UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI INGEGNERIA ELETTRICA ELETTRONICA E INFORMATICA

DOTTORATO DI RICERCA IN INGEGNERIA ELETTRONICA, AUTOMATICA E DEL CONTROLLO DI SISTEMI COMPLESSI
(XXIV CICLO) - PH.D. THESIS

MULTISENSOR DATA FUSION FOR ROBOTIC CONTROL

FILIPPO BONACCORSO

TUTOR: PROF. G. MUSCATO
COORDINATOR: PROF. L. FORTUNA

To Nino and Lina

Preface

Nowadays robots perform more and more tasks, from simpler ones such as automatic domestic vacuum cleaner, to highly skilled ones such as tele-surgery. To do so a robot continuously needs to know what to do and what has been done. A user controls robots through a system controller, which processes both his commands and information from the environment.

These information are collected by a measurement system, often called observer, which bring back them to the system controller. Measures are the information used to perform the control. The simpler measurement system can, in general, be considered as made up of two elements:

1. A sensor or transducer which is an element that produces a signal relating the quantity being measured. Sensors are elements that when subject to some physical change experience a related change.
2. A signal conditioner which takes the signal from the sensor and processes it to make it suitable for the specific application. The signal may be, for example, too small or too noisy.

Different kinds of sensors are used in robotics, such as position, velocity, force sensors and so on. Each application needs the right sensor to be chosen, the choice must satisfy different criterions such as performance, cost and feasibility. In this way both static, as range or sensitivity, and dynamic characteristics, as response time, of the sensor have to be considered.

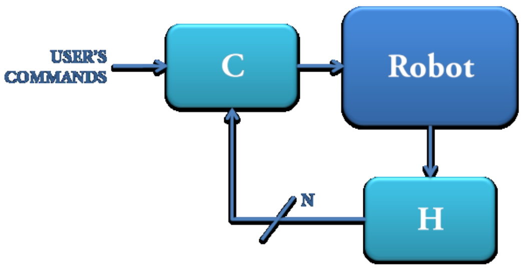
In robotic applications there are two main subclasses of sensors: internal and external. The former bring information about the robot; the latter bring information about the environment. In a mobile robot, for example, encoders are internal sensors, while ultrasound sensors are external.

Multisensor data fusion seeks to combine data from multiple sensors to perform inferences that may not be possible from a single sensor. Different sensors have different strengths and weaknesses, so fused data from multiple sensors provides several advantages over data from a single sensor:

- A wider and more accurate range of information by combining data from different types of sensors.
- Redundancy.
- Increased reliability.
- Weaknesses compensation.
- Sensors failure detection and handling.

Typical applications that can benefit from multiple sensors are, first of all, mobile robot navigation, target tracking, aircraft navigation and industrial tasks control.

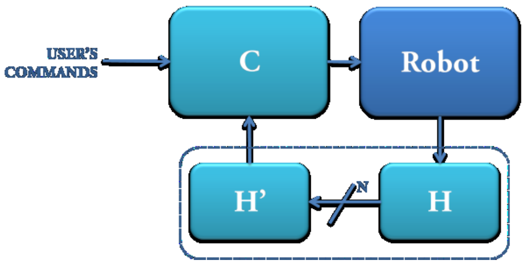
There are a number of different ways to integrate or fuse information provided by multiple sensors.



Multisensor input system controller

The simplest approach to multisensor data fusion, is to use the information from each sensor as a separate input to the system controller. This approach may be the most appropriate if each sensor is providing information concerning completely different aspects of the environment. The major benefit gained through this approach is the increase in the extent of the environment that is able to be sensed. The only interaction between the sensors is indirect and based on the individual effect each sensor has on the controller and so on the whole robot.

If there is some degree of overlap between sensors, concerning some aspect of the environment they are able to sense, it may be possible for a sensor to directly influence the operation of another one. In this way the value of the combined information that the sensors provide, is greater than the sum of the value of the information provided by each sensor separately. This synergistic effect can be achieved either by using the information from one sensor to provide cues, or guide the operation of other sensors, or by actually combining or fusing the information from multiple sensors.



Synergistic multisensor fusion

The whole research activity has dealt with two main topics: the *RAPOLAC Project* and the *Mobile Robots.* The former are discussed in the first four chapters while the latter in the following four.

The *Chapter 1* introduces Rapid Prototyping, dealings with its fundamentals, advantages and differences with respect to traditional manufacturing methods. After that the Shaped Metal Deposition process will be introduced.

In *Chapter 2* the estimation of the arc length, in the field of welding system will be covered.

Chapter 3 will cover the important aspect of the control of the SMD process. In particular the control strategy developed will be presented.

Chapter 4 concludes the first part of this thesis, showing some aspects concerning modeling and control of manipulators.

Chapter 5 deals with all the robotic mobile platforms involved in the research activity. It shortly describes the hardware and the sensor suites each robot is equipped with.

Chapter 6 introduces the environment used to develop application for mobile robot: *Microsoft Robotics Developer Studio*.

Chapter 7 introduces the sensor suite used, describing all the adopted sensors. This chapter will also shortly present how they are introduced in the Microsoft Robotics Developer Studio environment.

Chapter 8 concludes the second part of this thesis. It introduces how robots are simulated and used in real applications using Microsoft Robotics Developer Studio. Moreover it presents the main output of all the research activity this thesis deals with, that are algorithms suitable for self-localization, navigation and obstacle avoidance tasks for mobile robots.

I would like deeply thank the Ph.D. Coordinator Prof. Eng. Luigi Fortuna for his effort.

My very special thanks go to my Tutor Prof. Eng. Giovanni Muscato for his constant support and suggestions. He warmly taught me how to be a researcher. I hope I’m right!

Many special thanks go to my colleagues Donato Melita and Luciano Cantelli, I’m very glad to have work with them.

A special thought goes to all my laboratory colleagues.

AHRS	Attitude And Heading Reference System
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CC	Confidence Circle
CCR	Concurrency And Coordination Runtime
CLR	.Net Common Language Runtime
CNC	Computer Numerical Control
DLL	Dynamic Link Library
DSS	Decentralized Software Services
EKF	Extended Kalman Filter
ENH	East, North And Height
FPGA	Field Programmable Gate Array
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GTAW	Gas Tungsten Arc Welding
ICP	Iterative Closest Point
IDC	Iterative Dual Correspondence
IMRP	Iterative Matching Range Point
IMU	Inertial Measurement Unit
IMU	Inertial Measurement Unit
LLA	Latitude, Longitude And Altitude
LRF	Laser Range Finder
MCS	Maximum Common Subgraph
MIG	Metal Inert Gas
MRDS	Microsoft Robotics Developer Studio
MRPT	Mobile Robot Programming Toolkit
NI	National Instruments Co.
OA	Obstacle Avoidance
RAPOLAC	Rapid Prototyping Of Large Aerospace Components
RISC	Reduced Instruction Set Computer
ROS	Robot Operating System

RP	Rapid Prototyping
RPY	Roll, Pitch And Yaw
RPY	Roll Pitch Yaw
RTK	Real-Time Kinematic
SH	Step Height
SL	Self Localization
SLAM	Simultaneous Localization And Mapping
SM	Scan Matching
SMD	Shaped Metal Deposition
TIG	Tungsten Inert Gas
TS	Travel Speed
UI	User Interface
USS	Ultra Sonic Sonar
VOPF	Virtual Obstacle Potential Field
VPL	Visual Programming Language
VSE	Visual Simulation Environment
WF	Wire Feeder Rate
WY	Welding-Pool Width

HI HII

Chapter 1 Shaped Metal Deposition

This Chapter will shortly introduce Rapid Prototyping, dealings with its fundamentals, advantages and differences with respect to traditional manufacturing methods. Concepts regarding Rapid Prototyping can be easily extended to manufacturing processes like the Shaped Metal Deposition process which will be also introduced. The work presented here and in following three chapters has been carried out in the frame of the RAPOLAC European project [1], [2].

1.1. Rapid Prototyping

As reported by [3], a prototype should be defined as:

An approximation of a product (or system) or its components in some form for a definite purpose in its implementation.

This definition is the more general is possible, the key idea comes from the usual well known concept of the prototype being a physical object, but it covers all possible kind of prototypes like it could be an algorithm or a model representing a process. Besides prototyping is the process of realizing prototypes.

This and the following chapter will deal with the process of prototyping of mechanical parts.

As it is well known prototypes have an important role in the development of products, they are really useful tools for experimentation, testing and proofing. Prototypes can be useful in demonstrating ideas or concept and help in scheduling the product development process.

In product development, whichever it may be, time has become an important factor in determining the success or the failure of a product. As a matter of this the term *time-to-market*, for instance, is nowadays widespread. In this point of view, Rapid Prototyping (RP) technologies are the best because allow accurate physical prototypes to be build quickly and repeatedly with high precision.

The development of RP is closely tied in with the development of applications of computers into the industry. As a matter of this the emergence of RP systems could not have been possible without the introduction of Computer Aided Design (CAD) which has lead to Computer Aided Manufacturing (CAM) through Computer Numerical Control (CNC) machines.

A short history of the growth of RP and related technologies, [3] [4] [5] [6], is reported in Table 1.1.

Table 1.1 Development of RP

Year	Technology
1770	Mechanization
1946	First computer
1952	First numerical control machine tool
1960	First commercial laser
1961	First Commercial robot
1963	Early version of CAD
1988	First Commercial RP system

The basic approach to RP is the same, regardless of the technique involved in; it can be described by the following three steps:

1. A model which represents the physical part to be built is created through CAD;
2. The solid or surface model to be built is converted into a particular format approximating the shape of the part;

3. A Software analyzes data from previous step and act on the manufacturing system in order to build the part;

Nowadays RP equipments allow to build small production quantities with, sometime, an accuracy close to those made by machining. The traditional manufacturing methods, such as machining, are based on the material removal from the work piece, in order to obtain a desired final shape. In same case the melted material adopted has to be putted in a mould and, after the solidification, the prototype has to be machined in order to increase the surface finish. So, traditional manufacturing, in the worst case, starts by solid work-piece and by machining obtains the desired shape.

Otherwise, as it will be clearer later, there are many benefits of RP such as: reduced time-to-market, increased feasible complexity of the part and, for sure, reduced production costs.

RP systems can be classified in different ways; one of these is to classify RP systems by the initial form of the material the prototype is built with. Such a classification brings to three categories: liquid based, powder based and solid based. Further details can be found in [3]. The one we will deal with, the Shaped Metal Deposition process, belongs to the last one.

1.2. Shaped Metal Deposition

The Shaped Metal Deposition process (SMD) is a novel manufacturing technique developed and patented by Rolls-Royce ten years ago in order to produce mechanical parts directly from CAD models. The innovative aspect of the SMD process is a reversed production philosophy. Traditional manufacturing methods like machining are based on the material removal from the work piece in order to obtain a desired final shape, as it is shown for example in Figure 1.1. There are several evident drawbacks in this process, as the large waste of material in scraps and the consequent increase of the costs, depending of the material used.

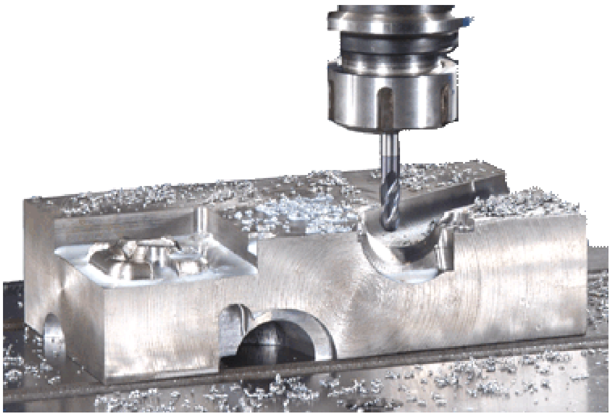


Figure 1.1 An example of machining

The SMD process, represented in Figure 1.2, reverses this destructive production philosophy and works by adding progressively, in a layer-by layer fashion, the material to the final work piece, in order to obtain the desired component final shape. This means that effectively no process scraps are produced thus minimizing the material used to the strictly amount required, [7] [8].

The process typically works using a welding system [9], where the injected material comes from the wire feeder. The melted metal is deposited in to shape the work-piece until the desired profile is reached.

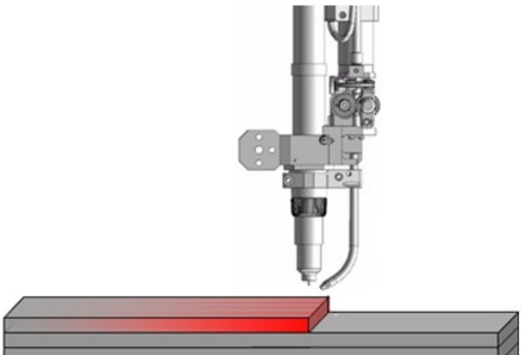


Figure 1.2 The Shaped Metal Deposition Process

One of the main advantages of the process is the substantial reduction of the inventory costs since the only inventory held is the metal wire. The parts can be built to order. Another significant advantage is the reduction of 60% in the lean-time necessary to produce the parts so that it is ideal for use in prototype production as well as in manufacturing. Finally microstructure analysis of produced parts confirms that there are no disadvantages from the quality point of view [10].

In a fascinating and futuristic view, manufacturers could keep just CAD models of their products and built them just when it is needed.

The work presented here and in the following chapters, as mentioned in the introduction, has been carried out in the frame of the RAPOLAC European project [1] [2]. RAPOLAC is the acronym of *Rapid Production of Large Aerospace Components*. The key idea behind the RAPOLAC project is the use of SMD in order to build aerospace components which are usually made by titanium. To do so, a manipulator is used to move the welding torch in order to follow the shape of the desired part to be built. Layer by layer, a constant amount of melted metal is deposited as shown in Figure 1.2.

As a matter of the complexity of the whole system, RAPOLAC aims also to investigate upon an automatic control system to free the operator from constant monitoring and manually acting on the process parameters.

1.3. Gas Tungsten Arc Welding

Fusion welding, Figure 1.3, is a process that uses fusion of the metal to make the weld. The three major types of fusion welding processes are: gas welding, arc welding and high-energy beam welding. The heat sources for welding processes are a gas flame, an electric arc and a high-energy beam, respectively. The power density increases from a gas flame to an electric arc and a high-energy beam, [9].

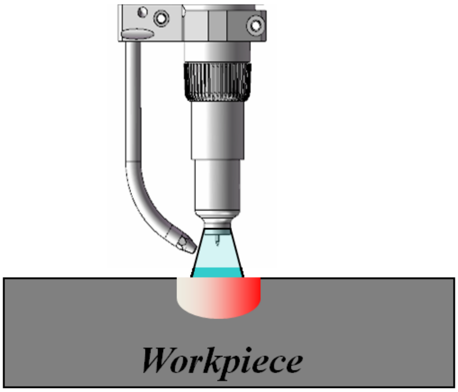


Figure 1.3 Gas Tungsten Arc Welding

The welding system we will deal with is named Gas Tungsten Arc Welding (GTAW). This welding technique melts metals by heating them with an electric arc established between the work-piece and a non consumable tungsten electrode connected to the power source: for this reason this process is often called Tungsten Inert Gas (TIG).

Both GTAW and TIG will be used to indicate the same welding process.

The melted region of the workpiece is called welding pool. The workpiece is connected to the power source through a different cable. The torch holding the electrode also emits a gas flow to shield the weld metal to avoid oxidation. An external system supplies metal, through a solid wire, to the melted region. Evidently the power emitted is related to the arc current.

Sometime some works cited deal with a different kind of welding equipment, called Metal Inert Gas (MIG). This welding system differs from the previous one for the electrode used which is, in this case, the same wire used to weld.

The transfer of heat in both this kind of welding is a well known aspect of welding theory, it is usually called "*travelling heat source*", [9] [11] [13] [14].

The travel of heat in a weldment, shown in Figure 1.4, is governed by the following three-dimensional diffusion equation:

$$\rho(\mathbf{T})C(\mathbf{T})\frac{\partial\mathbf{T}}{\partial t} = \nabla(\lambda(\mathbf{T})\nabla\mathbf{T}) - \nabla(\rho(\mathbf{T})C(\mathbf{T})\mathbf{vT}) + Q \qquad \textbf{Eq. 1.1}$$

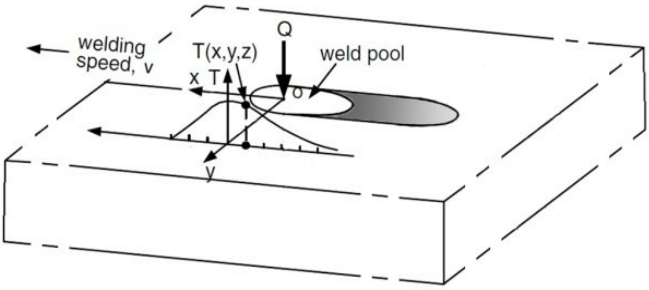


Figure 1.4 Travelling Heat Source

where:

***T*(*x*,*y*,*z*)** the temperature of the workpiece [°K]

x	coordinate in the direction of welding [mm];
y	coordinate transverse to welding direction [mm];
z	coordinate normal to workpiece surface [mm];
ρ	density of the metal [g/mm^3];
λ	thermal conductivity [J/mm s^{-1} K^{-1}];
C	specific heat [J g^{-1} K^{-1}];
v	speed of the torch often called Travel Speed (TS);
Q	Input heat rate moving at speed ***v*** [J/mm^3 s^{-1}];

The earliest and most used solution for Eq. 1.1 was founded by Rosenthal in 1938, [15]. The keys to Rosenthal solution are: the assumption of pseudo-steady state point heat source moving at a constant speed *v* on the surface of a semi-infinite plate with ***ρ, C*** and ***λ*** constant.

The solution for Eq. 1.1 is given by:

$$T - T_0 = \frac{Q}{2\pi\lambda R} e^{-\frac{v(w+R)}{2\alpha}} + Q \qquad \textbf{Eq. 1.2}$$

where:

T₀	initial temperature of the workpiece;
R	distance from the center of the arc $R=(w^2+y^2+z^2)$;

w *w=x-vt*;

α thermal diffusivity *α=λ/ρC*;

For the SMD process it is useful to consider a 2D solution on the ***xz***-plane. Eq. 1.2 can be rewritten as:

$$\frac{\partial T}{\partial t} = \alpha\left[\frac{\partial T}{\partial x} + \frac{\partial T}{\partial z}\right] - v_x\frac{\partial T}{\partial x} + Q \qquad \textbf{Eq. 1.3}$$

whose solution leads to the diagram reported in Figure 1.5.

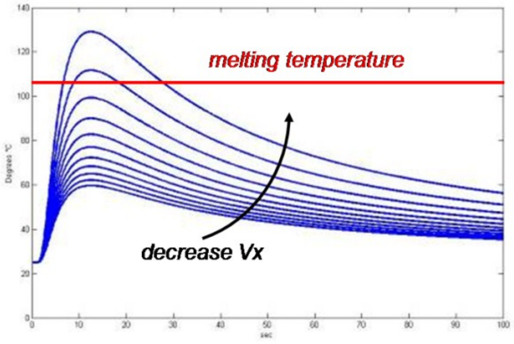


Figure 1.5 Rosenthal simplified solution on xz-plane

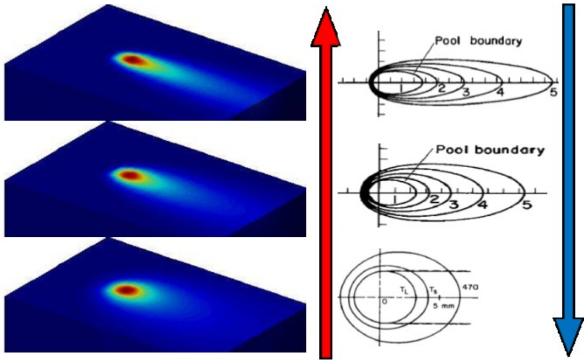


Figure 1.6 Welding pool section for different TS

Figure 1.5 shows the temperature of a generic point versus time while the point heat-source moves on. The temperature rises, the lower is the speed the higher is the temperature reached, until it rises over the melting point. The regions of the workpiece that reach the melting point of the metal will be melted.

Figure 1.6 shows how speed and also current change the welding pool geometry; the red arrow stands for increasing TS while the blue arrow stands for increasing current. Thus the lower is the travel speed, or the higher is the current, the thicker is the welding pool.

Several works related to arc welding could be found in literature, for example [16] deals with a complete robotic system for arc welding. However no one concern concrete implementations of control strategies on deposition processes.

1.4. Process Overview

A more detailed overview of the system is shown in Figure 1.7 while in Figure 1.8 the robot holding the welding torch (on the left) and the welding machine (on the right) are shown.

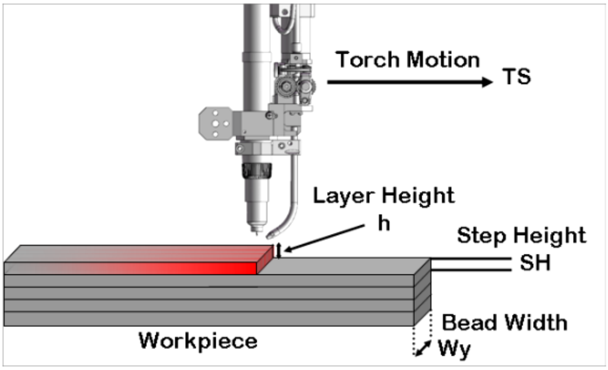


Figure 1.7 SMD overview

As said before the current and the travel speed affect the welding pool geometry, especially its width W_Y. This width is an important process parameter because it directly acts on the deposition. Indeed, after the torch has moved on, the welding pool solidifies and the width of the solidified material is thereabouts W_Y.

