapeModel3DID.*
  3. ShapeCreated.

- Control the status of the matching process:
  1. *CylingPoseMatch*.
  2. *hv_PoseTmp1.*
  3. *hv_NumPieces.*

- Show up the Match information in the Windows forms:
  1. *hv_Pose_Cylinder_in_RobBase.*
  2. *hv_PosX, hv_PosY, hv_PosZ.*
  3. *hv_RotX, hv_RotY, hv_RotZ.*

- In order to control the communication process:
  1. *IP.*
  2. *Port.*
  3. *Status.*
  4. *IPMemo.*
  5. *PortMemo.*
  6. *Socket.*

The code of the Model1.vb is outlined in the section *A6.10* of the appendix.

6.5.2.7 Halcon Error handling

The template offered by Halcon presents the feature of handling the error produced inside the Halcon procedures.

This errors are processed as exception in the Halcon procedures. This exception shows a message dialog with the information of the error and the options of break the running of the programme or continue the process.

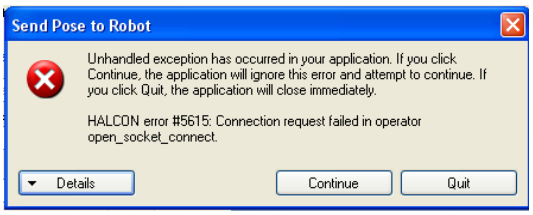


Fig. 123. - Examples of Halcon Error Handling

After the explanation of all the exported files and the different windows form created, we attach some pictures that show up the application once it is in running mode.