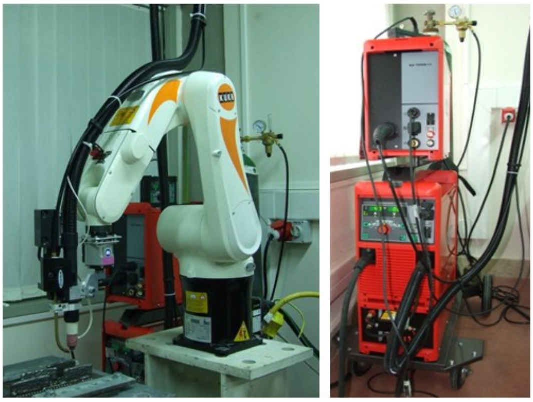


Figure 1.8 Plant overview

Another crucial aspect, from a mechanical point of view, is the height *SH* of each layer deposited since the thinner is the layer deposited the higher is the resolution of the final workpiece shape; thus it is important to precisely control the exact amount of material deposited in each layer [17] which is indicated by *h* in Figure 1.7.

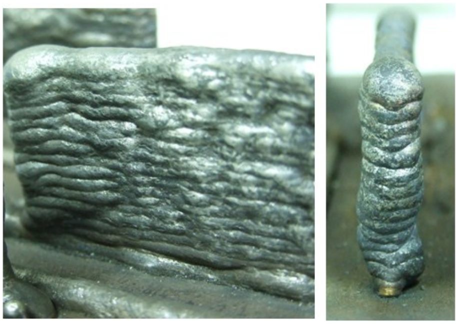


Figure 1.9 A simple process of deposition

This amount is mainly connected to the metal coming from the Wire Feeder of the welding machine indicated as Wire Feeder Rate (WF). This must be equal to the desired step height, in order to avoid depositing too much or too few material thus causing the system to become unstable.

Figure 1.9 shows a simple process of deposition. In this example it's clearly identifiable each layer deposited by means of the irregularities on the surface of each one. These are due to the involved plant which is a laboratory reproduction of the SMD and so it is not shielded from the external atmosphere. Consequently the surface is subject to a rapid oxidation. However the real SMD plant is housed inside a sealed chamber with argon gas purity of 99.999% which avoids oxidation, leaving clearer surfaces.

As a matter of these considerations the working variables needed for the control of the SMD process have been identified in four fundamental key parameters which are:

- The heat source provided to the process by the welding machine through the arc current *I*.
- The injected material mass rate indicated by the Wire Feeder rate *WF*.
- The torch speed indicated by the Travel Speed: *TS*.
- The desired thickness of each layer indicated by the step height *SH*.

1.5. Summary

This chapter has shortly introduced the RAPOLAC project dealing also with the SMD process and its differences with respect to traditional manufacturing methods.

Chapter 2 Sensing the Arc length

The previous chapter has shown how the SMD process works. Prototypes are built by adding progressively the material to the final work piece until the desired shaped component is obtained, as has been shown in Figure 1.7.

Several works related to arc welding automation could be found in literature, [18] [19] but, as far as we know, no one concern the automation of SMD process. Until today such a process needs an operator to constantly monitor process parameters and act on the process in order to keep it around a stable working point.

The basic idea for setting up an automatic controller is to measure by video feedback [17] the amount of material deposited on each layer, as said before, must be equal to the step height imposed for the process.

If too few material is deposited, the electric arc could become “long” and the power transmitted to the workpiece decreases, reducing welding pool dimensions. Otherwise if too much material is deposited turns up a risk of contact between the tungsten electrode and the workpiece, causing the electric arc to turn off.

According to this information, the WF is controlled in order to adapt the incoming material mass for each layer.

The *WF* represents the value corresponding to the stable process working point, in which all the material coming from the wire feeder is deposited in each layer, ensuring the stability of the process. The layer deposited height is indicated by *h*, as shown in Figure 1.7; in order to keep the process stable, as said many times before, the height *h* must be equal to the desired SH.

The aspects related to the control of the process will be shown on the following chapter, hereafter the estimation of the arc length will be covered. This choice is done because of the arc length *l* is physically related to *h* in fact, if a planar surface is supposed for the layer below the sum of *h* and *l* must be constant during the position of a new layer because of, as usual, the welding torch moves in a coplanar fashion with each layer.

Experienced human welders have sometimes been the best available closed-loop process controller in challenging welding applications. Their feedback comes from acoustic emissions and image processed by their brains. Improvements in process feedback and sensing had successfully achieved well closed-loop control for welding processes, [18] [19].

One such feedback parameter in GTAW is acoustic emissions

There are different possibilities to measure layer height [11], these are all connected to the arc length and, as summarized in Figure 2.1, are the arc emitted light, the arc emitted sound and the arc voltage.

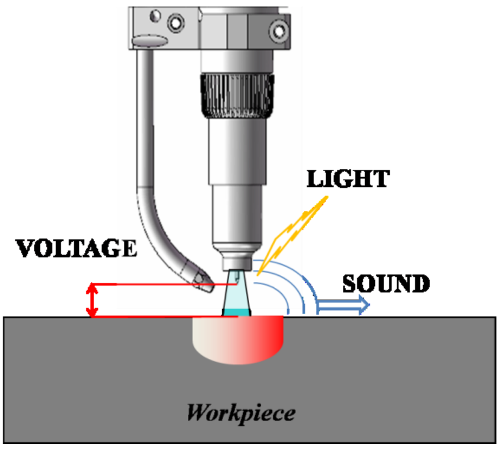


Figure 2.1 Sensing the arc length

2.1. Arc Light

In [20] a complete system to measure the arc length starting from the spectrum emitted by the arc is presented. In [21] a simple photodiode is used to automatically control a welding process. More generally a smaller portion of the spectrum emitted by the arc is analyzed. Theoretical and experimental results lead to the conclusion that the arc length is connected to the intensity of the emitted light.

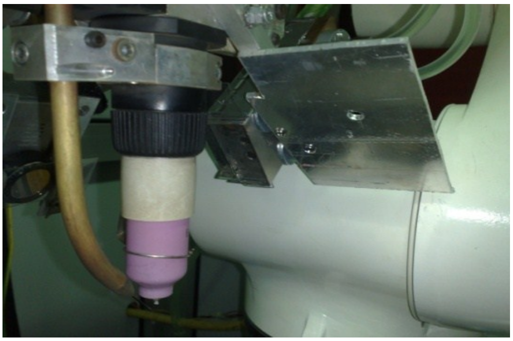


Figure 2.2 IR Interceptor

The system adopted is based on two IR phototransistors. Two sensors are used in order to increase redundancy. A box is connected to torch, like that shown in Figure 2.2, in order to intercept the light emitted by the arc.

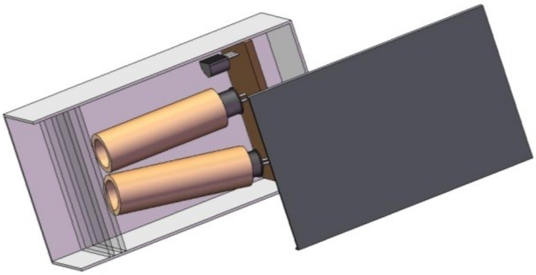


Figure 2.3 IR Interceptor detail

The box, represented in Figure 2.3, houses the IR sensors and also a temperature sensors in order to compensate the deviation of the measure. An external cooling system is used to avoid high and dangerous temperature due to the heat coming from the electric arc.

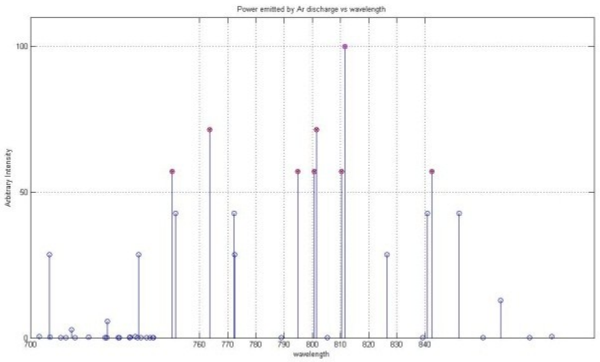


Figure 2.4 Argon gas discharge

The choice of the IR sensors was made to measure the power emitted by the Argon gas because like is shown in [20], the intensity of argon atom lines is related to the arc length and is independent of variation of welding conditions.

The Argon spectrum, reported by [22] and shown in Figure 2.4, has more power in the region between 750 and 1000 nm.

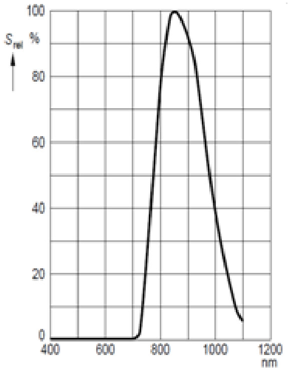


Figure 2.5 SFH313FA relative spectral sensitivity

As matter of this a set of different sensors was tested in order to find the better one in term of noise and linearity. Two Osram SFH133FA were selected. An optical filter has been used to reduce the power collected by the sensor in order to avoid saturation.

As it is shown in Figure 2.5, the sensitivity of the sensor is greater in the region of the spectrum with the higher power that has been shown in Figure 2.4.

Figure 2.6 reports the experimental relation found between the arc length and the power emitted by the arc with a current of 150 A.

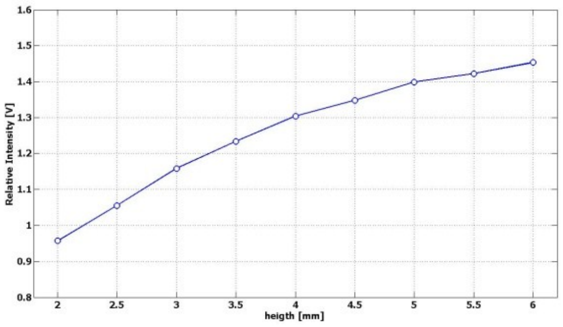


Figure 2.6 Power emitted versus arc length

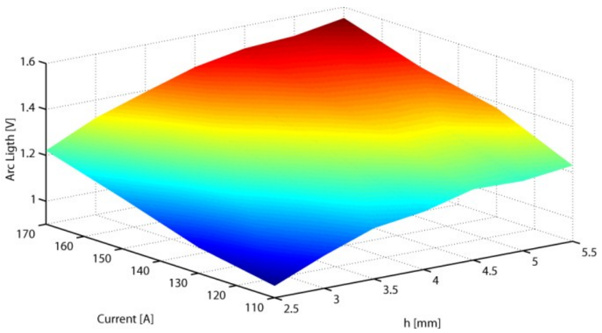


Figure 2.7 Light emitted by the arc, as measured by the IR sensors, versus Current & Arc length.

On the basis of an accurate experimental characterization the relations between the light intensity and the current/height, shown in Figure 2.7, has been approximated to be polynomial. The voltage output of the sensor, temperature compensated, has been used as the measure of the intensity.

More information among the use of arc light will be given in paragraph 2.5.

2.2. Arc Sound

As reported in [23] there have been relatively few studies performed in this area, while it is agreed amongst professional welders that the sound from an arc is critical to their ability to control the process. The arc-sound provides as much useful feedback as vision in controlling the process such that a relationship can be drawn between welding parameters and acoustic spectral characteristics.

In 1967, Jolly [24] published a study on GMA welding acoustics as an indicator of the occurrence of cracks in a welded joint. He concluded that sound pressure increases with arc length (voltage) and welding current. This relationship was formally defined in 1979 by Drouet and Nadeau [25], as:

$$S_a(t) = K[V(t)I(t)] \qquad \textbf{Eq. 2.1}$$

with $S_a(t)$ as a time integral of acoustic signal and K as a constant dependent by geometrical and physical factors.

Acoustic data from 0.05 to 18 KHz were collected through a microphone. Investigations that have been performed, however, have been met with poor results due to extraneous background noises or inadequate evaluation of the signal spectral content. Preliminary experimental data shows a linear relationship between the arc length and the power emitted at higher frequencies.

2.3. Video Processing

In order to carry out information by video feedback, two different setups have been used: one based on a standard camera and one with a *linlog* camera. The former uses a CCD camera with a 640x480 resolution with 24-bit colour and 30 fps refresh rate [26], [27]. The camera has been equipped with a composite filter system and a cooling system in order to preserve its operative conditions, because the melting point of steel is very hot (1100° C).

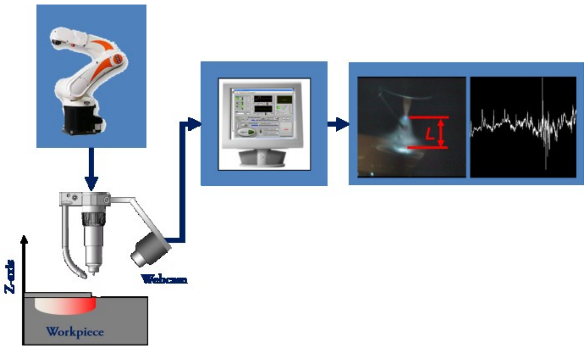
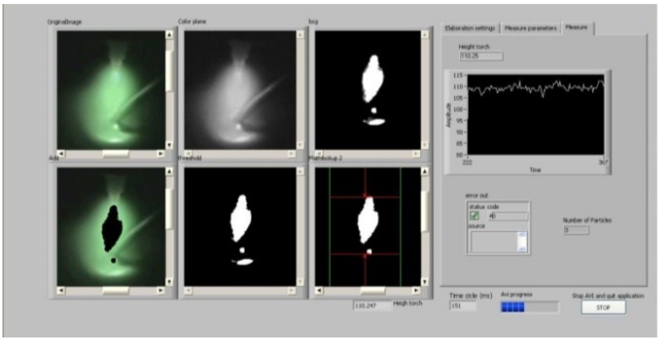


Figure 2.8 An overview of the electric arc

The other system uses a camera involving a particular type of CMOS sensor in which the response of the optical sensor logarithmically related to the light hitting the sensor.



Height torch
110.25

error out
status code ✓ 0
source

Number of Particles
3

Time cicle (ms)
151

Avi progress

Stop AVI and quit application

Figure 2.9 Image processing software

Several image processing algorithms have been implemented, in order to perform the measurement of welding pool dimension, as it is shown in Figure 2.8 [28], [29].

The algorithm implies, in addition to the image calibration, several processing phases, as single colour plane extraction, lookup table, thresholding, particle analysis, erosion, and edge detection. Figure 2.9 shows the software interface adopted, where some of the different processing phases are shown to the user.

Figure 2.10 is a detail of the measures done with image processing.

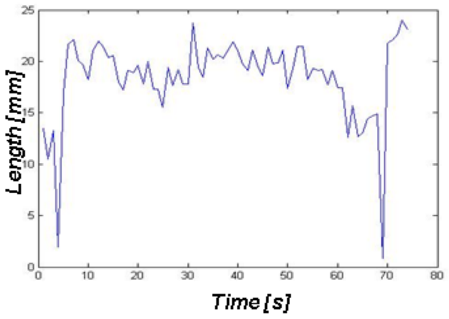


Figure 2.10 Arc length output

In spite of the final work piece shape, the module acquires information about the real amount of material deposited in each layer measuring the torch distance from the bead deposited.

2.4. Arc Voltage

The most used parameter to measure the arc length and consequently to automatically control welding processes, is the arc voltage thus all welding machine usually have an output referred to this parameter. In [30] and [31] different experimental laws, which describe the relationship between arc voltage and arc current or arc length or both, are presented. The most used is:

$$V = K_1 I + K_2 + \frac{K_3}{I} + K_4 L \qquad \textbf{Eq. 2.2}$$

where: V is the welding voltage, I is the welding current L represents the arc length; K are all parameters where K_1 is a constant related to the current while:

$$K_4 = \frac{dV}{dL} \quad \textbf{Eq. 2.3}$$

This means that the last constant changes with arc length.

Those parameters depend on the geometrical configuration of the tungsten electrode, its chemical composition and the shielding gas. See [30] for more details. The arc voltage versus the current and the arc length leads to Figure 2.11.

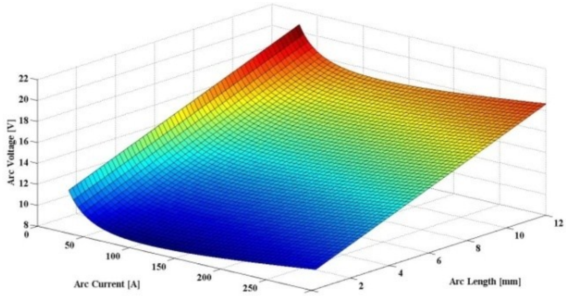


Figure 2.11 Arc Voltage Surface.

It can be seen that the nonlinearity of the voltage-current characteristic of the arc is strong, while the nonlinearity of the voltage-length characteristic is slight.

In [30] is shown how Eq. 2.2 is the combination of two more simple experimental laws which separately deal with arc current and arc length are introduced [32].

The first law deals with voltage-current characteristic:

$$V = \xi_1 + \xi_2 I + \frac{\xi_3}{I} \quad \textbf{Eq. 2.4}$$

This, assuming an arc length of 4mm, leads to the diagram reported in Figure 2.12.

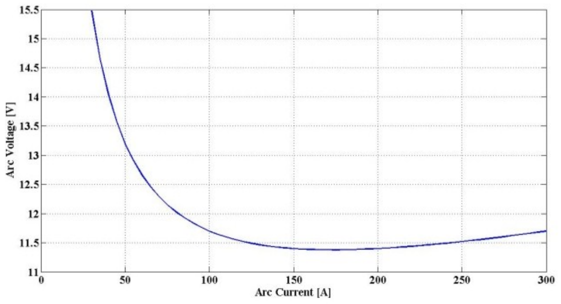


Figure 2.12 Arc Voltage vs. Arc Current [L=4mm]

The second law deals with voltage-length characteristic:

$$V = \eta_1 + \eta_2 L + \eta_3 L^2 \qquad \textbf{Eq. 2.5}$$

This, assuming a current of 100 A, leads to the diagram reported in Figure 2.13.

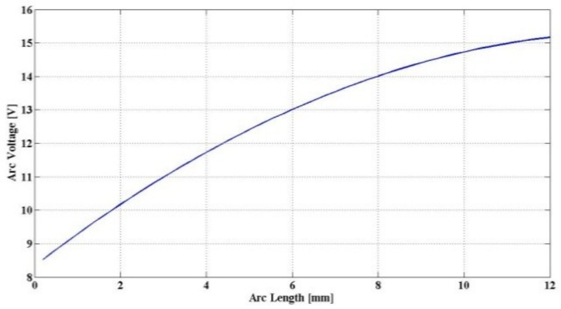


Figure 2.13 Arc Voltage vs. Arc Length [I=100A]

The nonlinearities of the voltage current and voltage length are most severe for low currents and short arc lengths than at high currents and long arc lengths. Therefore, a constant arc voltage does not mean that the arc length is constant if the current changes.

2.5. Sensing the arc length

The block diagram of the whole architecture used to study and develop the control laws for the process, is shown in Figure 2.14

and is mainly based on the NI Labview™ real time target architecture [33]. The choice to use Labview™ was made to quickly develop the investigated algorithms because of, up to now, there is no software with the same performances in term of modularity and expandability.

A workstation is configured as a real time unit, with a low level real time operative system provided by NI, able to drive directly the NI sensors and actuators control boards installed on a PC. As it is shown in Figure 2.14, the NI Labview™ real time target PC represents the core of the control architecture, providing the connection to the other modules, like the motor control (Wire Feeder), the video feedback, the temperature measurement and all the other sensors. The software Interface used will be described in the next paragraph.

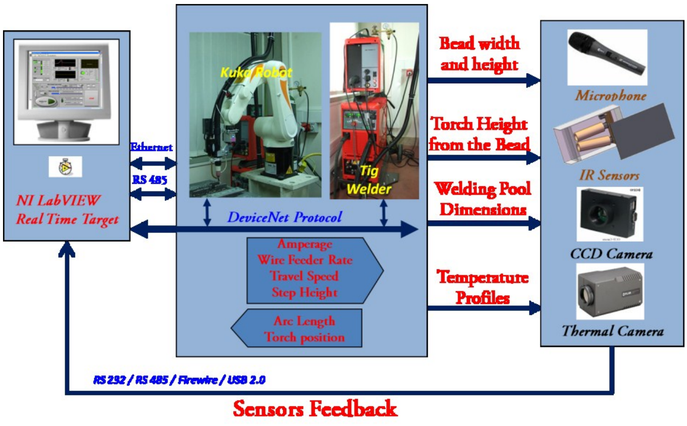


Figure 2.14 Plant Overview

The arc sound, as said before, was not been an useful source of feedback, as resulted, on the contrary, the arc light.

Figure 2.15 reports the light intensity, expressed trough the voltage red on the sensors, compared to the arc voltage. Green and blue dots represent the outputs of the two different sensors. The red line is an interpolation of these data. It implies there is a linear relation among these two sources.

Both the two source of information has a common drawback. In case of an high electric arc the wire may not hit the centre of the melted welding pool, in this situation the wire starts to drip and fall down the welding pool. In this case both voltages show a jitter for each drop formed by the wire.

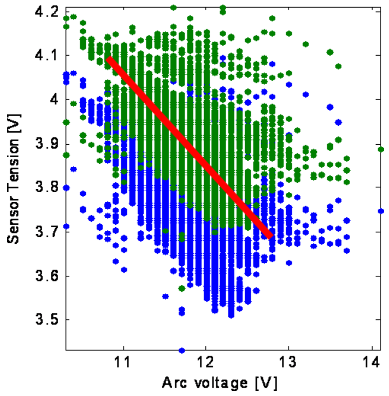


Figure 2.15 Light and Voltage Relation

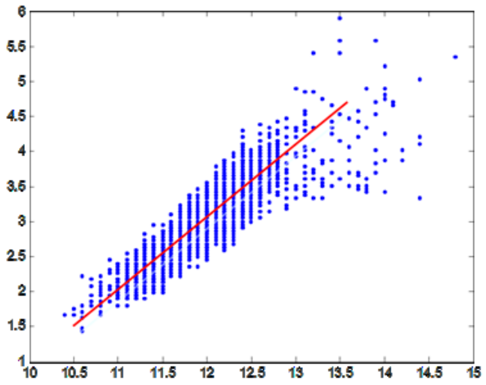


Figure 2.16 Image Processing and Voltage Relation

Image processing is still a work in progress, anyway it has shown good results. In Figure 2.16 the output of the image processing algorithm, expressed in millimetres, is compared to the arc voltage. The red line interpolates again those data implying another linear relation among them.

Image processing algorithm has shown a greater reliability because of it is not sensible to the drawback expressed before.

Despite of all the technique shown in previous paragraphs, the most used parameter to measure the arc length and consequently to automatically control welding processes is the arc voltage, as reported in [30] and [31], since all welding machine usually have an output referred to this.

The welding voltage, rather than the other parameters of the system, will be used then used as the primary source of feedback in order to perform the control action.

2.6. Software Interface

It should be observed that, even if one of the final aims of the RAPOLAC project is to make the SMD process autonomous, in the developing phases the intervention of the operators has been really fundamental. Thus the implemented software user interface (UI) has been very useful, because it has allowed to monitor all the variables and, if needed, to change in real time the process parameters.

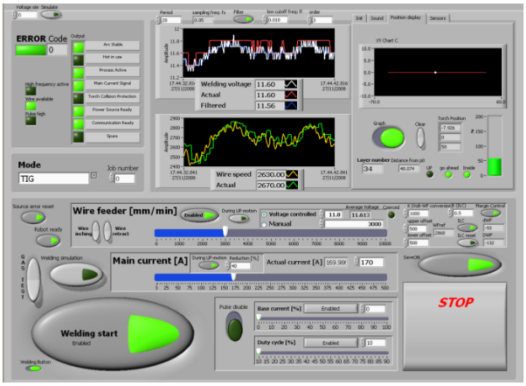
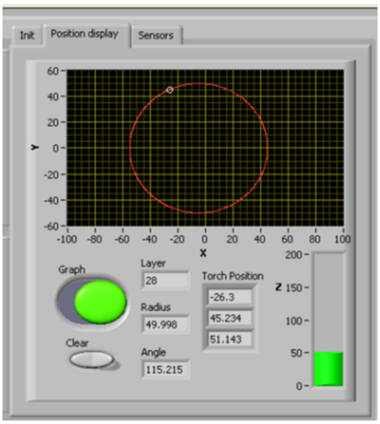


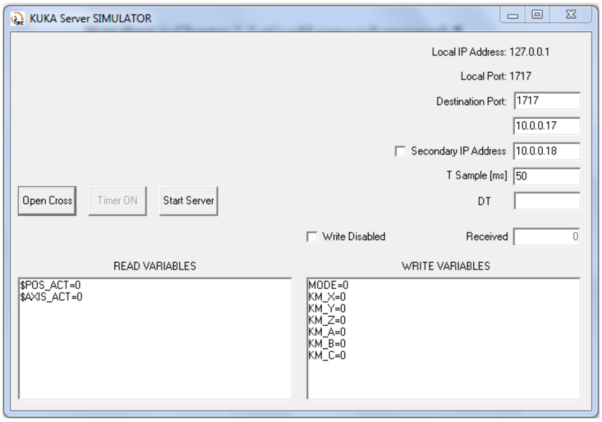
Figure 2.17 The user interface

Figure 2.17 shows the main form of the UI developed with NI Labview™ while Figure 2.18 shows the section of the UI showing the position of the tip of the welding torch. Further details can be found in [34].



Layer: 28
Radius: 49.998
Angle: 115.215
Torch Position: -26.3, 45.234, 51.143

Figure 2.18 Robot position on UI



Local IP Address: 127.0.0.1
Local Port: 1717
Destination Port: 1717
10.0.0.17
☐ Secondary IP Address 10.0.0.18
T Sample [ms] 50
DT
☐ Write Disabled
Received 0

```
$POS_ACT=0
$AXIS_ACT=0
```

```
MODE=0
KM_X=0
KM_Y=0
KM_Z=0
KM_A=0
KM_B=0
KM_C=0
```

Figure 2.19 KUKA Server

Figure 2.19 shows the developed Server Application which runs on the robot controller and, through simple UDP messages, establishes the communication between the robot and the UI.

2.7. Summary

This chapter has shown how estimation of the arc length, in the field of welding system, can be performed. The estimation of the arc length l is physically related to the height of the deposited material h which is a key parameter for the developed control algorithm. Aspects related to the control of the process will be shown on the following chapter.

Chapter 3 Rapolac Control

This chapter will deal with SMD process control aspects. As said previously, the control strategies presented here are the first, as far as we know, dealing with this kind of process.

Figure 1.7 is reported again below, in order to refresh the main features of the process.

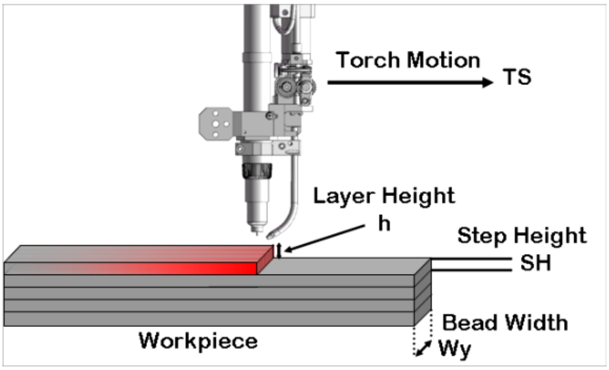


Figure 3.1 SMD Overview

Figure 3.2 shows an overview of the process of deposition. The red arrow on the left represents the incoming material from the wire feeder, which is melted in the welding pool, depicted in red in the center. By means of the motion of the welding torch, at speed *TS*, the melted material solidifies and is left deposited backward the welding pool itself. Figure 3.3 shows real images of these actions.

As described in Chapter 2, the main parameters involved in the SMD process are four. The final work piece production time constraint imposes in some way a value of travel speed *TS,* which usually is maintained constant. The final workspace shape resolution imposes a proper step height *SH* choice. The need of maintaining constant the welding pool width, imposes an uniform energy rate, thus a constant current *I*.

As a result, the wire feeder rate *WF* becomes the control input for the process.

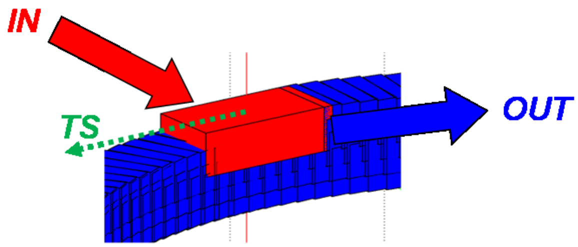


Figure 3.2 Metal deposition overview

Figure 3.3 Deposition screenshots

Figure 3.4 reports a top view section.

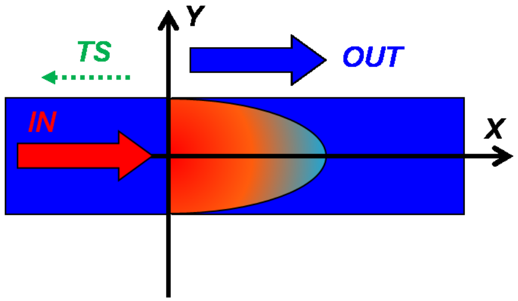


Figure 3.4 Section

The incoming mass rate from the wire feeder can be evaluated with:

$$\dot{m}_i = \rho \dot{V}_W = \rho S_W v_W \qquad \textbf{Eq. 3.1}$$

where ρ is the density of the metal, V_W is the incoming volume of the wire, S_W is the section of the wire, while v_W is the speed of the wire, which is *WF*.

The output deposited mass rate can be expressed by means of:

$$\dot{m}_d = \rho \dot{V}_d = \rho(\dot{\overline{xyz}}) = \rho(\dot{x}yz + x\dot{y}z + xy\dot{z}) \qquad \textbf{Eq. 3.2}$$

The mass deposited can be considered symmetrically deposited, this means that the quantity upon the x axis in Figure 3.4 equals the quantity below.

Eq. 3.2 leads to the following:

$$\frac{1}{2} S_W v_W = \dot{x}yz + x\dot{y}z + xy\dot{z} \qquad \textbf{Eq. 3.3}$$

The system behavior can be described by:

$$\begin{cases} \dot{z} = \dfrac{1}{2}\dfrac{S_W v_W}{xy} - \dfrac{v_x}{x} z - \dfrac{\dot{y}}{y} z \\ \dot{x} = v_x \end{cases} \qquad \textbf{Eq. 3.4}$$

The last derivative of *y* represents the process of growth of the deposited material along that axis. Experimental results have shown that, in normal conditions, the growth of the deposited material always reaches a constant dimension, which is the welding pool width W_Y. As a matter of this and of the fact that just the steady state behavior is considered here, the derivative along *y* axis can be ignored. So the system in Eq. 3.4 can be rewritten as:

$$\begin{cases} \dot{z} = \dfrac{1}{2}\dfrac{S_W v_W}{x W_y} - \dfrac{v_x}{x} z \\ \dot{x} = v_x \end{cases} \qquad \textbf{Eq. 3.5}$$

The equations before represent a second order system in which v_W corresponds to the control input *WF* and *z*, as it will be clearer later, is the output h. It is useful to highlight that both the state variables are observable.

The first equation of the system shown in Eq. 3.5, rewritten below, shows the nature of the deposition process, in which two opposite phenomena take act. In some way, one can be considered constructive, the other destructive.

$$\dot{h} = \frac{1}{2}\frac{S_W}{xW_y}WF - \frac{TS}{x}h \qquad \textbf{Eq. 3.6}$$

The former is related to the first term in which the incoming material increases the height of the deposition. The latter, being v_x/x (TS/x) positive, decreases the height, because the motion of the torch spreads the melted metal among a wider surface. The greater is the TS the lower is the h, being other parameters constant.

Two control loops are implemented in order to let h to converge to the desired *SH*. So the layer height is a key-role parameter of the process and, moreover, is a key-role parameter for the different control strategies investigated.

Before introducing the adopted control laws, it is useful to highlight how several experimental tests have lead to identify some stable working point conditions, in terms of *I*, *TS*, *SH*, *WF*. Those stand for working conditions which keep, even if temporarily, stable deposition processes.

3.1. WF Control

The basic idea for setting up an automatic controller is to measure the amount of material deposited on each layer h (see Figure 3.1). This amount must be equal to the *SH* imposed for the process, in order to avoid depositing a different quantity of the material, thus causing the system to become unstable. According to this information, the wire feeder rate must be controlled, in order to adjust the incoming material mass for each layer.

Being the deposition process modeled as a second order system, the control strategy adopted is, in short, a linear action on the WF on the basis of the error between the height h of the deposited layer and the desired step height *SH*, in order to compensate for an incorrect amount of material injected.

A control law for a similar process has been already shown in [17], where h can be directly measured through image processing. But this parameter is not simple to be directly measured in the SMD process, owing to the particular operative conditions. Moreover another aspect of the process as the torch height from the welding pool bead, gives indirect information on the deposited material. The basic idea is that if the system succeeds in maintaining the same gap during the deposition of a layer after another, it means that the system has deposited the exact amount of material corresponding to the imposed *SH*.

The torch height from the bead is represents the length of the electric arc. On the basis of these considerations, the control law reported in [17] has been modified, using the arc voltage as arc length feedback, in:

$$WF(t) = \overline{WF} + K_L(V(t) - V_{REF}) \qquad \textbf{Eq. 3.7}$$

where $\overline{WF}$ and V_{REF} represent the above mentioned stable working point conditions, K_L is the control loop gain, *V(t)* is the feedback voltage, while *WF(t)* is the control input for the system. For instance, the test reported in [35] has been performed using the below reported values.

Table 3.1 Example of stable working point conditions

Parameter	Value
TS	3.75 mm/s
SH	1 mm
I	150 A
V_{REF}	13.6 V
$\overline{WF}$	1300 mm/min

The wire feed rate $\overline{WF}$ represents the value corresponding to the stable process working point, in which all the material coming from the wire feeder is deposited in each layer, thus ensuring the stability of the process. This value is strictly connected to the value of the other parameters: I, TS and SH. The gain K_L represents the voltage to wire feeder rate conversion factor, tuned on the basis of several experimental tests.

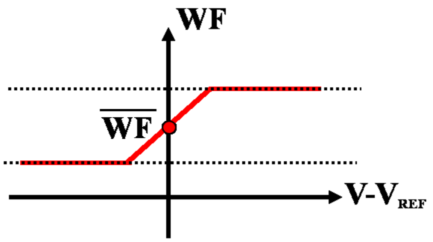


Figure 3.5 The control loop gain

A linear action on the *WF* is evaluated on the basis of the error between the measured arc voltage value and the arc reference voltage. *WF* is corrected in order to compensate for an incorrect amount of material injected. For a null error, the wire feeder rate $\overline{WF}$ is imposed, meaning that the desired step height is respected.

Saturation is performed to assure deposition of uniform layers, as represented in Figure 3.5.

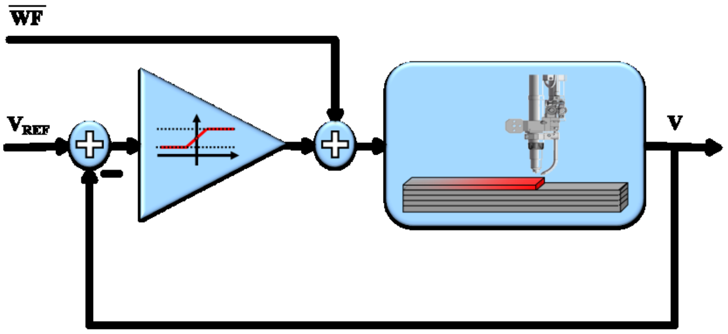


Figure 3.6Control Scheme

Figure 3.6 represents the adopted control strategy.

A generalization of the previous control strategy has been implemented too. In this case precise information on the initial stable working point is not required, and the $\overline{WF}$ value is automatically calculated.

The automatic modulation of the $\overline{WF}$ value is obtained by means of an integral action, taking advantage from the iterativity of the process.

$$\overline{WF}_k = \overline{WF}_{k-1} + K_G \int_{Layer\ k-1} (V(t) - V_{REF})\, dt \qquad \textbf{Eq. 3.8}$$

The value of $\overline{WF}$ is evaluated adding iteratively the weighted mean of the errors measured during the previous layer deposition. The constant K_G is the weighting factor used to ensure the convergence to the right $\overline{WF}$ allowing the steady state *SH* error to be zero.

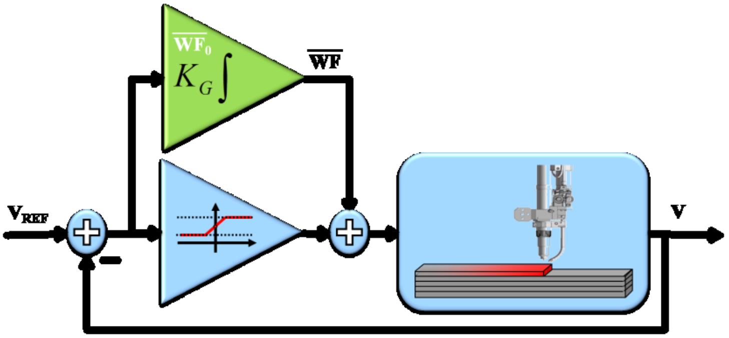


Figure 3.7 Adaptive Control on $\overline{WF}$

As reported in Figure 3.7, the mass flux injected to the system *WF* is calculated by a proportional action working on a local steady state value $\overline{WF}$, that is calculated from an integral action based on the mean error measured during the previous layer deposition. The first action operates at the same sample time adopted to collect all data from sensors, while the second one operates at a lower frequency, because of the value of $\overline{WF}$ is changed once each layer is deposited.

The result of the control law expressed before is reported in Figure 3.8, which shows a stainless steel made cylinder having a diameter of 60 mm.

The reported test has been executed using the initial working point conditions reported in Table 3.1.

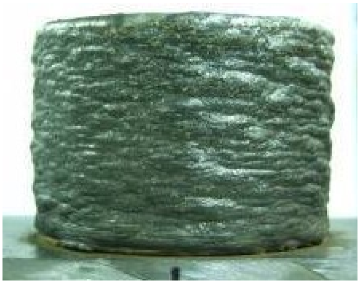


Figure 3.8 A cylinder deposited with stainless steel

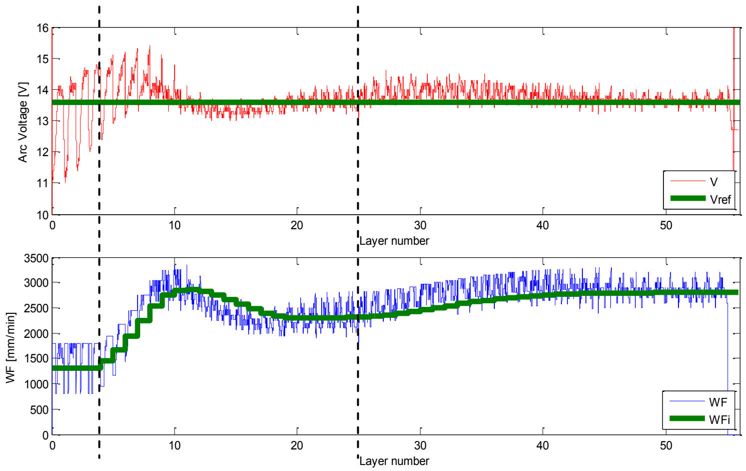


Figure 3.9 Data time series

The initial value of $\overline{WF}$, $\overline{WF}_0$, has been set without a given rule. Moreover during the pre heating phase, used to warm the basement on which the piece has to be built, a deformation of the 5mm basement plate has happened. This is due to its too small thickness, but it has permitted to test the system performance in extreme conditions.

Indeed, as it can be seen from the arc voltage time series in Figure 3.9, the deformation causes oscillations of the arc voltage during the deposition of first layers.

Even though the lower part of the structure is compromised, the control has been able to compensate this disturbance and to locally correct the distortion.

Initially, during the first layers deposition, only the linear action has been used and the structure appears definitely regular up to the10th layer.

When the integral action has been turned on (around at the 4th layer), the control system works in order to find the steady state WF value, that previous experimental tests have set at around 2300 mm/min.

After the 25th layer deposition, the travel speed has been updated to 5 mm/s. Consequently the changed conditions require a calculation of a new working point, with the new value of wire feeder rate of 2800 mm/min, which assures a stable and regular growth of the structure.

3.2. V_{REF} Control

As said in Chapter 2, the arc voltage is related to the arc length so, as many experimental tests have shown, the adoption of a particular value of V_{REF} equals to set a value of arc length. In other words this means to set a distance of the torch from the welding pool bead.

In some cases, especially during the deposition of the first layers, the arc voltage can be very different from the desired V_{REF} this implies, by means of the proportional action, a great value of WF.

Experience has shown that, even if the electric arc is able to melt a great amount of incoming mass, sometime the excessive mass can drop down from the workpiece outside the welding pool.

This causes irregularities in the structure, that have to be recovered, and waste of material. During depositions a V_{REF} value is imposed to the process.

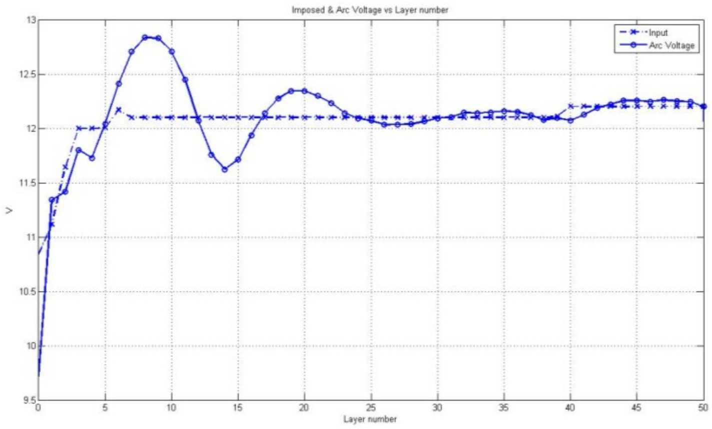


Figure 3.10 Average layer voltage

In Figure 3.10 V_{REF} and the average voltage for each layer are shown. The former, dashed line, is changed somewhere during the test, while the latter, shown with solid line, has been analyzed and modeled as a second order system.

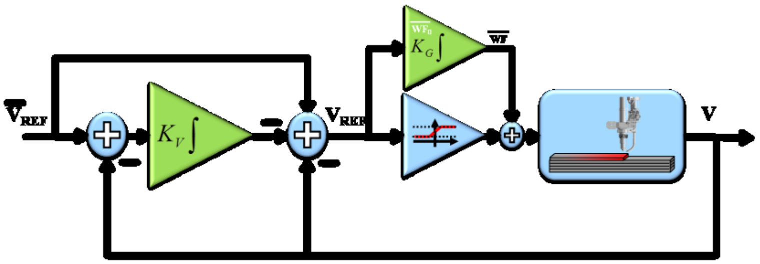


Figure 3.11 Global Overview

For all these reasons another control loop, shown in Figure 3.11, has been implemented and tuned up. The way V_{REF} is evaluated is close to the one reported in Eq. 3.8.

3.3. Conclusions

The above exposed control laws have been deeply tested in really different and severe conditions, in order to check their robustness and to tune the gains involved in. The set of control laws has been also called *SMD Controller*.

Figure 3.12 shows a complex shape, built during several deposition tasks, using stainless steel. In this example a rotating table was used in order to build those parts which looks inclined.

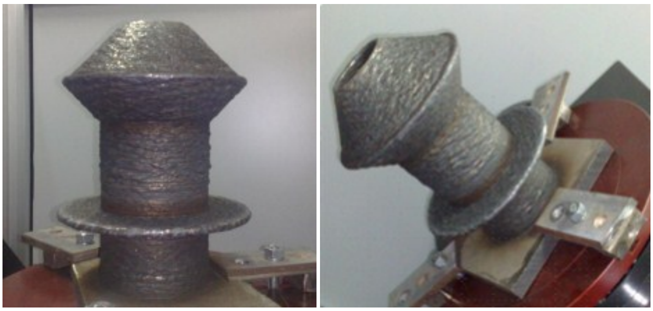


Figure 3.12 Stainless steel 316 structure deposited using 2 axis rotating table

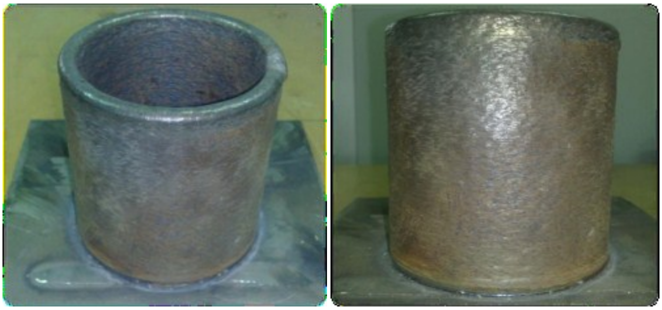


Figure 3.13 Titanium cylinder deposition

Figure 3.13 besides, shows a cylinder built with titanium. It is important to notice that the control algorithm has been able to maintain stability even if the material used is different from all those built with steel.