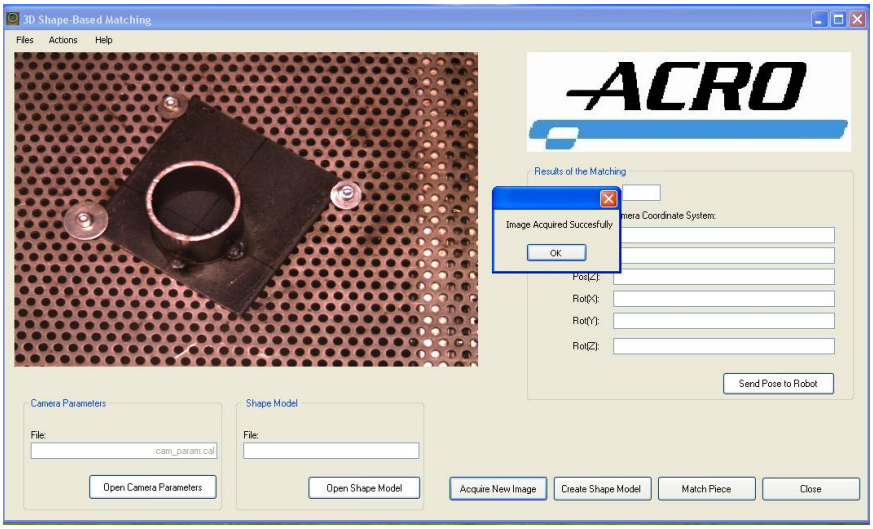


Fig. 124. - Vision Tool After finishing the Acquirement Process

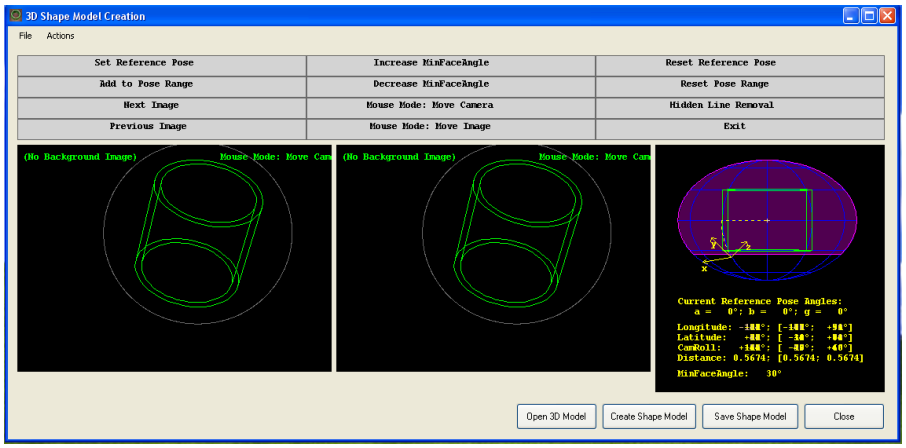


Fig. 125. - Shape Creation Form during 3d object model inspection

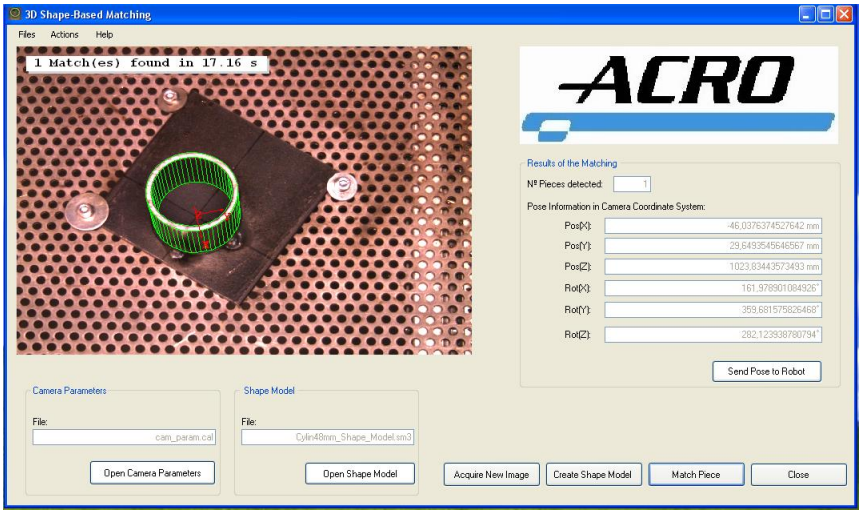


Fig. 126. - Vision Tool Form after matching process

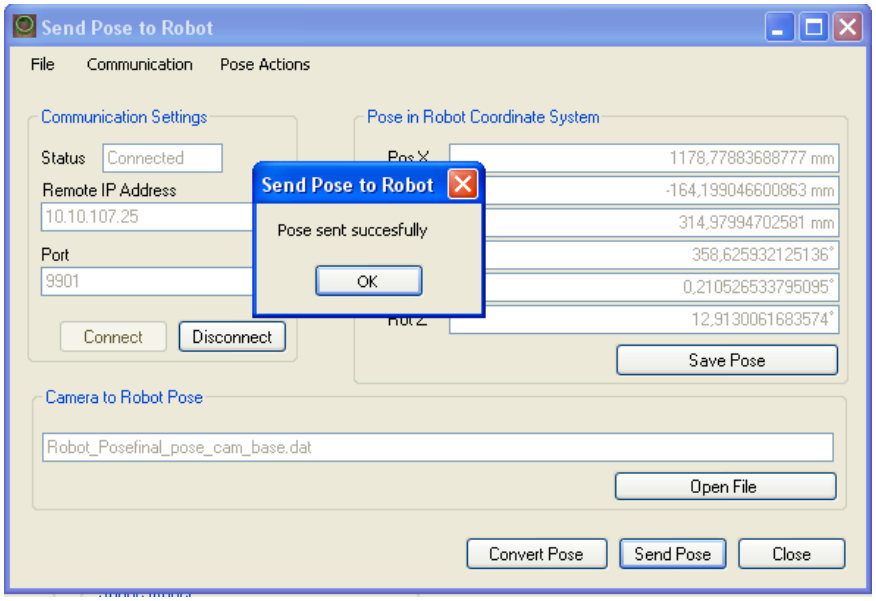


Fig. 127. - Communication Form after finishing the sending process

Chapter 7.

7. Automated Robotic Welding Cell

7.1. Final Results Exposition

After the vision system is finished and the communication between it, the robot controller and the welding system is properly developed we are in the situation of realising some recognitions and welds of the cylinder in different positions and orientations.

Once the cylinders have been properly matched and its pose information converted and sent using the "*3D Shape Based Matching.exe"* (visual basic application deployed) it is possible to present some picture of the welding results obtained in order to evaluate the validity of them and of the project.

Fig. 128. - a,b,c,d Cylinder welding results

As we can see in the previous image, the solution presents some variations in the final pose results that provoke some variations in the welding beam obtained. Despite this, we can establish that the accuracy level achieve is good enough to perform proper welds.

According, to the welding point of view. The results also present a high level in accuracy and quality even when this levels were not an initial objective of the project.

In conclusion, we are able to establish that the objectives that were laid out at the beginning of the project have been successfully achieved and in that way, the validity of the project is fully demonstrated.

7.2. Limitations of the solution

Although, all the objectives presented at the beginning of the project have been successfully achieved, the final tool obtained is no more than an initial prototype. Because of that, it presents some limitations that would be have to be solve before installing it in a real manufacturing process where robustness and repeatability are strong requirements.

Some of this limitations are presented in the following list:

- The process is only optimized for a cylinder of 48 mm of high and diameter, if the piece is different the laborious process of parameter optimization have to be boarded again.
- The system doesn't have the flexibility to change the models and the piece and obtain proper results.
- The reachability of the robot are badly injured by the welding torch and the wire-feeder. That provokes that the cylinder can only be welded following the same patron in different positions that limits the field where the cylinder can be properly welded.
- The 2D camera has a small field of view of the work-table, which limits highly the available welding positions of the cylinder.
- Using the 3D Shape Based Matching, the 2D camera tries to simulate the functionality of a 3D camera inside the pose range of the shape model. This obtains good results, but the levels of accuracy and repeatability are going to be always lower than using a real 3d vision system.
- The welding techniques present some faults.

In section 8 of this paper, future upgrades of the solution will be described in order to eliminate completely or reduce at maximum this limitations.

7.3. Budget of the project

One of the initial objectives of the project was achieve a proper economic solution, in order to demonstrate that this has been fully achieved a budget of the project with the principal parts of every system is outlined.

In this budget there is not all the components of the project, only the especially bought to develop the solution. The budget as well as the project can be divided into three system: Robotic System, Welding System and Vision System.

Automate Robotic Welding Cell Budget		
2014 Prices		
Robotic System		
Component	**Detailed Description**	**Price (€)**
Robot and Robot Controller	KUKA KR5 ARC / KUKA KR C2 (software KUKA.KSS 5.5 included)	17000
Application Software	KUKA Arct.Tech Digital Software Package A.20	805
Connection Software	KUKA Connect PC/KRC	950
Sensor Interface	KUKA Robot Interface	3200
Total		**21955**
Welding System		
Component	**Detailed Description**	**Price (€)**
Welding Power Source	Fronius TPS 4000	5782
Cooling Unit	Fronius Fk 4000 R	1162
Carriage	Carriage PickUp	397
DeviceNet Kit	I-Kit DeviceNet	1872
Wire-Feeder	VR 1500 4R/W/E	1804
	I-Kit Cover VR 1500	106
	Mounting VR1500 MO/K	193
	I-Kit blow out torch Basic	63
	I-kit PMR4000 PullMig	291
	QuickConnect DFS BL VR1500	64
Hose Pack Connection	Connec. Hose pack W/8m/95mm2	966
Weld Torch	Robacta500 36˚	355
	Robacta Drive Ext. W/E/1,5m	3245
	Collision box XL	778
	Adapter LK40	135
Welding Wire	Wire coil mounting on robot KUKA	373
	QuickConnect DFS BL VR1500	64
	Wire Feed hode ID 8,0/OD12mm 1m	2,7
Torch cleaner cell	Binzel Torch Cleaning Station	450
Ground Cable	GroundCable 95mm2/10m	329
Total		**18431,7**
Vision System		
Component	**Detailed Description**	**Price (€)**
Camera	UI-1465LE-C CS-Mount/C-Mount adaptor incl. (2048x1536 pixel 11 CMOS)	404,1
Lens	Fujinon 1:1.4/25 mm HF25HA 1B	152,31
USB Cable	USB 2.0 cable (5m)	12
HALCON Sowtware Package	Runtime (for a company) Dongle included	1450
	Development license (for a company)	4500
	Development for an institution	1750
Total		**6518,4**
Total Project Price	**46.905,10 €**	

Fig. 129. - Project Budget

Chapter 8.

8. Future Upgrades and Conclusions

In this section we present some possible solutions that could be applied in the project in order to improve its characteristics and solve its limitations. Furthermore, because of all the objectives of the project have been achieved, some conclusions are commented about the work I have realised and how I felt during my stance in ACRO.

8.1. Future upgrades of the Welding Cell

Following the limitations established in the section 7.2, the majority of them resides in the accuracy obtained during the matching process. Some possible solutions to increment the accuracy would be:

- A real 3D camera could be applied, changing the matching process to another with higher results in accuracy such as 3D Surface-Based Matching.
- Introduction of a laser system to apply a Sheet of Light recognition process. With this solution, the challenge of the laser movement over the piece appears. This could be solved through: use a second robot to move the laser (including the possibility of sequence the process of recognition in big pieces or pieces with complex parts). Or use a PLC and a motor to control and manage a fixed movement of the laser.
- Introduction of simulation software of KUKA (KUKA.SimPro equipped with KUKA.ArcTech) to check the reachability of the robot during the welding process in big pieces or during complex beams and study the time cycles.

Although the welding quality was not an initial objective of this project, it would have to be improved in order to introduce the project in a manufacturing environment. Some of the possible steps that could applied are:

- A more accurate adjustment of the welding parameters in the power source saved in the different JOBs.
- Correct selection of the welding wire and the protective gas to the kind of welding that is going to be performed.
- Re-definition of the Fronius signals in the robot controller file *Config.dat* to use all the potential of the Fronius TPS 4000.