The control algorithm has been also tested with another SMD plant housed at AMRC laboratories in Sheffield UK [2]. Figure

3.14 reports a comparison of two cylinders built with an operator acting on the system (left) and with the SMD controller (right). The test consists in a deposition of 40 layers Titanium Cylinder in air.

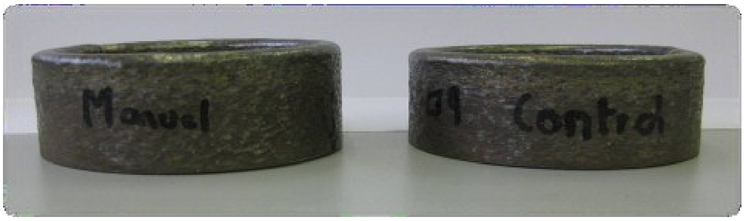


Figure 3.14 Comparison

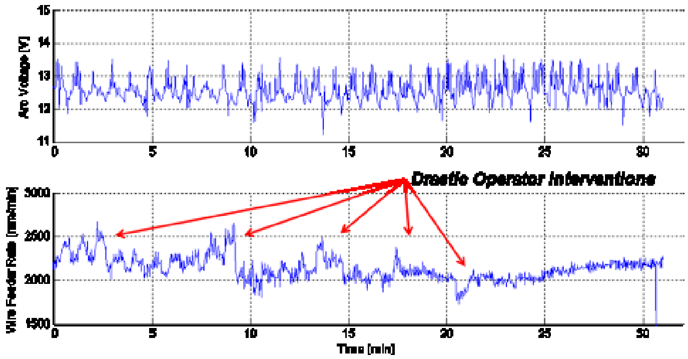


Figure 3.15 Operator deposition: arc voltage and wire feed rate time series

More attention has to be paid to the time series reported in Figure 3.15, where it is highlighted how the operator act somewhere to the *WF* even drastically changing its value. In this kind of deposition the operator works being constantly looking to welding parameters, such as arc voltage, and to images acquired through a special video system.

As can be highlighted from the arc voltage and wire feed rate time series reported in Figure 3.16, while during the manual test the operator have to manually correct the wire feed rate several times, in the other case the controller automatically find the steady state condition, so that the process can be leaved unattended.

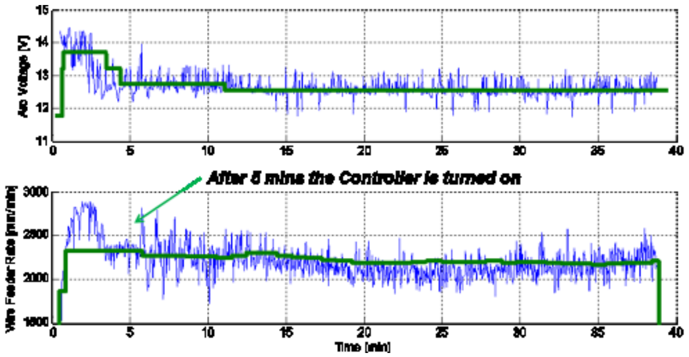


Figure 3.16 Controlled deposition: arc voltage and wire feed rate time series.

Figure 3.16 shows another test performed to produce a 115 layers titanium square in argon, with the initial conditions reported in Table 3.2. Also this test has been performed in the English plant.

Table 3.2 Initial conditions

Parameter	Value
TS	5 mm/s
SH	1 mm
I	115 A
$\bar{V}_{REF}$	12 V
$\overline{WF}_0$	900 mm/min

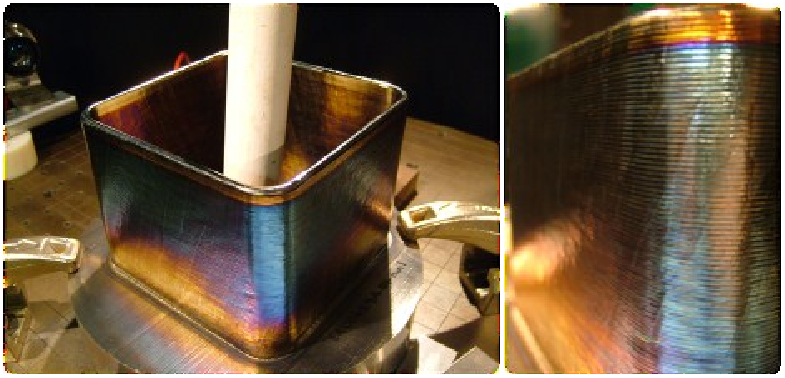


Figure 3.17 A titanium square built in argon atmosphere

As it can be observed from Figure 3.18 and Figure 3.19, even though the initial wire feed rate is totally wrong, the control is capable to find the steady state condition in a few minutes.

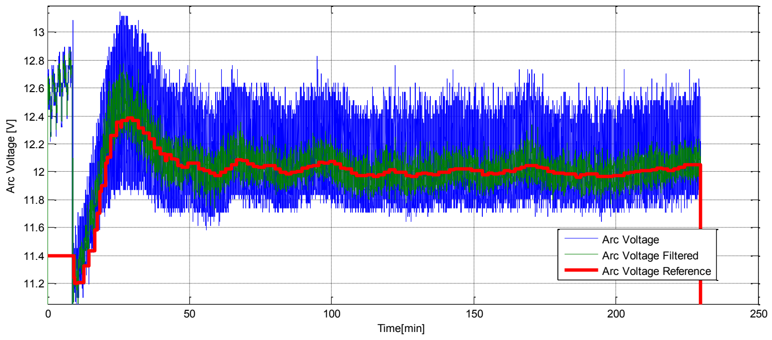


Figure 3.18 Square deposition: arc voltage time series

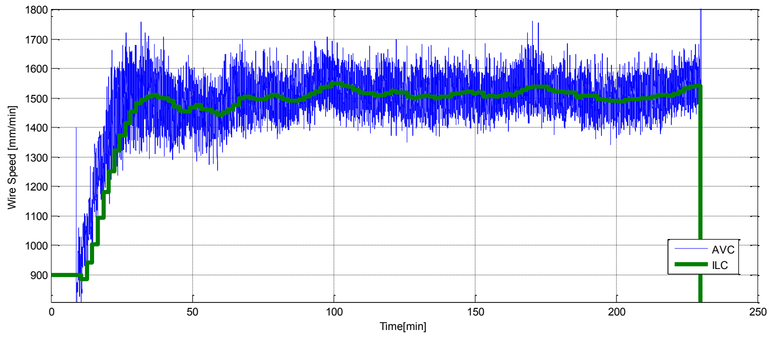


Figure 3.19 Square deposition: WF time series

3.4. Summary

A closed loop welding controller for SMD process has been presented.

The control strategies developed permit to find and to maintain the process stability conditions, so that the process results totally automatic. These steps are performed locally, during the single layer deposition in order to eventually correct the basement or previous layer imperfections.

Integral actions are also performed, taking advantage from the fact that the process is iterative, in order to reach a stable working point.

Chapter 4 Robot Modeling and Control

The dynamic model of a manipulator consists of mathematical equations describing the dynamic behaviour of the manipulator in order to perform simulations, analysis of robotic structures and developing control algorithms.

Another aim of the RAPOLAC project was to investigate on the model of the manipulator that has to move the welding torch, used in the SMAD process. This chapter presents the software framework developed to simulate robotic manipulators and to test control strategies.

4.1. Manipulator Dynamic

A dynamic model is used to study real manipulators and to help designing new ones. The equation that defines the dynamic model of any robotic structure is:

$$\tau = B(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \qquad \textbf{Eq. 4.1}$$

where τ is a vector representing the joint torques, q, $\dot{q}$ and $\ddot{q}$ are vectors representing the joint positions, speeds and accelerations respectively.

Equation above defines the relation between the joint torques and the robot motions. As shown in Figure 4.1 the joint torques vector τ is the input of the robot and the joint position and speed vectors, q and $\dot{q}$, are the outputs.

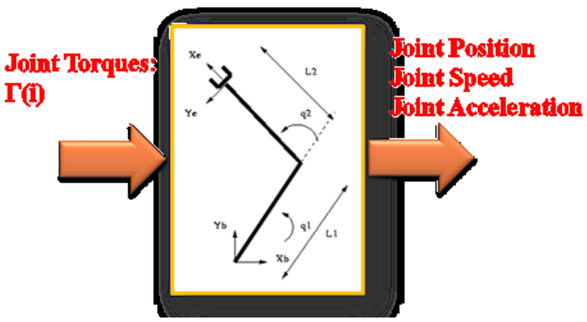


Figure 4.1 Manipulator Dynamic

There are two main ways to evaluate the dynamic model of a robot.

The first formulation is called *Lagrange-Euler Formulation* and has the following properties:

- Energetic formulation
- Dynamic model in closed form (symbolic)
- Useful in the analysis and design of control systems
- Allows easy addition of contributions (e.g., The dynamics of actuators)

The second one is the *Newton-Euler Formulation* that uses a different approach and has the following properties:

- Balances of forces and torques on individual links
- Dynamic model in recursive form (numeric)
- Useful for efficient computation of inverse dynamic

The former approach has been adopted in order to simulate the manipulator behaviour.

4.2. Lagrange-Euler Formulation

The basic strategy behind this method is to define a function, called *Lagrange function*, defined as:

$$L = \Gamma - U \qquad \textbf{Eq. 4.2}$$

where Γ is the total kinetic energy of the structure while U is the total potential energy. For a given dynamical system the following relation holds:

$$\frac{dy}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i \qquad \textbf{Eq. 4.3}$$

where q_i and $\dot{q}_i$ are again the position and the speed of the i-th joint while τ_i is the generalized torque at *i-th* joint. Figure 4.2 shows more details.

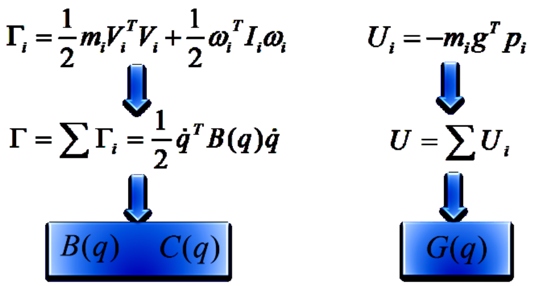
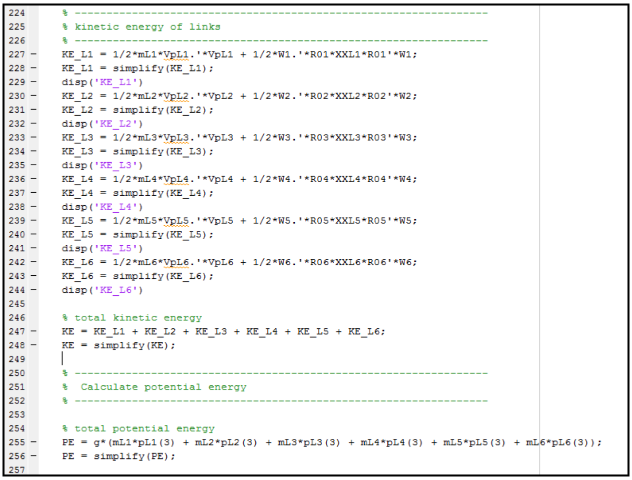

$\Gamma_i = \frac{1}{2} m_i V_i^T V_i + \frac{1}{2} \omega_i^T I_i \omega_i$
$U_i = -m_i g^T p_i$
$\Gamma = \sum \Gamma_i = \frac{1}{2} \dot{q}^T B(q) \dot{q}$
$U = \sum U_i$
$B(q) \quad C(q)$
$G(q)$

Figure 4.2 From Lagrange function to model matrices

In this formulation several mechanical properties are adopted, as the inertia tensor I_i or the mass m_i, of all the robotic links.



```
% ----------------------------------------------------------------
% kinetic energy of links
% ----------------------------------------------------------------
KE_L1 = 1/2*mL1*VpL1.'*VpL1 + 1/2*W1.'*R01*XXL1*R01'*W1;
KE_L1 = simplify(KE_L1);
disp('KE_L1')
KE_L2 = 1/2*mL2*VpL2.'*VpL2 + 1/2*W2.'*R02*XXL2*R02'*W2;
KE_L2 = simplify(KE_L2);
disp('KE_L2')
KE_L3 = 1/2*mL3*VpL3.'*VpL3 + 1/2*W3.'*R03*XXL3*R03'*W3;
KE_L3 = simplify(KE_L3);
disp('KE_L3')
KE_L4 = 1/2*mL4*VpL4.'*VpL4 + 1/2*W4.'*R04*XXL4*R04'*W4;
KE_L4 = simplify(KE_L4);
disp('KE_L4')
KE_L5 = 1/2*mL5*VpL5.'*VpL5 + 1/2*W5.'*R05*XXL5*R05'*W5;
KE_L5 = simplify(KE_L5);
disp('KE_L5')
KE_L6 = 1/2*mL6*VpL6.'*VpL6 + 1/2*W6.'*R06*XXL6*R06'*W6;
KE_L6 = simplify(KE_L6);
disp('KE_L6')

% total kinetic energy
KE = KE_L1 + KE_L2 + KE_L3 + KE_L4 + KE_L5 + KE_L6;
KE = simplify(KE);

% ----------------------------------------------------------------
%  Calculate potential energy
% ----------------------------------------------------------------

% total potential energy
PE = g*(mL1*pL1(3) + mL2*pL2(3) + mL3*pL3(3) + mL4*pL4(3) + mL5*pL5(3) + mL6*pL6(3));
PE = simplify(PE);
```

Figure 4.3 Some Matlab™ Code

This methodology has been implemented using Matlab™ Symbolic Toolbox [36]. Figure 4.3 shows some code defining the kinetic and potential energy. A program has been developed in order to generate the dynamic model of any desired structure, starting from the mechanical specification such as link length, masses, etc…

The output of the program created is a Matlab-based function which allows any user to simulate the robot behaviour and design and refine control algorithms.

4.3. Simulations

First of all the program has been tested with simple robotic structures as a two-link manipulator. To test the goodness of the results the manipulator has been simulated using the Matlab environment and the Samcef Mecano™ [37]. In both cases open-loop simulations where done, with an input, the joint torques, imposed to the system and the output, joints position and speed, is recorded. Simulations have shown the same results.

In order to define a controller for the structure, a canonical approach has been implemented. The approach is known as *Inverse Dynamic Controller* and is shown in Figure 4.4, where $\eta(q,\dot{q}) = C(q,\dot{q})\dot{q} + G(q)$.

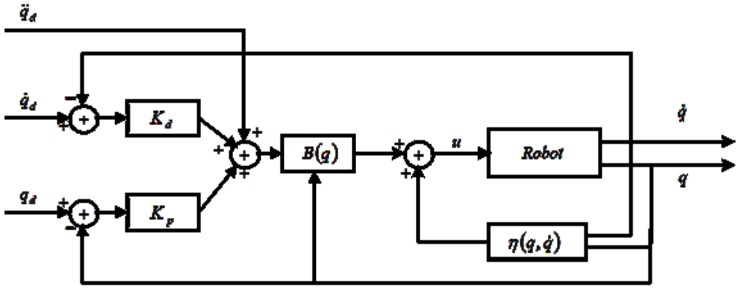


Figure 4.4 Inverse dynamic control

The inverse dynamic approach is based on the knowledge of the dynamic equations of the robot as it uses again the matrices *B*, *C* and *G* of the dynamic model.

This approach has been tested both with some simple structures, like the reported before, with an ideal anthropomorphic manipulator and with a model of the Kuka Kr16 robot used in the project. In detail the performed simulations have to follow these steps:

1. Evaluate the robot dynamic model using Matlab program.
2. Define the control architecture.
3. Tune the control parameters.
4. Perform *Co-Simulation*.

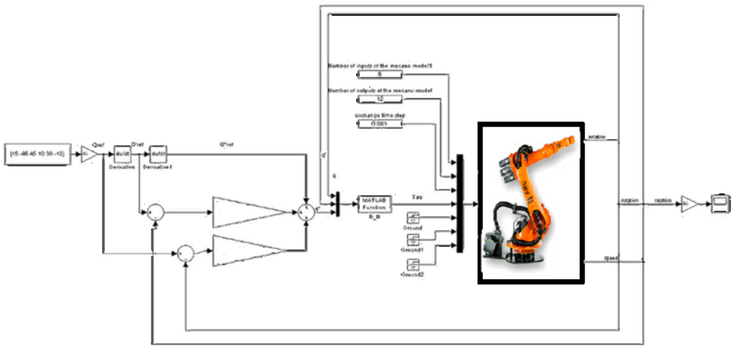


Figure 4.5 Co-Simulation

The controller has been tested and refined in Matlab environment.

Co-Simulation is a term used to indicate *Coordinate Simulations*; Co-Simulations, as shown in Figure 4.5, make use of the capabilities of both the environments, indeed the controller implemented in Matlab has been used to control the robot defined in Mecano environment.

Some troubles have been experienced simulating the behaviour of the Kuka Kr16 robot. First of all, the Matlab program needed a very long execution time to generate the functions of the dynamic model. Moreover the functions generated, which are a Matlab functions, are made of 18Mbytes of code.

This led to an extremely high simulation time. To avoid such a problem, a variant of the same control strategy has been developed in order to reduce the dimension of the function and, consequently, the execution time. All the inertia tensors I_i have been approximated, as reported in Eq. 4.4 below, to a diagonal form, in the evaluation of the model matrices for the controller.

$$I_i = \begin{bmatrix} IL_{XX} & -IL_{XX} & -IL_{XZ} \\ -IL_{XY} & IL_{YY} & -IL_{YZ} \\ -IL_{XZ} & -IL_{YZ} & IL_{ZZ} \end{bmatrix} \cong \begin{bmatrix} IL_{XX} & 0 & 0 \\ 0 & IL_{YY} & 0 \\ 0 & 0 & IL_{ZZ} \end{bmatrix} \quad \textbf{Eq. 4.4}$$

In this case the absence of extra-diagonal terms, led to a smaller code and a faster execution time.

This is an approximation, but it is good enough in case of low speeds, like in the case of the Rapolac project. So the Mecano representation of the robot has been controlled with this simplified controller after some preliminary tuning of the K_P and K_D constants. Figure 4.6 shows the results of the co-simulation, when the controller parameters K_P and K_D are set to *200* and *30* respectively. The robot reaches the desired configuration in less than 1 second and keeps it in a stable way.

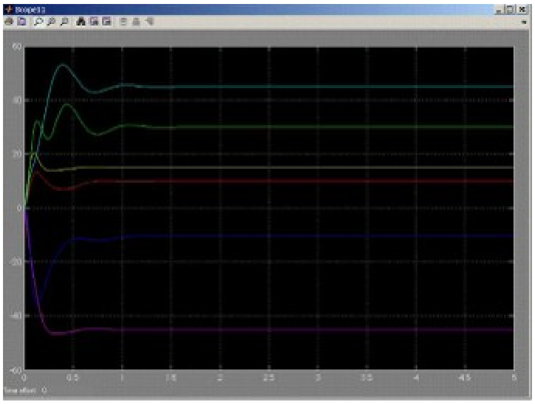
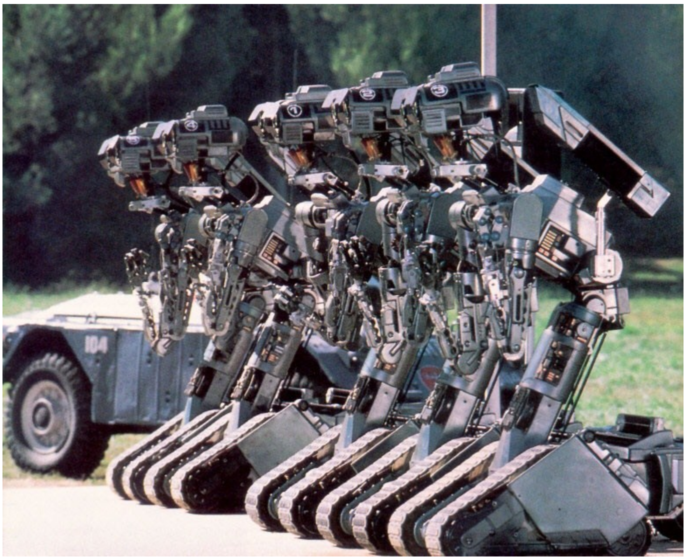


Figure 4.6 Co-Simulation Results

4.4. Summary

This chapter has shown the software framework developed to simulate robotic manipulators by means of their dynamic model. The robot has been simulated using *finite elements* software while control strategies has been developed using the dynamic model developed with Matlab tool.


104

Chapter 5 Robotic Mobile Platforms

This chapter deals with all the robotic mobile platforms involved in my research activity. It shortly describes the hardware and the sensor suites each robot is equipped with.

Sensors will be deeply described in Chapter 7 while Chapter 8 describes how their information is used.

The platforms presented here are four: Robovolc, 3Morduc, U-Go Robot, Segway Rmp200.

5.1. Robovolc

Robovolc was the original name of a research project that was initially funded by the European Commission from 2000 to 2003. The partnerships included two Universities (Università degli Studi di Catania, Italy, and University of Leeds, U.K.), two industrial organizations (Robosoft, France and BAE Systems, U.K.) and two research organizations that provided their expertise in volcanology and cartography: INGV, Italy and the Institute de Physique du Globe de Paris, France. A more detailed description of the project with the latest updates can be found at the project WEB site [38] or in the papers [39] [40] [41] [42] [43] and [44]. The main result of the project was the design, realization, and testing of the *ROBOVOLC* platform, shown in Figure 5.1 and Figure 5.2. Since the conclusion of the project, the robot has been continuously updated and tested and is a powerful tool for the investigation of the adoption of robotics in volcanology.

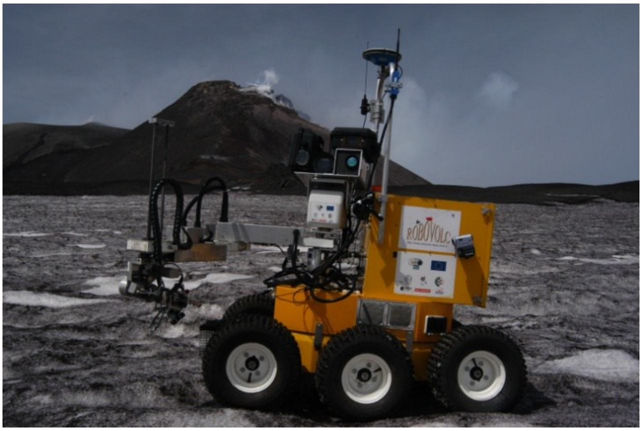


Figure 5.1 Robovolc on a snowy terrain

Robovolc uses a six-wheeled system with an articulated chassis and its dimensions are W x L x H = 80cm x 130cm x 180cm and its total weight is 350kg. The robot is skid-steering since it is able to rotate by exploiting the different speeds of the wheels of right side from those of the left side. The wheels are actuated by using six independent DC motors.

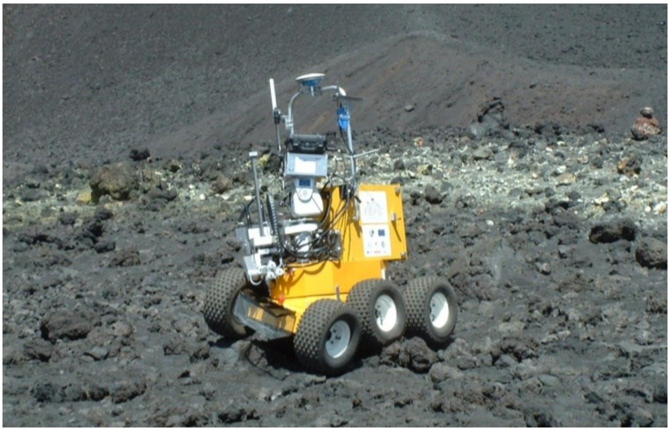


Figure 5.2 Robovolc trial

Several tests performed with different prototypes (see [39] [40] [41] [45] [47] [48] for further details) have shown that an articulated chassis satisfies all the requirements and guarantees an adequate mechanical robustness. The front and rear axles of the robot have two possible movements: a rotation along the longitudinal axis of the robot, that is only passive, and a second rotation along the lateral axes (parallel to the rotation axes of the central wheels), which can be actively controlled by means of two electric motors. Figure 5.2 reveals a typical situation on rocky terrain where the chassis' capability of adapting to the terrain is fundamental.

Power supply is guaranteed by means of 4 sealed lead acid batteries coupled to form two 24V units. The autonomy of the system in typical working conditions has been tested to be 2 hours, while the typical distances that can be covered are in the order of 3km.

The robot in most of situations can be tele-operated by human operators on the base station located at a safe distance from dangerous sites. However, since distances can reach several kilometers, it can be difficult to recognize the terrain by means of video cameras onboard the robot; likewise, in some cases, the radio link signal can be broken. Therefore, the ability of the robot to move autonomously is crucial.

The navigation and localization system of the ROBOVOLC has recently been renewed by adopting an architecture similar to that installed also on the U-GO robot that will be introduced in section 5.3. The on- board sensors suite is composed by:

- Stereo camera Videre STH-MDCS3-VAR Stereo Head [51]
- Attitude and Heading Reference System (AHRS) X-Sens MTi [52]
- Laser range finder (LRF) Sick LMS200 [53][54]
- Ultra Sonic Sonars (USS) Devantech SRF08 [55]
- Global Navigation Satellite System (GNSS) receiver Ashtech Z-Xtreme [56]

5.2. The 3Mo.R.D.U.C.

3Mo.R.D.U.C. is the acronym of "3rd version of the Mobile Robot D.I.E.E.I. University of Catania". It is shown in Figure 5.3.

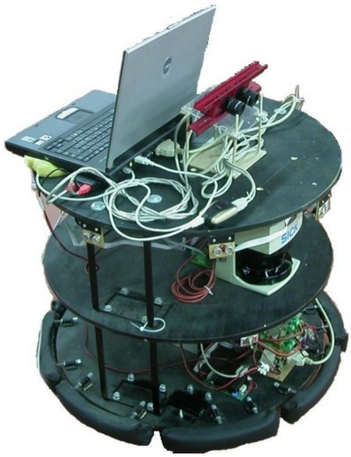


Figure 5.3 The 3Mo.R.D.U.C

The applicative scenarios for the 3Mo.R.D.U.C., shown in Figure 5.4, are tele-operation and tele-manipulation. The communication between the client and the server is based on the HTTP protocol.

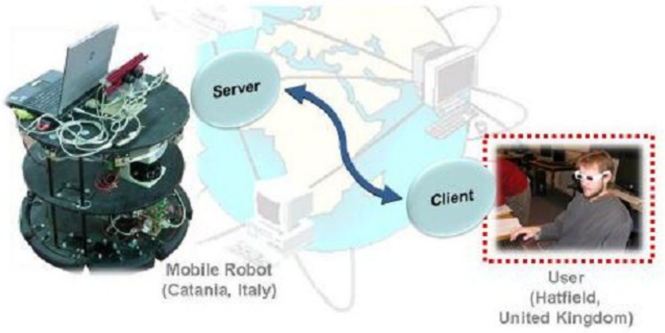


Figure 5.4 Application

In order to carry simple tele-manipulation tasks, the robot has been equipped with a small robotic arm: AL5A by Linxmotion [57].

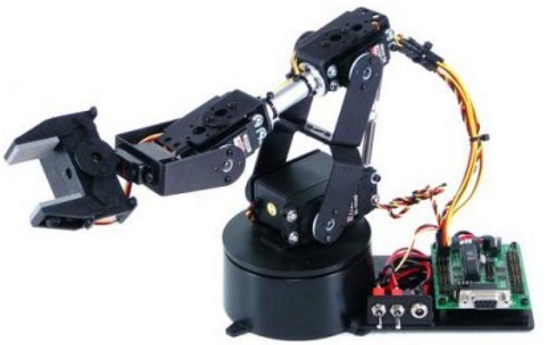


Figure 5.5 The AL5A Manipulator

The 3Mo.R.D.U.C. is a wheeled mobile robot in differential drive configuration. Two 40W DC motors, Maxon F2260 [58] with gear boxes (gear ratio 1:19), are used to move the robot. Two rubber wheels (diameter of 125 mm) are linked with the gearboxes axis and a third castor wheel provides mechanical stability. On the robot there are also two incremental encoders HP HEDL 5540 with a resolution of 500 pulses/turn [59].

The robot structure has three cylindrical shelves linked together. On the lower one, there are two lead batteries (12V/24Ah), which provide the power supply. The robot autonomy is about 60 min. for continuous working. An on board electronic rack controls the modules of the robot (motion, sensors and communication). On the other shelves there are several sensors monitoring the workspace and the robot state (bumper, laser range finder, sonar and stereo camera). On the top shelf there is a laptop where the robot control application is running.

A belt of bumpers, around the entire perimeter of the robot, is mounted on the base, over the wheels (Figure 5.6). These sensors have to recognize and reduce damages in case of collisions. The bumpers are simple switches pushed when there is a collision. The bumpers are connected to an I2C bus shared with the sonar sensors.

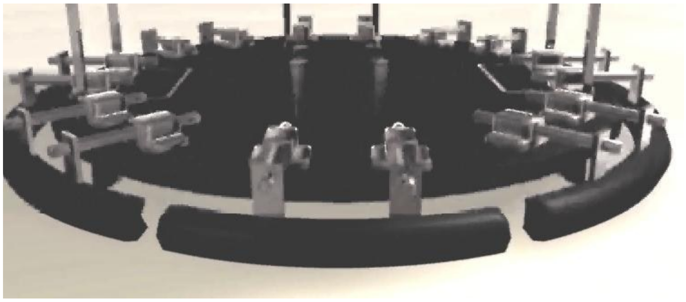


Figure 5.6 Bumpers belt

A circular ring of sonar sensors is mounted below the top shelve. All other on board sensors (laser range finder and stereo camera) and their applications, will be covered in Chapter 7.

The robot has been equipped with a custom made electronic board rack, a modular stacked structure, composed by several electronic boards, which accomplish a specific task through a standard bus (3MO.R.D.U.C.-Bus), guaranteeing the system modularity and providing communication between the boards.

The cards on board are:

a. Supply board: provides the power supply for the laser and for the other boards on the rack.
b. Power board: is composed by two MD03 (H-Bridge); these cards manage the current flow in the motors.
c. ST10 board: controls the motor axes of the robot; the installed firmware allows the communication between the 3Mo.R.D.U.C. and the laptop via RS232 or CAN-bus. The ST10 board implements two PID motor control loops, the PWM output values for the motors are calculated using the speed/position references and the encoders reading. It sends the commands to the power-board through the I2C bus.
d. Encoders board: processes the signals coming from the encoders. In particular, the Encoders board provides a differential to single signals conversion, so that the on-board microcontroller can properly acquire them.

e. PC Interface board: allows the communication between the laptop and the boards on the rack; two protocols I2C and RS-422 were implemented.
f. Multi I\O board: interfaces the bumpers and the sonar sensors. It manages 16 digital input/output and provides an expander I2C connector.

In order to improve modularity and to increase the computational ability of the on board electronic boards shown in points c to f have been changed with a single board based on a *Field Programmable Gate Array* (FPGA). The board used is an Altera DE1 shown in Figure 5.7, [60] [61] [62].

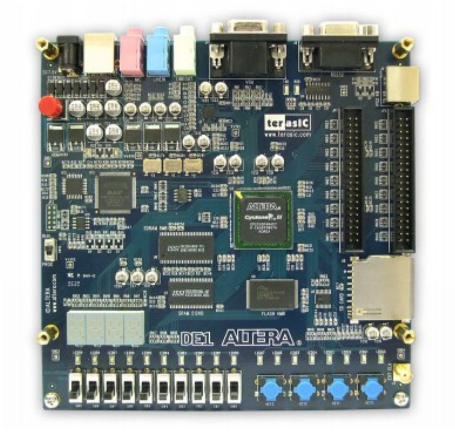


Figure 5.7 Altera DE1 Board

The DE1 board features a state-of-the-art Cyclone® II 2C20 FPGA in a 484-pin package, [63] [64]. All important components on the board are connected to pins of this chip, allowing the user to control all aspects of the board's operation. The DE1 board includes a sufficient number of robust switches, LEDs, 7-segment displays and, for more advanced applications, there are SRAM, SDRAM, Flash memory chips, standard interfaces such as RS-232 and PS/2 and many other I/O devices that will not be described here.

The Altera's Nios II soft processor, [65] [66], has been instantiated and used instead of the old adopted microcontroller.

The Nios II processor has a Reduced Instruction Set Computer (RISC) architecture. Its arithmetic and logic operations are performed on operands in the general purpose registers. The data is moved between the memory and these registers by means of Load and Store instructions.

The word length of the Nios II processor is 32 bits. All registers are 32 bits long. Byte addresses in a 32-bit word can be assigned in either little-endian or big-endian style. The assignment style is one of the options that the user may select during the configuration. The Nios II architecture uses separate instruction and data buses, which is often referred to as the Harvard architecture.

This soft processor has been therefore used as the architecture's base platform connecting, on the hardware side, a central processing unit to various peripherals as PWM-signals generators, encoders, I2C-buses and RS232 handlers. These peripherals are all implemented as hardware modules on board the same FPGA chip.

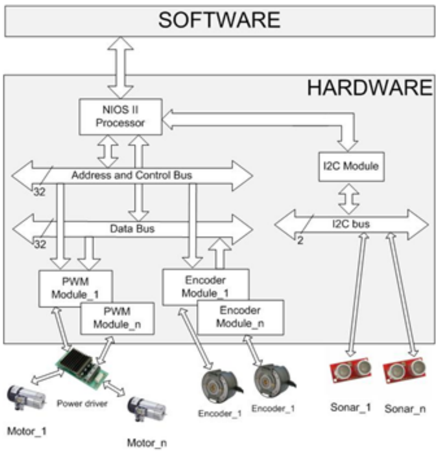


Figure 5.8 FPGA overview

Together with the hardware, a whole C-based software suite, running on the soft processor, has been developed. It provides a high-level user interface to act on and monitor controlled-system's behaviours and, moreover, it allows communication with the higher levels such as the laptop used to control the robot.

To do so, the Nios II soft processor communicates with the outside using the same bus, RS232, and the same protocol of the old electronic on board, allowing users to use the same software.

A software component called *serial handler* manages the dispatching of the incoming messages. This is accomplished by means of two threads working in a *producer-consumer* fashion.

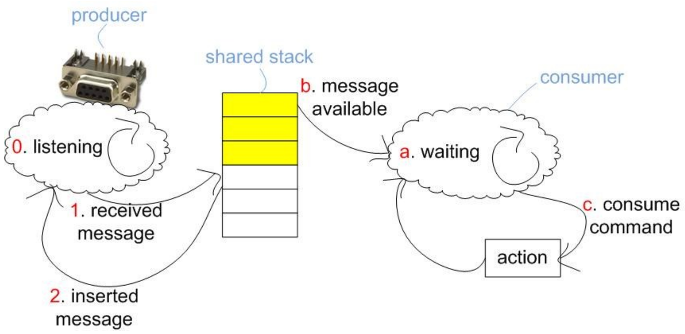


Figure 5.9 Serial handler behavior

The thread listening on the serial port is the producer. It manages data flow through the bus and classifies messages, if valid, as commands to be executed.

Valid messages became resources to be consumed. Those messages are stored in a circular FIFO structure.

Once a message is written into the shared structure, the producer returns to wait for a new command coming.

The thread dispatching commands to the hardware is the consumer. It search for the older message, if any, and sends appropriate commands through the internal buses.

This concurrent management of incoming messages allows, during runtime operations, to compensate for possible delays.

Thread synchronization is performed by means of traffic lights and mutex. All those features are supplied by *μCOSII* kernel which runs on NIOSII processor.

The final result of the project was an easy-to-use, fastly-growable, and, most of all, really versatile architecture.

5.3. U-Go Robot

U-Go Robot is an acronym that means “Unmanned Ground Outdoor Robot”. This robot has been developed at DIEEI Robotic Laboratories mainly to solve problems like transportation, navigation and inspection in very harsh outdoor environments. Moreover, there are different active projects in which the robot is used as a multifunctional vehicle able to mainly operate inside greenhouses for precision farming applications, to perform inspections into volcanic environments and as a test bed for new GNSS localization technologies.

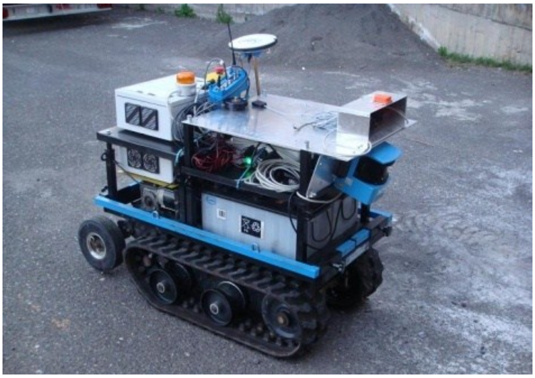


Figure 5.10 The U-Go Robot

The mechanical structure has been designed in order to be compliant to different requirements. First of all, the robot had to be able to move inside greenhouse corridors; moreover it must be able to move on different uneven terrains and must not generate too high pressure on the terrain (in order to meet agricultural requirements). The robot is capable to carry at least 200 kg payload over a flat road (in order to be able to carry agricultural tools like a spraying machine or others) and climb on sloping roads with some payload.

According to these specifications, the robot two main dimensions are 0.6 m width and 1.2 m length. Moreover it uses rubber tracks instead of wheels for locomotion and its weight is about 150 kg. Figure 5.10 shows the U-Go Robot. Two 12 V – 180 Ah sealed lead-acid batteries are mounted on the mechanical structure on the rear side of the robot. Each rubber track is actuated by means of a 24 V – 600W brushed DC motor and suitable gearboxes are mounted in the front side of the robot. Two roller chains connect gearboxes output shaft with tracks input shaft. Above the two DC motors it is possible to see the box that contains the power electronics.

Finally, on the top of the robot there is another box that contains a computer, all the necessary electronic circuits needed for autonomous navigation, the emergency stop button and the safety flash. The computer box can be disconnected if only tele-operation is needed.

The control electronics have been designed to provide different choices for control modalities. The simplest one is the tele-operated modality [67] [68]. In this situation a remote user, using a joystick or simply a computer keyboard, can send simple direct commands to the robot in order to move forward, backward or turn left or right at different speeds. The only components required for this operating mode is the power electronics box and a radio link between the robot and the remote station.

The other two possible modalities for controlling the robot are semi-autonomous mode and autonomous mode. Both these two modalities rely on an onboard computer and on several sensors mounted on the robot.

In the first mode, a remote base station is also required. In the second mode, the remote base station could be avoided, however for safety and logging reasons it is always better to keep it. In semi-autonomous mode, the remote user send high level commands to the robot (like “go forward”); then the robot, using on-board sensors can compensate the whole trajectory for unexpected disturbances and to reach waypoints and targets.

In autonomous mode, the robot will be able to find its way through corridors in the greenhouse, to find optimal path on a not well defined road, to reach a target by GPS waypoints and to avoid obstacles. For safety reasons a remote or local user can always issue an emergency “stop” command to the robot.

U-Go robot shares the same sensors suite of the Robovolc robot. By using a modular HW and SW architecture, the two robotic platforms can be easily exchanged maintaining the same programming environment and hardware components.

5.4. Segway

The Segway RMP200 (Robotic Mobility Platform model 200) is based on the design of the Segway Human Transporter (HT).

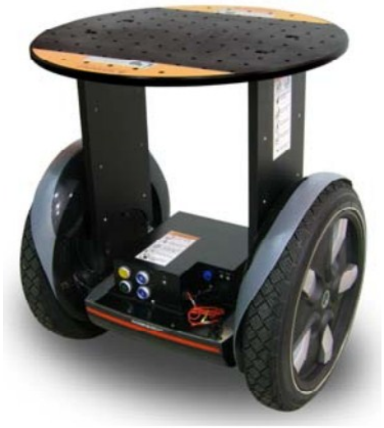


Figure 5.11 The Segway RMP200 Robot

The RMP is a transportation platform designed for integration into a system that has a control processor. Some Characteristics of this platform are reported in Table 5.1. The control, or host processor must create velocity and steering commands and output these in the correct message format to the RMP. The control processor may communicate with the RMP using either CAN bus or USB.

Table 5.1 RMP200 Specifications

Data	Value
Top Speed	16 Km/h
Payload	45 Kg
Turning Radius	0 cm
Maximum Climbing and Descending Capability	10°
Range under Optimal Test conditions	24 Km
Range under Good conditions	19 Km
Range under Severe conditions	13 Km
Run time, stationary	8 hours
Motor Torque Constant	0.071 Nm/A
Motor Drive Peak Current, per wheel	70 A
Motor Drive Continuous Current, per wheel	24 A
Gearbox Ratio	24:1
Battery Pack Capacity	380 Wh
Battery Pack Voltage	72 V
Tire Diameter	48 cm
Wheel Track Width	53 cm

The control processor may be a laptop receiving joystick commands from a stationary PC, a small microcontroller or some other device that can send velocity and steering commands to the RMP via USB or CAN bus.

RMP200 has two operating modes: “Tractor Mode” and “Balance Mode”. The former provides statically stable operation, with a larger footprint and lower payload height. The limits of stability are defined by the distribution of the contact points and the height of the center of gravity.

This mode is a non stabilized, differential steer driver mode. It may be useful when dynamic stabilization capabilities are not needed. An additional ground contact must be provided by the operator (for example a castor wheel on a rigid bar.

The latter, the balance mode, provides dynamically stable operations with a smaller footprint and higher payload height. This has the benefit of allowing for a much higher center of gravity while still maintaining a small footprint. Once Balance Mode is enabled the RMP200 is able to balance itself.

The RMP200 has a control architecture, shown in Figure 5.12, that consists of three distinct processors. Two processors, named CU_A and CU_B, are used to perform the closed loop control on the motors. These processors perform all sensing, control and fault detection functions so that the robot may continue to operate even in the case of a fault. The third processor is a User Interface processor which manages communications to a host processor as well as providing E-stop, watchdog and programming functions for the other two processors.

The main processors in the powerbase communicate with the UI processor via two CAN serial buses while the UI communicates over USB to an host processor.

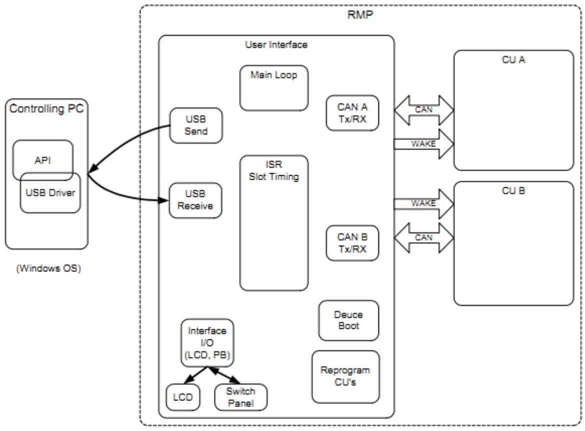


Figure 5.12 Segway RMP200 Control Architecture

Messages that control the movement of the Segway RMP may be sent by the host processor at up to 100 Hz. CU_A processor will slew the velocity command to zero in the absence of new commands. As a result, control messages should be sent by the host processor at a frequency no slower than 2.5 Hz in order to maintain the commanded motion and avoid unintended decelerations. More detail on the RMP could be found in [69] [70] and [71].