Finally, once all the limitations have been solved, futures features may be added to the project:

- Introduction of safety measures that control the interruption of the welding process or the robot movement if some external objects enter in the work-space. This measures could be carried out with fences. Also it is possible to use a conjunction of lasers and receptors in order to detect the intromission of external bodies and to not limit the use of the workspace when the project is not working.
- Creation of a better interface that lets the user select the star and end points of the welding beams through clicking the mouse in the image of the piece.
- Real-time monitoring and control of the welding process that ensure the modification of the welding parameters and the beams during the performance of the welding process. To achieve that some of the solution presented in section 2.4.4.1 could be applied (through-the-arc-sensor, Laser Scanning solutions, etc.).

8.2. Conclusions

At the beginning of this project I felt a little overwhelmed, because I hardly had work with a real industrial robot during my studies in Cartagena and I had never studied Machine Vision. However, now I am glad to establish that the project has been finished successfully and all the initial requirements and objectives are satisfied.

Although, the project final results is no more than a simple welding task, the form of obtained present a complete robotic and vision application. Even more, this solution presents a great capability of flexibility to integrate future upgrades of the project in order to increment the potential of the solution. All of that make me very proud of my development work.

Attending to my internship in ACRO, this stance has been my first experience of a real engineering work outside of my university in Spain and the result could not have been better for me. I am very satisfied of the result obtained, the knowledge obtained and the most important, the new form of thinking that this internship have gave me in order to solve problems by myself and to not hesitate to contact with different supply companies to receive advice and information.

Finally, I want to thank again all the staff of ACRO, they have helped me without any reserves every time I asked for their help and despite the language barrier (English is neither my mother tongue nor them), I am very grateful that they have tried their best in order to understand me and create a satisfactory work environment.

Chapter 9.

9. Bibliography

- "*White Paper Sensor Based Adaptive Arc Welding*". (2009). ABB.
- "Welding with vision. Vision systems error-proof, add flexibility to robotic welding" (2009). Terry Tupper. FANUC Robotics America Inc.
- "*Welding Robots. Technology, System Issues and Applications*". (2006). Pires, Loureiro, Bölmsjo. Springer. ISBN: 978-1-85233-953-1.
- "*KR 5 ARC Specification*". (2011). KUKA Roboter GMBH. Pub Spez KR 5 arc en.
- "*Controller KR C2 Specification*". (2011). KUKA Roboter GMBH. Spez KR C2 sr V4 en.
- "*KUKA System Software 5.5 Operating and Programming for End Users*". (2010). KUKA Roboter GMBH. KSS 5.5 END V2 en.
- "*Software KR C2/C3 Expert Programming. KUKA System Software (KSS*)". (2003). KUKA Roboter GMBH. ProgExperteBHR5.2 09.03.00 en.
- "*KR C ArcTech Digital 2.3 Configuration".* (2006). KUKA Roboter GmbH. ArcTechDig_P_R23 04.07.00 en.
- "*TransPulsSynergic 4000 Operating Instructions* ". (2007). FRONIUS INTERNATIONAL GMBH. ISBN: 4204260001, EN 012005.
- "*FK 4000 Operating Instructions*". (2007). FRONIUS INTERNATIONAL GMBH. ISBN: 42,0426,0017, EN. 012008.
- *"VR 1500 Operating Instructions*". (2007). FRONIUS INTERNATIONAL GMBH. ISBN: 42,0426,0006,EN 009-10042012.
- *"FRONIUS Robacta, Robacta Drive Brochure".* (2007). FRONIUS INTERNATIONAL GMBH.
- *"HALCON The Power of Machine Vision, Quick Guide*". (2012). MVTec Software GMBH.
- *"HALCON Solution Guide I,II-a,II-b,III-c".* (2012). MVTec Software GMBH.
- *"HALCON Programer's Guide".* (2012). MVTec Software GMBH.
- *"HALCON/HDevelop Reference Manuall".* (2013). MVTec Software GMBH.

- "*DeviceNet Network Configuration, User Manual*". (2011). Rockwell Automation. Publication: DNET-UM004B-EN-P.

- "*DeviceNet Technical Overview*". (2001). Open DeviceNet Vendor Association (ODVA). Publication: PUB00026R1.

- "*Programming in VisualBasic 2010. The Very Beginner's Guide".* (2010). Jim McKeown. Dakota State University. Cambridge University Press. ISBN: 978-0-521-89653-5.

- Web Bibliography

- www.kuka.com.
- www.microsoft.com
- www.msdn2.microsoft.com
- www.mvtec.com/halcon
- www.ueye.com
- www.fronius.com
- www.ab.rockwellautomation.com
- www.odva.org

Figure Index