The robot has been used to realize a robotic museum guide, as shown in Figure 5.13. Several works covering these topics could be found in literature such as [72], [73], [74], [75].

During this research activity a graphical UI for vocal interaction between man and robot was also developed. The developed software allows the recognition of words spoken by a human operator and the voice synthesis, so to establish a verbal communication between the user and the robot.

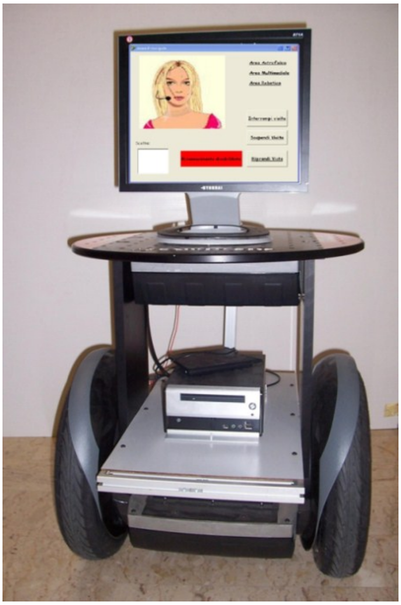


Figure 5.13 The Robot Museum Guide

The iteration with the user is performed by means of two software tools one performing Speech Recognition and the other performing Speech Synthesis.

The global task, the navigation inside the museum interacting with users, is achieved by means of a state machine reported in Figure 5.14.

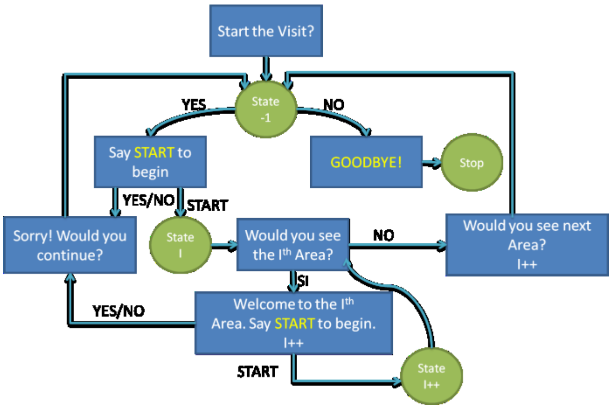

Start the Visit?
State -1
YES
NO
Say START to begin
GOODBYE!
Stop
YES/NO
START
Sorry! Would you continue?
State I
Would you see the Ith Area?
NO
Would you see next Area? I++
SI
Welcome to the Ith Area. Say START to begin. I++
YES/NO
START
State I++

Figure 5.14 The State Machine for the Museum

The Robot Museum is also equipped with a LRF to perform obstacle avoidance. In the kind of application the robot is involved in, the robot has to carefully avoid collisions with human being.

5.5. Conclusions

The four robotic mobile platform presented in this chapter have few aspects in common each other. Two use differential drive configuration: 3Mo.R.D.U.C. and Segway RMP200. Two use skid-steering configuration: Robovolc and U-Go. The latter, moreover, uses track instead of wheels.

All these robots usually share the same sensor suite: GNSS, AHRS, LRF, Stereo camera, etc...

More details about these sensors will be given in Chapter 7 while in Chapter 8 a global overview on the Navigation and Localization algorithms will be presented. Details concerning how simulations, involving these robots, have been performed will be presented too.

5.6. Summary

This chapter exposed the robotic mobile platform involved in the research activity: Robovolc, 3Morduc, U-Go Robot, Segway Rmp200. It has shortly described both the hardware and the particular sensor suite each robot is equipped with.

Chapter 6 MRDS

Most people were really surprised to hear that Microsoft has a robotics team and that they are actively working on software for robots. It was in late 2003 that robotics started attracting Microsoft's managers.

Several universities have at least one robotics project while there are a growing number of enterprises which are focusing on robotic applications. There are, for instance, robots in house [76], robots for education [77] [78] and many robots in industrial plants.

Microsoft sees some parallelism between the state of the robotics industry now and the state of the personal computer industry 25 years ago. Tandy Trower, the general manager of Microsoft Robotics Group, said:

> *Anyone who has been around for a while recognizes that the current robotics industry has many similarities to the early PC industry. Both are characterized by both the tremendous passion and anticipation of the early pioneers and questions about the value of the technology. With the advent of word processing, spreadsheets, and thousands of other applications, no one asks me any longer why I own a PC. In fact, most of them also own PCs. Similarly, the personal robotics market that is just emerging has the same or even greater potential if the creativity of its community can be unlocked.*

Anyway there was and still there is a big business opportunity. So Microsoft decided to found the "*Microsoft robotics team*" drawing developers from several areas of the expertise.

Version 1.0 of the Microsoft Robotics Studio SDK was released in December of 2006, and version 1.5 followed in May of 2007, with a refresh in December 2007. The name of the SDK has been changed in *Microsoft Developer Studio* with the Standard Edition released in November of 2008. Release two (R2) and three (R3) followed in June of 2009 and May of 2010 respectively. Last release MRDS R4 has been released in September 2011.

Microsoft Robotics Developer Studio (MRDS) provides a software platform and a development environment which enables software written for one robot to be also suitable for another similar robot. MRDS aims to help the robotics industry to move forward, by helping companies to invest more in robot behavior and algorithms than has been possible in the past.

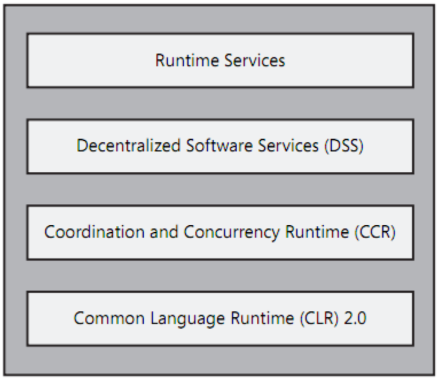

Runtime Services
Decentralized Software Services (DSS)
Coordination and Concurrency Runtime (CCR)
Common Language Runtime (CLR) 2.0

Figure 6.1 MRDS Runtime

Applications built with MRDS run over the web or a local intranet as a collection of state-bound services that are isolated from each other. This allows developers to build web-based and flexible robotics applications. Services will be discussed later.

MRDS consists of a number of components shown in Figure 6.1. It makes use of three lower-level runtimes: Concurrency and Coordination Runtime (CCR), Decentralized Software Services (DSS), and the .NET Common Language Runtime (CLR) 2.0.

CCR and DSS comprise the run-time environment. They are both managed libraries, so the robotics services that operate within their environments are also implemented using managed code.

CLR supports both base runtimes and provides access to the .NET Framework.

The CCR supplies the underlying infrastructure that enables multiple tasks to execute concurrently on a single computer. DSS adds another layer for combining CCR applications, called *Services*, and at the same time it enables these services to run on completely separate computers and communicate via the network.

Services are the basic building blocks for robotics applications in MRDS.

6.1. Coordinator and Concurrency Runtime

The CCR is a message-oriented model that allows applications to coordinate asynchronous processes and exploit concurrency in a very efficient way.

More in detail, the CCR is a managed library that provides classes and methods to help with concurrency, coordination, and failure handling. CCR allows writing segments of code that operate asynchronously and independently.

Communications is performed through messages as shown in Figure 6.2 Messaging. When a message is received, it is placed in a queue, called a port, until it can be processed by the receiver.

Each code segment can run concurrently and asynchronously, and there is often no need to synchronize them because of the message queues. When it is necessary to wait until two or more operations have completed, the CCR library provides the necessary constructs.

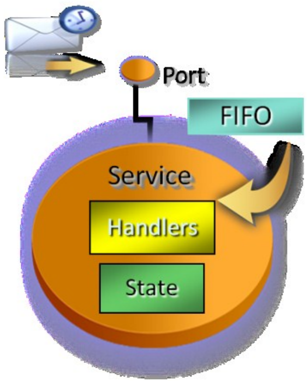


Figure 6.2 Messaging

Moreover it is possible to use exception handling to isolate failures within a single code segment. Errors are reported back to the segment of code that initiated the operation.

6.2. Decentralized Software Services

DSS is a lightweight, service-oriented runtime that combines the principles used to power the World Wide Web with the architecture used to design Web services.

The CCR enables segments of code to pass messages and run in parallel within a single process. The DSS library extends this concept across processes and even across machines. An application built with DSS consists of multiple independent services running in parallel. Each service has a state associated with it and certain types of messages that it receives called operations. When a service receives a message, it may change its state and then send additional messages and notifications to other services.

6.3. Runtime Services

As said before, services are the basic building blocks for robotics applications in MRDS. Services manage state and handle sending and receiving messages. Services handle subscriptions and communication with other services and also handle output and embed other resources such as a reference to an external dynamic link library (DLL) file.

Moreover MRDS defines a set of generic contracts that describe commonly used robotics services such as motors, sonar sensors, and even webcams.

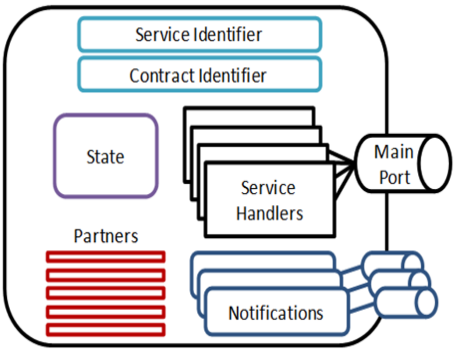


Figure 6.3 Structure of a Service

Figure 6.3 shows services consist of several components.

The *Contract Identifier* defines the messages that can be send to a service, as well as a globally unique reference which identifies the service and is expressed in the form of a *Universal Resource Identifier* which is the *Service Identifier*.

The *Internal state* stores information that the service maintains to control its own operation.

Handlers are a set of operations that the service can perform and specifies the service behavior. This set is defined when the service is created. A simple operation could be returning the state when it is requested. Operations are defined by the ports that represent them, and ports use particular classes (or data types) for their messages. Handlers respond to specific requests (or messages) that income through the *Main Port*.

Notifications are asynchronous source of information that are not related to a specific request. For example a notification can occur when an user send a command to the robot, this is a completely asynchronous event. Notifications, see Figure 6.4, are sent through a different output port.

Partnerships allow services to be combined, as partners, with other services. A service is thus able to use partner's capabilities to create more structured applications.

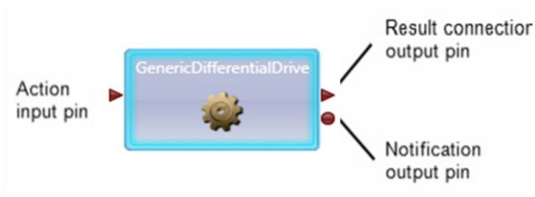


Figure 6.4 Picture of a Service

Services can be created and destroyed dynamically, they are usually specified at startup through a manifest. This is an XML file that lists the required services and their partner relationships.

The state of a service can be retrieved programmatically by sending a *Get* message to the service or it can be retrieved and displayed using a web browser. Services may subscribe to be notified when the state of a service changes or when other events occur.

6.4. Messages

The basic function of every service is to send and receive messages. This is what enables services to communicate with each other. Messages are strongly typed, and message types are specified within a service. DSS requires that a message must contain the following three elements: *Action*, *Body* and *Response Port*.

The *Action* informs the runtime about what operation should be performed. For example, *Get* indicates that the service should get the latest version of the state, and *Update* indicates that the service should update the latest version of the state.

The *Body* contains the request represented trough XML code.

The *Response Port* is the port to which the response should be sent. If specified, a response should be sent regardless of whether the action results in a success or failure.

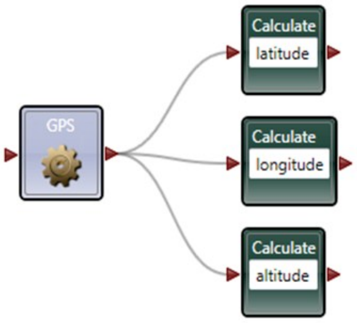


Figure 6.5 Connecting blocks

Figure 6.5 shows an example of how connections between services are realized. The service on the left is providing simple GPS information. It responds to a *Get* request and outputs on its main port the state which is, in this case, used to extract information such as latitude, longitude and altitude.

The complexity should be increased at will.

6.5. Visual Simulation Environment

The Visual Simulation Environment (VSE), included with MRDS, is a 3D simulator with full physics simulation that can be used to prototype new algorithms or robots. VSE offers a way to develop robot software without any robotics hardware. Powered by the AGEIA PhysX engine, VSE renders advanced physics graphics. The simulation engine itself is just another service. It uses a native physics engine wrapper and library that are built on top of the AGEIA PhysX engine as shown in Figure 6.6.

VSE is great to save time or money to invest in expensive robots. Time is saved because there is no need to wait for the robot to be available in order to perform some test. Money is saved because programmers can test their code before they risk damaging expensive equipments.

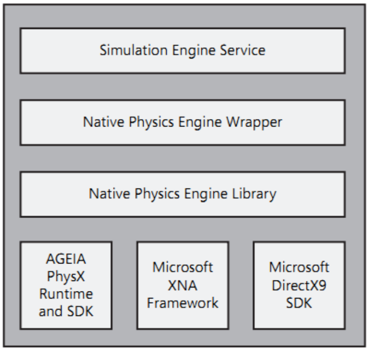

Simulation Engine Service
Native Physics Engine Wrapper
Native Physics Engine Library
AGEIA PhysX Runtime and SDK
Microsoft XNA Framework
Microsoft DirectX9 SDK

Figure 6.6 The Simulation Engine

Clearly, the negative aspect of using a simulation tool is that it is not entirely realistic. During in-field activity robots encounter unforeseen obstacles and can react in unexpected ways; moreover sensors readings are affected by noise. For these reasons, simulation is useful at the beginning of a project, but only using a real robot will yield “real world" results.

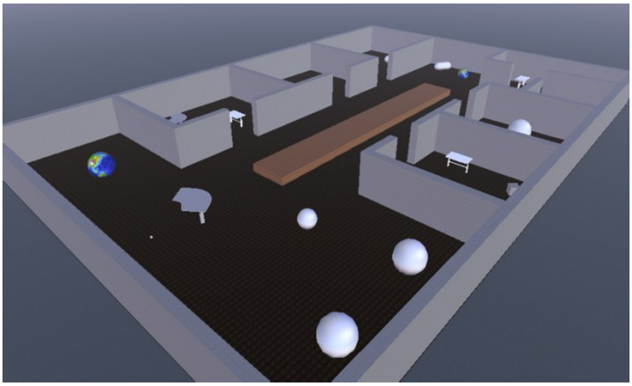


Figure 6.7 Simulated Environment

6.6. Visual Programming Language

The Visual Programming Language (VPL) is a graphical programming environment that can be used to implement robotics services. Figure 6.5, for instance, is a section of a grater application as shown in Figure 6.8.

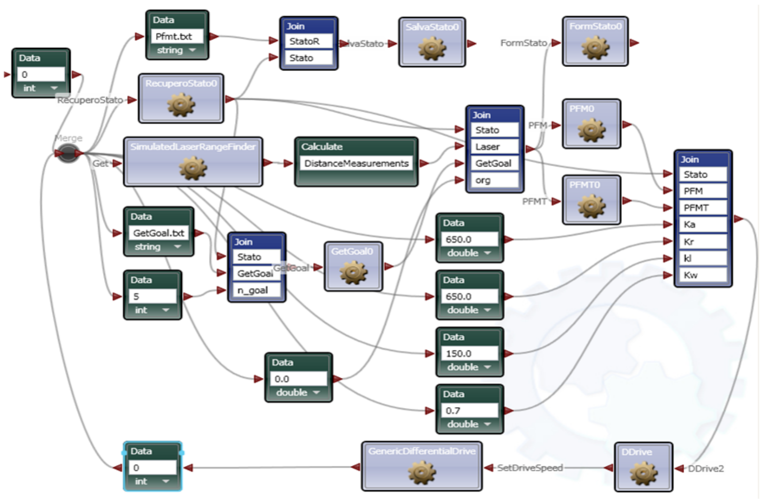


Figure 6.8 An example of VPL

Robotic applications are built connecting blocks that implements services. A block, or service, could be involved in reading a particular sensor while another could generate a command, suitable to act on the robot; this command, or message, is sent to a service which can be either a simulated robot in a simulated environment or a software interface to a real robot. This kind of application, built with VPL, is usually called *diagram*.

6.7. Conclusions

Further details about MRDS could be found in [79] and [80]. MRDS has encountered during past years opposite assessments. It has been subjected to some criticism such as being not completely open-source or being too much founded in the .NET framework.

Several alternatives cannot be forgotten and have been considered before starting this research activity.

Robot Operating System (ROS) is a software framework for robot software development, [81]. ROS was originally developed under the name switchyard by the Stanford Artificial Intelligence Laboratory. From 2008 development continues primarily at Willow Garage, but many other institutions are collaborating.

The *Player Project* is a project to create free software for research into robotics and sensor system, [82]. Its components include the *Player* network server and Stage and *Gazebo* robot platform simulators. Player is one of the most popular open-source robot interfaces.

The Mobile Robot Programming Toolkit (MRPT) is a cross-platform and open source C++ library aimed to help robotics researchers to design and implement algorithms related to Simultaneous Localization and Mapping (SLAM), computer vision, motion planning and obstacle avoidance, [83].

Services are really fundamental to build robotic application with MRDS. From a programmer point of view they are blocks of code that can be programmed with one of the language supported by the .NET framework. In our work the C# programming language has been chosen.

A key aspect of MRDS is that it forces the user/programmer to think robotic application as blocks linked together. This means that even complex applications must be developed with a great modularity. A great modularity allows to quickly change some parts such as a sensor or an algorithm.

Some navigation algorithms, as that reported in the diagram shown in Figure 6.8, have been deeply tested using the simulated environment, shown in Figure 6.7. This environment represents a museum and all these algorithms have been developed for the robot museum presented in section 5.4.

Following some further tuning, these algorithms have been tested with the real robot before making some simple changes. Indeed, in the diagram reported in Figure 6.8, simulated LRF and robot services have been changed with services interfacing with the real hardware.

6.8. Summary

This chapter has presented the Microsoft Robotics Developer Studio environment covering its structure and its fundamentals. The Visual Simulation Environment has been also presented.

Chapter 7 Sensors

This chapter will introduce the sensor suite used. The adopted sensors, such as GPS receivers or AHRS, are widely used in robotic applications. This chapter will also shortly present how they are introduced in the MRDS environment.

7.1. GPS

In Figure 7.1 the adopted Ashtech Z-Xtreme [56] GPS receiver is shown.

Figure 7.1 AshTech Z-Xtreme

Real-time differential positioning involves the adoption of a reference station receiver, called *base-station*, computing the satellite range corrections; the computed corrections are transmitted in real time to the remote receiver through a radio link. The remote receiver applies the corrections to the measured ranges, using the corrected ranges to compute the exact position.

Real-time Kinematic (RTK) positioning uses the carrier signal in addition to the code signal and is much more accurate. Although transmitted messages and performed calculations are different, RTK is essentially a special form of differential positioning.

A base station receiver is required to transmit RTK data to remote receiver. The remote receiver uses the RTK data to compute a corrected position.

As stand-alone, the receiver can compute a position with a precision of about 10 meters. Differential GPS achieves sub-meter precision at the remote receiver, and RTK positioning achieves centimeter accuracy at the remote receiver, as reported in Table 7.1.

Table 7.1 AshTech Z-Xtreme Specification

Positioning Mode	ID	Typical Horizontal Accuracy	Maximum Update Rate
Autonomous	1	100 m	5/10 Hz
RTCM code differential	4	1.0 m	5/10 Hz
RTCM-RTK	5	1.6 cm	5/10 Hz

An MRDS Service has been created in order to access AHRS data from the VPL. As shown in the code reported below, the service creates a thread which has to acquire data and save GPS data in the state of the service.

```
GPSSP = new SerialPort();
[…]
Thread t = new Thread(new
              ThreadStart(Sampler));
t.Start();
```

While the *State* contains following code.

```
[DataContract]
public class FBGPSState
{
    [DataMember]
    [Description("GGA String as received")]
    public string GGAString;
    [DataMember]
    [Description("UTC Time [hhmmss]")]
    public string UtcTime;
    [DataMember]
    [Description("Latitude [deg]")]
    public double Latitude;
    [DataMember]
    [Description("Latitude: N=Nord or S=Sud")]
    public string LatitudeNS;
    [DataMember]
    [Description("Longitude [deg]")]
    public double Longitude;
    [DataMember]
    [Description("Longitude: E=Est or W=West]")]
    public string LongitudeEW;
    [DataMember]
    [Description("Fix Quality")]
    public int FixQuality;
    [DataMember]
    [Description("# of satellites tracked")]
    public int NumSats;
    [DataMember]
    [Description("HDOP")]
    public double HDOP;
    [DataMember]
    [Description("Altitude [m]")]
    public double Altitude;
    [DataMember]
    [Description("Time [S] since last DGPS update")]
   public double LastDGPS;
    [DataMember]
    [Description("DGPS station ID number")]
    public string StationID;
    […]
   public NEHTriple NEH;
    [DataMember]
    [Description("TimeStamp")]
   public double Timestamp;
    [DataMember]
    [Description("Sigma Mesure")]
    public double sigma;
 }
```

One of the utility provided by the developed service is to extrapolate from the incoming GPS strings, which use a standard called NMEA0183, the geodetic coordinate system representing Latitude, Longitude and Altitude (LLA). Moreover the service also provides the East, North and Height (ENH) representation.

This representation allows to have ENH positions expressed in meters from a fixed reference point. See [84] for further details.

7.2. MTi

The XSens MTi, shown in Figure 7.3, is a complete miniature inertial measurement unit (IMU) with integrated 3D magnetometers accelerometers and gyroscope. By means of an embedded processor is able, after a calibration stage, of calculating roll, pitch and yaw in real time [52].

Figure 7.2 Mti

Calibrated sensor readings (accelerations, angular speeds and earth magnetic field) are in the right handed Cartesian co-ordinate system, body-fixed to the device, defined as the sensor reference system. Figure 7.3 shows the reference system.

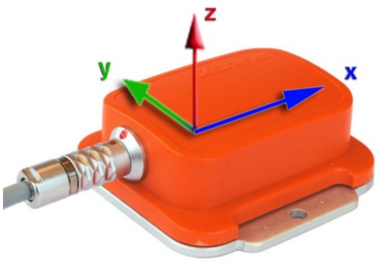


Figure 7.3 MTi Reference system

The MTi is a miniature, gyro-enhanced Attitude and Heading Reference System (AHRS). Its internal processor, shown in Figure 7.4, provides also drift-free 3D orientation The MTi is an excellent and wide used measurement unit for robotic applications.

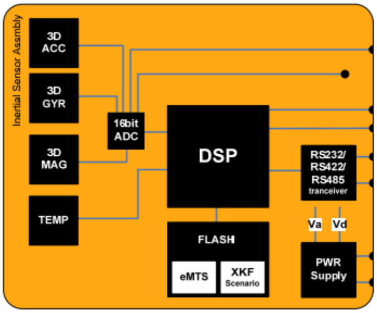


Figure 7.4 MTi System overview

The orientation of the MTi is computed by the *Xsens Kalman Filter* algorithm, called XKF-3, by means of signals from gyroscopes, accelerometers and magnetometers. The output is a statistical optimal 3D orientation estimation, with high accuracy and, moreover, with no drift for both static and dynamic movements.

The XKF-3 algorithm is clearly a sensor fusion algorithm working with the measurement of accelerometers and magnetometers. These are compensated for the slowly, but unlimited, increasing errors, due to the drift, from the integration of gyroscope data. Table 7.2 reports some specifications of the MTi.

Table 7.2 MTi Specifications

Data	Value
Angular Resolution	0.05°
Repeatability	0.2°
Static Accuracy (roll/pitch)	0.5°
Static Accuracy (heading)	1°
Dynamic Accuracy	2° RMS
Max Update Rate	120 Hz

XSens provides a full SDK that allows to access AHRS data with several programming languages.

Also in this case an MRDS Service has been created in order to access AHRS data from the VPL.

```
Thread MTiThread;
[…]
MT.cmtGotoMeasurement();
MTiThread =
    new Thread(new ThreadStart(sampler));
MTiThread.Start();
[…]
```

As shown in the code reported before, the service creates a thread which has to acquire data and save Roll Pitch and Yaw data (RPY) in the state of the service reported below.

```
[DataContract]
[Description("Service State Type")]
public class StateType
{
 [DataMember]
 [Description("Roll Angle")]
 public double Roll;
 [DataMember]
 [Description("Pitch Angle")]
 public double Pitch;
 [DataMember]
 [Description("Yaw Angle")]
 public double Yaw;
 [DataMember]
 [Description("TimeStamp")]
 public double TimeStamp;
}
```

Once RPY data are stored in the state of the service, they are accessible. The use of a thread allows to have each time the most recent data.

Figure 7.5 shows an example developed with VPL. It is a very simple application, where the sensor is continuously asked for new data by means of a timer.

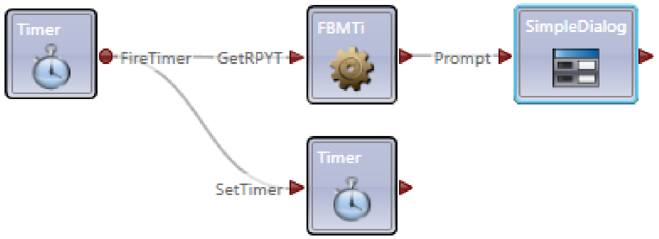


Figure 7.5 MTi VPL Example

7.3. Laser Range Finder

The LMS200 system, shown in Figure 7.6, operates by measuring the flight time of a pulsed laser light beam that is reflected by obstacles. An internal rotating mirror deflects the transmitted pulsed laser beam, so that a scan is made of the surrounding area [53] [54].

The shape of the target object is determined from the sequence of the received impulses. The measurement data is available in real time for further evaluation via RS232/RS422 serial interface.

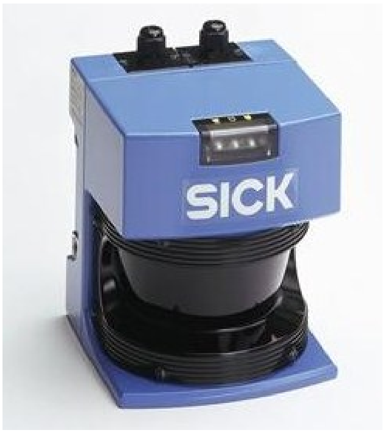


Figure 7.6 Sick LMS200

The time between transmission and reception of the light pulse is directly proportional to the distance between the scanner and the object (time of flight).

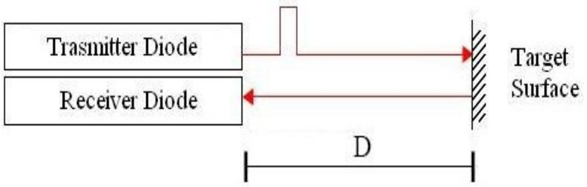


Figure 7.7 Time of fligth measurement

Being *D* the distance between the laser and the obstacle, *c* the light speed and *T* the time of flight, as shown in Figure 7.7, the distance of the objet from the LRF is equal to:

$$D = c \cdot T$$

This device provides also automatic fog correction, for outdoor use. Raindrops and snowflakes are cut out using pixel-oriented evaluation. It is possible to configure three separate angular resolutions (0.25°/0.5°/1°) and the maximum scan angle (100°/180°); each scan is in clockwise mode. The max distance resolution is 10 mm and the max distance range is 80 m, enough for indoor use. The minimum acquisition time, with angular resolution of 1° and scan angle of 180°, is 13.32 ms (maximum sample frequency of 75 Hz). This frequency can be obtained using RS422@500Kbaud. For our purposes, static or quasi-static environments, a sample frequency of 5 Hz is enough, so the scanner laser communicates via RS232@32Kbaud.

Figure 7.8 LRF MRDS Service

Being this device really well known, both MRDS and other developers [85] provide a service suitable to access its data. Figure 7.8 shows the MRDS service, while Figure 7.9 shows the output of an LRF scan; each scan is composed by a set of points.

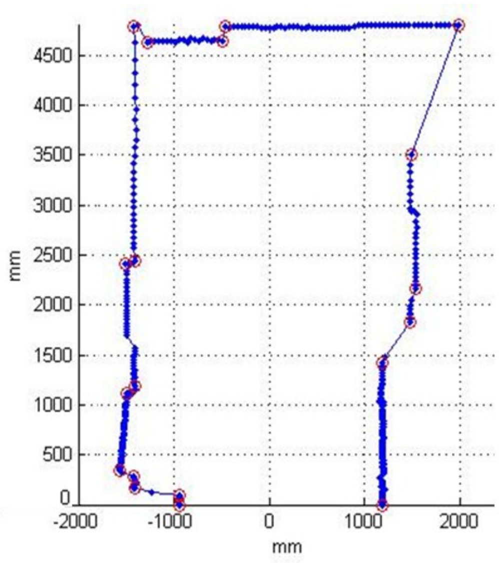


Figure 7.9 LRF Output

7.4. Sonar Sensors

The sonar sensors measure the distance from an obstacle using the flight time of an ultrasound signal produced by means of a vibrating piezoelectric transducer.

The sonar is a device composed by two parts, transmitter and receiver. The transmitter creates an ultrasound pulse, often called “ping", then the receiver listens for reflections, the “echo” of the pulse. The ultrasonic pulse is electronically generated using a Sonar Projector composed of a signal generator, a power amplifier and an electro-acoustic transducer/array.

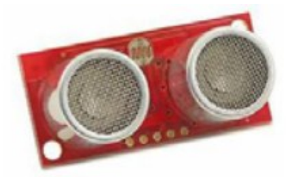


Figure 7.10 Sonar

The flight time is measured and then converted into a range of distances, knowing the sound speed.

The sonar field of view is a cone and the sensitive area increases proportionately with the distance. It is necessary to introduce an inhibition time to avoid the false obstacles due to the ping signal. The inhibition time does not allow reading too short distances. The SRF08 [55], shown in Figure 7.10, is interfaced using the I2C bus.

This sonar ranger, from Devantech, offers a wide range from 3cm to 6m. The beam pattern of the SRF08 is conical, with the width of the beam being a function of the surface area of the transducers and is fixed. The beam pattern of the transducers used on the SRF08 is mostly concentrated in the range [-30° +30°] and is shown in Figure 7.11.

The SRF08 integrates also a photocell sensor on-board. The reading increases as the brightness increases, so a maximum value in bright light and minimum value in darkness can be read. It should get close to 2-3 in complete darkness and up to about 248 in bright light.

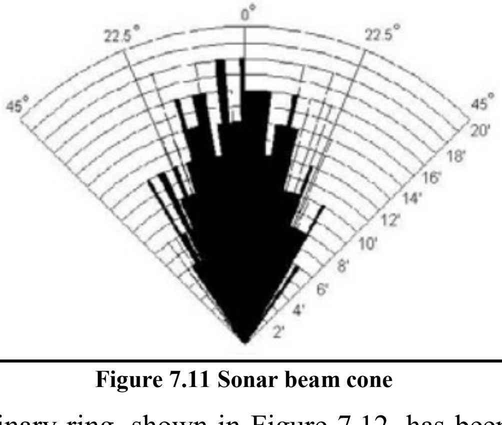


Figure 7.11 Sonar beam cone

A preliminary ring, shown in Figure 7.12, has been created in order to test the proposed algorithm, without using the real set of sensors on board the U-Go robot presented in section 5.3.

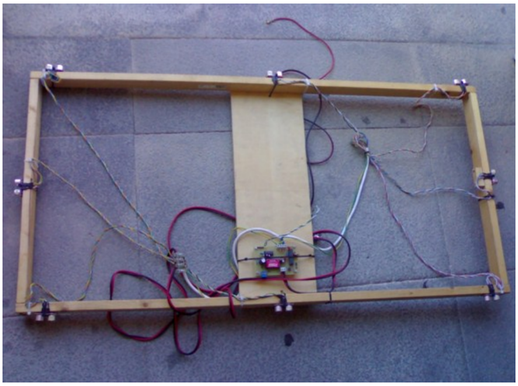


Figure 7.12 US Ring

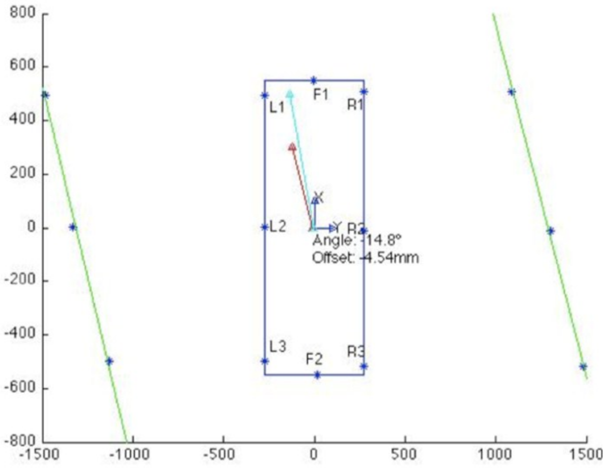


Figure 7.13 Representation of US Ring

The robot is provided with eight ultrasonic sonar sensors. Two are placed on the front and on the back side of the robot and are generally used for obstacle avoidance, while the other six are placed three for each lateral side of the robot. The latter, as it will be clearer later, are used to make the robot navigate, keeping the centre of a corridor.

Figure 7.13 shows an output of the US Ring processed with Matlab™.

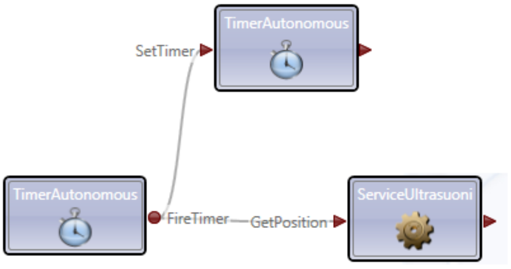


Figure 7.14 US Ring Service

Even in this case a MRDS service has been created, as shown in the diagram in Figure 7.14, which allows to access to all US sensors data and to other information that will be presented later in section 8.4.

7.5. Stereocam

All the robots can be also equipped with two high quality stereoscopic cameras, having each one a resolution of 1.3 Megapixel, each one also equipped with fixed focus lens of 4.0 mm.

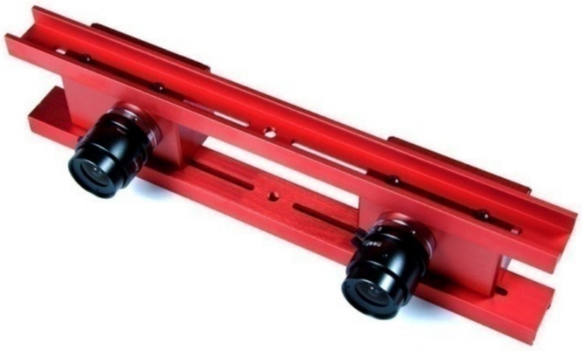


Figure 7.15 Videre Stereocam

The CCD sensors of these cameras have a good noise immunity and sensibility; moreover, it is possible to adjust all the image parameters, e.g. exposure gain, frame rate, resolution. The cameras are mounted on a rigid support, as shown in Figure 7.15, which permits to adjust in a simple way the camera distance in a range 5-20 cm. The images coming from the two cameras are synchronized with an 8 KHz clock, generated by using an IEEE1394 interface [51].

This camera has been used to develop an algorithm suitable to detect the terrain morphology that will be presented in Chapter 8.

The terrain morphology is used both to detect the presence of obstacles in the mission path and to recognize and define a drivable surface in the proximity of the robot.

Image processing is based on the Intel OpenCV open source library [86] that provides useful, quick and reliable basic functions to build up computer vision algorithms. An auto-calibration tool for stereo camera has been also developed, in order to easily and quickly compute cameras intrinsic, extrinsic and distortion parameters.

The algorithm is developed in C++ language and transmits its output to MRDS through simple UPD messages.

7.6. Conclusions

This chapter introduced the sensors used for the presented research activity. As said before, during the simulations of all the proposed algorithms, sensors have to be substituted with the virtual ones. Being all the simulated robots derived from the generic differential drive, positions and heading are obtained querying the Simulation Engine, as shown by the code reported below.

```
[ServiceHandler
    (ServiceHandlerBehavior.Concurrent)]
public IEnumerator<ITask>
    GetStateHandler(RecuperoStato op)
{

 VisualEntity robot = new VisualEntity();
 robot.State.Name = "MotorBaseWithDrive";
// Query to the Simulation Engine

yield return
Arbiter.Choice(
    SimulationEngine.GlobalInstancePort.
    .Query(robot),
    delegate(QuerySimulationEntityResponseType
    response)
    {
     //Robot Position
        double X =
     response.Entity.State.Pose.Position.Z;
        double Y =
     response.Entity.State.Pose.Position.X;
        double Z =
     response.Entity.State.Pose.Position.Y;
     //Quaternion for the Attitude and Heading
           double w =
    response.Entity.State.Pose.Orientation.W;
           double x =
    response.Entity.State.Pose.Orientation.X;
           double y =
    response.Entity.State.Pose.Orientation.Y;
           double z =
    response.Entity.State.Pose.Orientation.Z;

    //Conversion to obtain Heading

           double or =
    (Math.Atan2(2 * y * w - 2 * x * z, 1 - 2 *
    Math.Pow(y, 2) - 2 * Math.Pow(z, 2))) + Math.PI;

    op.ResponsePort.Post(
           new GetStateResponse(X, Y, Z, or));
    },

    […]
    );

   yield return
    Arbiter.Receive(false, TimeoutPort(10000),
   delegate { });
}
```

This code is a part of a MRDS Service that allows to access data from the simulation environment. Position and heading so obtained are used to simulate GPS and AHRS sensors.

Simulated LRF and Sonar are both available in VPL.

Figure 7.16 shows a screenshot of a simulation involving the 3Mo.R.D.U.C. presented in section 5.2.

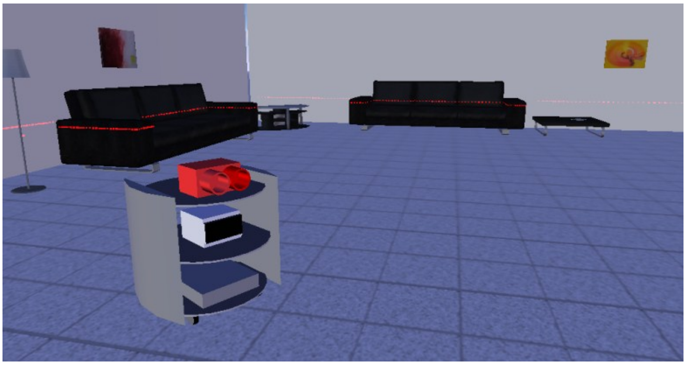


Figure 7.16 Simulated 3Mo.R.D.U.C.

7.7. Summary

This chapter has introduced the sensor suite used and how it is used within the MRDS environment.

Next chapter will presents how robots are implemented and simulated in MRDS and, moreover, all the algorithms developed.

Chapter 8 Multi Sensors Data Fusion

This chapter will firstly introduce how robots are simulated and used in real applications using MRDS services. After that, the robot state estimation stages are presented. The next sections will present the main results of all the research activity this thesis deals with, that are algorithms suitable for self-localization, navigation and obstacle avoidance tasks.

8.1. Robot Services

As said before, during the simulation of all the proposed algorithms, sensors have to be substituted with the virtual ones while all the simulated robots have been derived from the generic differential drive.

MRDS provides a service called *Differential Drive Entity* which has been used to create services suitable to simulate the real robots. This has been done for the U-Go robot and for the 3Mo.R.D.U.C.

```
namespace 3Morduc
{
 [DataContract]
 public class Robot3Morduc:
                  DifferentialDriveEntity
 {
  public Robot3Morduc(Vector3 StartPosition)
  {
    #region Physical Properties
    MASS = 50f;  // mass
```

```
        CHASSIS_DIMENSIONS =// width,height,length
              new Vector3(0.70f, 0.66f, 0.70f);
        FRONT_WHEEL_MASS = 0.5f;// wheel mass
        CHASSIS_CLEARANCE = 0.20f;//wheel base
        FRONT_WHEEL_RADIUS = 0.06f;//wheel radius
        FRONT_WHEEL_WIDTH = 0.02f;// wheel width
        CASTER_WHEEL_RADIUS = 0.06f;
        CASTER_WHEEL_WIDTH = 0.025f;
        FRONT_AXLE_DEPTH_OFFSET = -0.15f;
        #endregion

        base.State.Name = "Robot Morduc";
        base.State.MassDensity.Mass = MASS;
        base.State.Pose.Position = initialPos;

        […]

        State.Assets.Mesh = "morduc.obj";
        // 3D Model from google SketchUP Pro
      }
    }
  }
```

As can be seen from the code reported before, the robot model has been derived from the `DifferentialDriveEntity` where all physical properties, such as mass and dimension, can be specified in order to fit with the real robot.

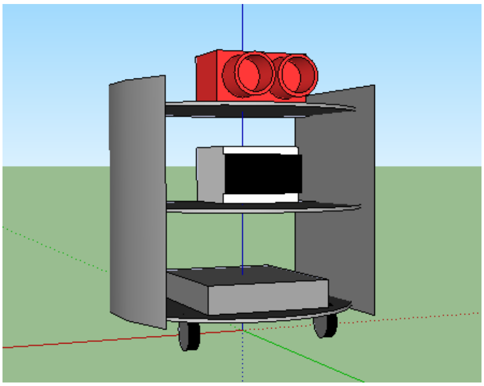


Figure 8.1 3D model for the 3Mo.R.D.U.C.

The three-dimensional model, reported in Figure 8.1, has been created using Google SketchUp Pro [86], a simple and intuitive vector graphics software, and exported as .obj filetype.

In the service developed for the robot, some part of code adds to the simulator all other sensors, as shown in Figure 8.2. For example, the portion of code for the LRF is reported below.

```
private LaserRangeFinderEntity insertLaser()
{
 Vector3 positon = new Vector3(0,0.45f,0);
 LaserRangeFinderEntity laser =
    new LaserRangeFinderEntity(
           new Pose(position));
 […]
 return laser;
}
```

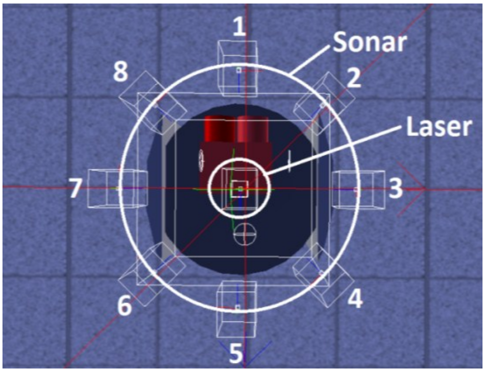


Figure 8.2 Sensors Overview

Although the U-go is not a differential drive robot, the service for its simulation was realized in the same way of the one reported before. Moreover, at an higher level, both the simulation services and the services for real robots have been realized in order to be controlled with the same two inputs, which are the linear and the angular velocity.

As it is well known for differential drive robots the speed of the two wheels, v_L for the left one and v_R for the right one, can be evaluated according to the following equations:

$$V_L = v - \frac{b}{2}w \quad V_R = v + \frac{b}{2}w \qquad \textbf{Eq. 8.1}$$

where is b the wheelbase, while v and w are the linear and the angular speed respectively.

In case of simulations, the simulated environment itself is an MRDS service, so even this one has to be tuned in order to define all the object such as the ground, the sky and all other objects the users want. However this aspect will not be furthermore covered here.

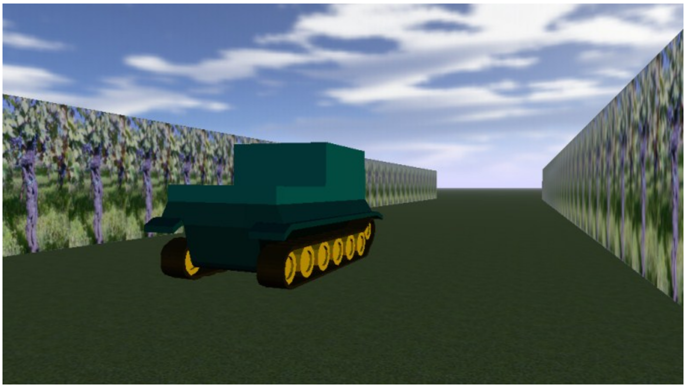


Figure 8.3 Simulation of U-Go inside a vineyard

Figure 8.3 shows a simulation of the U-Go robot inside a vineyard while Figure 8.4 shows another one involving the 3Mo.R.D.U.C.

In case of simulations, VPL diagrams use services for simulated entities, such as the robot and the sensors, while, for in-field experimentation, these services are substituted with those able to communicate with real robots and sensors.

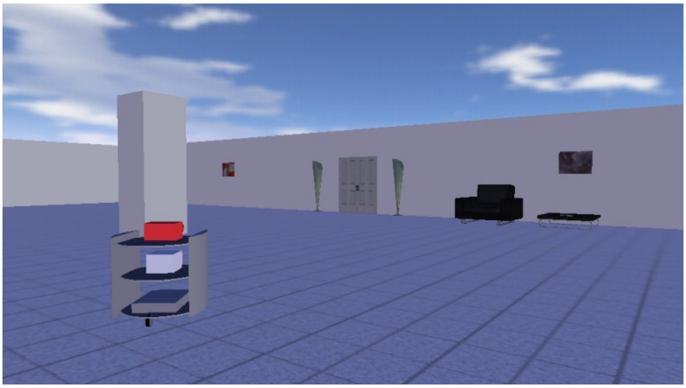


Figure 8.4 Simulation of the 3Mo.R.D.U.C.

8.2. Robot State Estimation

Before developing robust navigation or localization algorithms, a robust framework able to estimate the state of the robot was developed, as it is shown in Figure 8.5. In order to estimate the state, an Extended Kalman Filter (EKF) has been used [88], [89],[90].

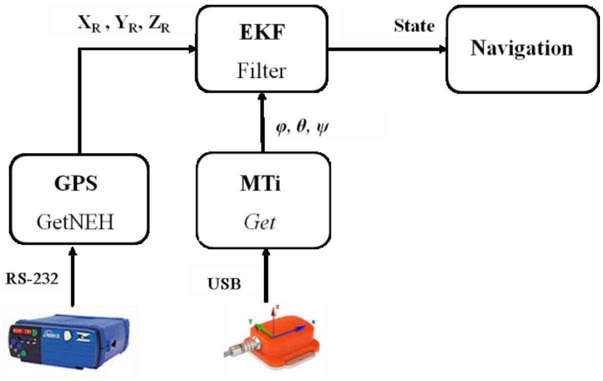


Figure 8.5 EKF

The filter state *X* can be expressed by:

$$X = \begin{bmatrix} s \\ v \\ a \\ \Theta \\ \Omega \end{bmatrix} \qquad \textbf{Eq. 8.2}$$

where *s* is the vector representing the position of the robot in the global reference frame, which usually is the reference frame of the GPS base station, *v* and *a* are the vectors of speed and acceleration of the robot in the robot reference frame which is the same of the AHRS mounted on board, that has been shown in Figure 7.3. Θ and Ω are the angle of attitude/heading in Euler representation and the angular speeds respectively [91] [92]. The continuous time model can be expressed by means of the following equations:

$$\begin{cases} \dot{s} = \bar{R}(\Theta)v \\ \dot{v} = a \\ \dot{a} = 0 \\ \dot{\Theta} = \Omega \\ \dot{\Omega} = 0 \end{cases} \qquad \textbf{Eq. 8.3}$$

The measurements vector is composed by:

$$Z = \begin{bmatrix} s \\ v_X \\ a \\ \Theta \\ \Omega \end{bmatrix} \qquad \textbf{Eq. 8.4}$$

where *s* is the position of the robot as given by the GPS receiver, v_X is the linear speed of the robot, *a,* Θ and Ω are the measures of acceleration attitude/heading and angular speeds, as given by the MTi.

Error covariances for the measures are known data for each sensor, as reported in Table 7.1 and Table 7.2. Moreover the GPS service is able to estimate the GPS measures covariance accessing to the data from the GPS String.

8.3. Self-localization Algorithm through LRF

Self-localization in mobile robotics is a canonical task. A robot needs to know its own position both in a well-known and in a totally unknown environment. Along with localization, map construction, another well known task, is usually associated. The combination of these tasks is known as SLAM or Simultaneous Localization and Map building.

Odometry often leads to unbounded position errors, mainly because it is based on integration of incremental motion information over time. It is well understood that also errors are integrated and consequently odometry alone could not be sufficient to provide precise robot localization.

To avoid this problem several techniques have been developed. Some of these are based on the use of natural landmarks from the environment surrounding the robot. Landmarks can be acquired by different ways, such as video-cameras or range data using sonar systems or laser range finders. Information coming from natural landmarks is usually combined with odometry, GPS data or both, in order to allow the robot to precisely know its position in the environment. Once the localization task is performed, information from the environment, such as landmark position, could be used to build the map.

2D laser range scans, as shown in most of the works cited, have been fully demonstrated to be successful for localization. In order to perform the Self-Localization (SL) task, features extracted from the environment are usually used as natural landmarks. The features extracted could be corners or walls of the environment, which are seen as vertices or lines in the scan representation or, as it will be shown later, more complex geometric entities. Features extracted from each scan, represents a local map, or a feature map, of the environment. To perform Scan-Matching (SM) common features between each scan and the previous one have to be identified. The process of Scan-Matching gives information on the roto-translation between the two scans and consequently about the relative displacement of the robot between the two scans.

The work presented in this section deals with a SM algorithm developed to perform odometry-free dead reckoning, suitable for SL of mobile robot. The proposed algorithm makes use of data coming from laser range finder only and, therefore, does not make use of any other information from the robot, such as speed, orientation or the kinematic model.

The Scan-Matching process is based on consecutively acquired laser scan data. These scans are compared in order to estimate the transformation that maps one of the scans onto the other. This transformation, as shown in Figure 8.6, corresponds to the movement of the robot between the two scans. The transformation that lies beyond is usually defined by a roto-translation between the points of the two scans:

$$P_2 = R(\theta, T) \cdot P_1 \qquad \textbf{Eq. 8.5}$$

where θ is the angle representing the rotation between the two scans, while T is the vector representing the relative translation.

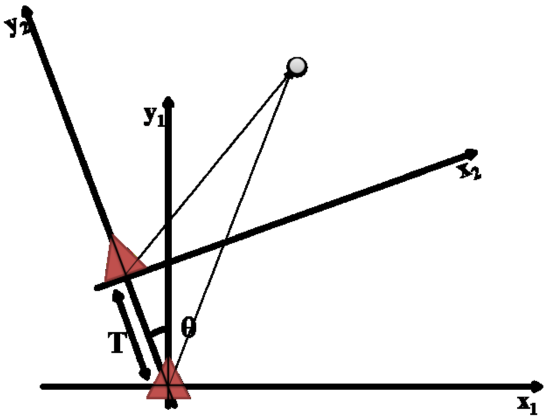


Figure 8.6 Point of the environment seen from two different positions

To match the two scans an error function is usually defined in order to minimize the error by using one of the many gradient descent methods available. For example the Iterative Closest Point (ICP) algorithm works in this is the way, while in [93] it has been shown that this kind of problem has a linear closed form solution.

Scan Matching Algorithms can generally be divided into three main groups: *point-to-feature*, *point-to-point* and, the one this paper deals with, *feature-to-feature*.

The most known algorithms belonging to *point-to-point* category are: the Angle Histogram [94], ICP [95], Iterative Matching Range Point (IMRP) [96] and Iterative Dual Correspondence (IDC) which uses IMRP for rotation and ICP for translation in each iteration [97].

ICP algorithm, for example, is a correspondence based method based on correspondence search and error minimization. While being very simple to be understood, many research activities have been devoted to ICP method, also to overcome its drawbacks, mainly due to the iterative nature of the algorithm itself:

- Computational demand;
- Convergence to local minima;
- Slowness;
- Outliers sensitivity;

As a matter of this, the IDC algorithm, introduced before, is usually seen as an improvement of the ICP. Moreover, in order to speed up execution time of the algorithm some unusual approach, such as GPU implementations [98], have been proved to be feasible.

All point-to-point algorithms need to associate each point in the actual scan with a point in the previous one. Moreover each point can require all points in the other scan to be checked. Some search restrictions can reduce computational costs. In the pioneering work [97], the adopted approach is to start with an approximate alignment of the two scans using odometry and then iteratively improve the alignment of the two scans.

An example of *Point-to-feature* is reported in [99] where data points from current scan are matched to more global features, lines in this case, from a global a-priori map. In this type of algorithm, to increase the efficiency and robustness of the process, sensor data have to be efficiently processed before, comparing them with those stored in the a-priori map. Data processing could be more crucial if the sensor information is used to even build or correct the map. These algorithms lie between the previous and the following ones.

In *feature-to-feature* matching approaches, features such as line segments, corners [100] or range-extrema [102] are extracted from laser scans, and then matched. Such approaches interpret laser scans and require the presence of chosen features in the environment. Features can be more abstract as in [103] where arc of circle are used as features representing landmarks associated with real environment such as tree-like objects or cylindrical columns.

Pioneering work are presented in [104] and [105] where it is again clearly highlighted that the key for right scan-matching, and so for mobile robot localization, is the association of the features between the two scans. Data association is the process of relating features observed in the actual feature map to features viewed in the previous feature map or, as in the case of point-to-feature SM, to features stored in the a-priori global map. Correct feature association is crucial to perform the self-localization.

Further details about this aspect will be given in the following sections, were the proposed algorithm will be presented.

As reported in [104] or in [105], feature-to-feature SM algorithms can be represented following these steps:

1. Extract features within each scan obtaining the actual features map.
2. Data Association, in other words, search for pairwise corresponding features from both features maps.

3. Calculate the roto-translation between two consecutive scans.
4. Update the robot pose.
5. Save the current feature map for the following iteration.

After the feature extraction, each laser scan is transformed in a feature map. A feature map consist of a set of geometric entities, such as, points, lines and so on, in a local coordinate reference frame. The feature map created by the last iteration could be used even to update the global map of the environment, if Map-Building is also running.

In [100] several feature extraction algorithms are presented covering different performance aspects such as execution time, number of features found and sensitivity to angular resolution of the laser range finder. For the work presented here the *Split and Merge* algorithm has been used. Split & Merge is one of the most known and used feature extraction algorithms for laser range scanner data. As shown in [103] and [100], it is simple and fast, two important factors for real-time applications, but, similarly to almost all other FE algorithms, it is sensible to data noise and it does not work well with complex or irregular environments.

In the case of laser range finders, points, and so features themselves, are detected almost simultaneously. In case of robot motion, or motion of something in the environment, data could be affected by noise. Being data acquired almost at the same time or with sufficiently small temporal differences, let them to be represented with very precise local coordinates using, in the worst case, a motion compensation step.

The set of features extracted by each scan will be considered as a local map, so that it is a representation of the environment. Normally the set of features found, changes during the motion of the robot in the environment, so to have new features if a previously occluded area gets into sight and to lose features in the opposite case. The process of scan-matching lets to work with features that are sought in both the two consecutive scans.

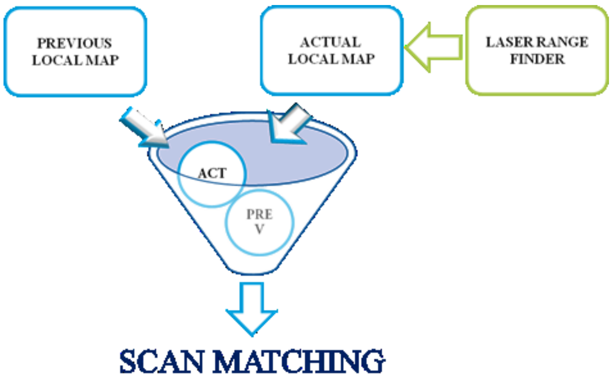


Figure 8.7 Scan Matching

In [106] a graph theoretic method that is applicable to data association problems is presented. The feature map is transformed into a feature graph in which each node represents a feature while the geometric inter-features relationship establishes an edge. This algorithm relies on the fact that a local map will possess invariant inter-feature relationships (e.g. distance between two corner, angle between two lines or length of lines).

In this way, data association between two local maps can be expressed as the graph-theoretic problem of finding the Maximum Common Subgraph (MCS) between two graphs. This results in the maximum portion of the local map where all the inter-feature constraints are mutually satisfied. Anyway the problem of matching two feature-maps could be seen as a canonical scan matching. The main difference is in the number of points to be matched. See [105] for further details on the problem of data association.

As said before, once that common features, seen in two consecutive scans, are identified, the self-localization problem is transformed in a problem of error minimization, because of the roto-translation matrix, which minimizes the error between the two set of features, has to be found. This will be the approach of the presented algorithm.

After the data association process is done, two sets of features are found. Each feature in each set is related to only one feature in the second set. The relationship between the two sets of features has been introduced in Eq. 8.5. The roto-translation matrix R is defined by:

$$R(\theta,T)=\begin{bmatrix}\cos\theta & -\sin\theta & T_X\\ \sin\theta & \cos\theta & T_Y\\ 0 & 0 & 1\end{bmatrix} \qquad \textbf{Eq. 8.6}$$

The error function to be minimized is defined in Eq. 8.7. It is the mean square error of the Euclidean distance.

$$E=\|P_1-P_2\|=\|P_1-R(\theta,T)\cdot P_1\| \qquad \textbf{Eq. 8.7}$$

Right values of θ and T are found, as said before, minimizing this error thus solving:

$$[\theta,T]=\min_{\theta,T}\|E\| \qquad \textbf{Eq. 8.8}$$

The algorithm proposed in [106], even if deals with point-type and line-type features, mainly consider distance as an inter-feature relationship. The angle between two lines is considered only in case of line-like features.

As said before, the algorithm presented here, called D&A Algorithm, works only with point-type features and so it has to use distance as inter-feature relationship but, as it will be clearer later, it introduces also angle-like inter-feature relationship. Moreover even if it draws inspiration from some assumption reported in [106] it avoids using algorithm from graph theory.

The Distance and Angle (D&A for short) algorithm is based on the following assumptions:

1. Each inter-feature relationship implicitly defines a segment *d* connecting two point-type features; the relationship is defined by the length of the segment itself.
2. The inter-feature relationship defined by *d* must be invariant between two feature maps if features are both seen in the two maps.

3. The angle between two segments *d* must be, as for inter-feature relationship, invariant between two maps if they are visible in the two maps.

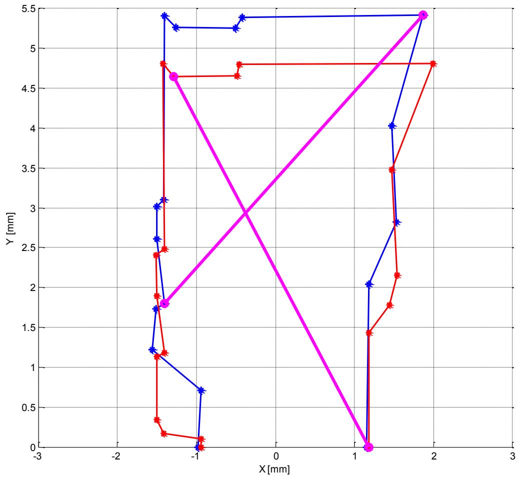


Figure 8.8 Example of wrong data association between the first maps (red) and the second one (blue).

Considering only the length of the relationship, as done in [106], could led to wrong data association, as it could be clearer looking at the case shown in Figure 8.8, where the two purple (diagonal) segments are wrongly seen as common features just because they have similar length. Otherwise, if more than one inter-feature relationship is considered, the process of data association could be surely improved.

The whole process of data association consists of finding the association *d*-*d* between the two feature maps. The goal is to find segments of similar length among those maps. Anyway, as said before, the association has to rightly identify couple of points between the two maps.

To avoid wrong association, the D&A algorithm works as follow:

1. A first d_R-d_R association between two maps is found searching firstly between longer segments (purple segments

in Figure 8.9). The first association is initially considered as reference.

2. Following *d-d* association are searched (green segments in Figure 8.9).
3. The angle between each segment *d* and the reference d_R for its map should be “close” to the one in the other map to let association to be considered as valid.
4. If any *d-d* association were found, the reference is considered as wrong so it is changed and the algorithm starts again.

An example of the results is shown in Figure 8.9.

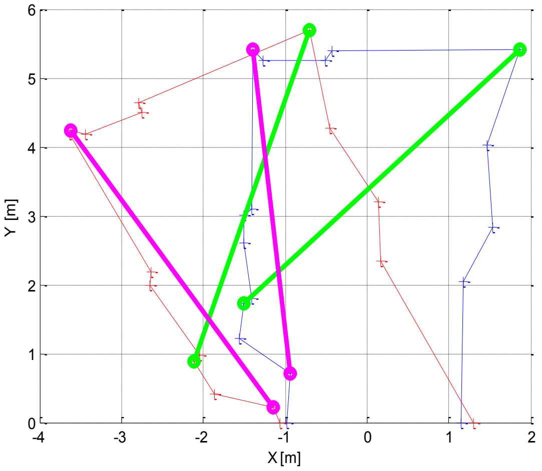


Figure 8.9 Example of right data association between the first maps (red) and the second one (blue).

Purple segments represent the references, while the green ones represent the second data association. The difference of length (first inter-feature relationship considered) between the two green segments is less than a given threshold and, as stated in point 3, the difference of angle, between each segment and the reference one, is less then another given threshold. Only in this case the association is considered as valid. As a matter of fact the greater is the number of valid data association found, the greater is the validity of reference chosen in the first iteration of the algorithm.

Moreover, each right association increases by two the number of common valid features between the two maps. The algorithm as, stated in point 1, starts comparing longer segments between the two feature maps. Moreover even with the following association the search-task is firstly performed among longer segments.

In order to perform the search process among inter-feature relationship, a matrix, as indicated in Table 8.1, is built for each map. The matrix, called *distance-matrix*, contains on its first two columns the incremental number of two features, while the inter-feature distance is stored in the third column. The number of rows reflects the total number of inter-feature relationship found.

Going on with the data association process, each time an inter-feature relationship from the first map is associated with one in the second map, the correspondent rows are deleted from the relative distance matrix. This is done because each relationship of a map can be associated with only one of the other.

It is useful to highlight again that both the inter-feature distance and the angle of the segment with respect to the reference one, must lie below to the given thresholds in order to consider the association as valid.

Table 8.1 Distance Matrix

i	*j*	d_{ij}
1	2	d_{12}
1	3	d_{13}
…	…	…
2	3	d_{23}
2	4	d_{24}
…	…	…
n-1	n	$d_{(n-1)n}$

The LRF has been used to acquire data over 180 degrees using a resolution of 1 degree. Using this resolution such a device is able to perform scans at 75Hz, allowing the user to have each new scan in about 13 ms. The device is able to work with higher resolutions but, as it is shown in [100] the higher resolutions don't improve the performance of the Split & Merge algorithm, in terms of number of features found, while the lower resolutions allow to have features maps at an average rate of one each 4.3 ms.

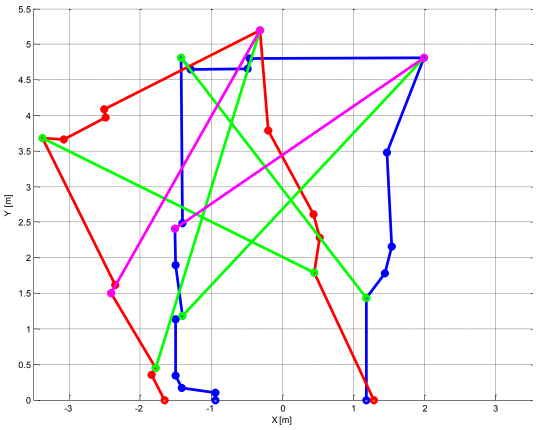


Figure 8.10 Right data association between the first maps (blue) and the second one (red). Three data association were found correspond to six features with the following results: TX= 3.27 [mm] TY= -10.27 [mm] and θ= 25.97°

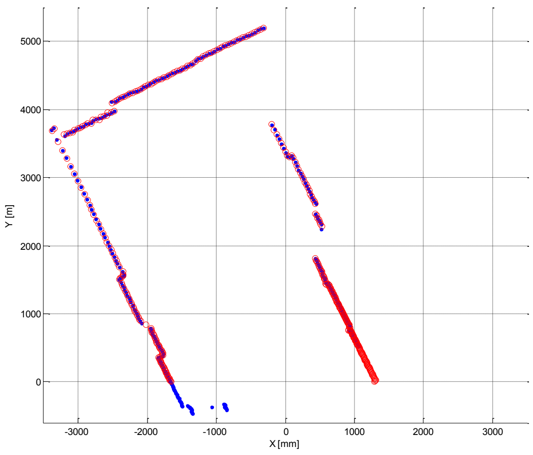


Figure 8.11 How the first maps (blue) is mapped into the second one (red)

The SM process is performed in around 25ms so the whole SL process is performed in less than 35ms (25+4.3+4.3=33.6).

Figure 8.10 shows three correct data association between two consecutive scans. This means that six points for each scan are correctly associated. In this case the mean square error on the position is less than 5mm. Figure 8.11 shows how points from the first scan are exactly roto-translated on the second one by means of the information obtained from the SM process.

The algorithm, written in Matlab™ environment, has been compiled with MATLAB Builder NE tool, [101], in order to obtain a dynamic link library suitable to be used inside C# code, thus into an MRDS Service.

Because of both the simulated LRF and real one have the same output, an array of integer, the service can be used both in simulation and in real navigation. It is yet under test the fusion of the output of D&A, which is a set of three incremental displacement *ΔX ΔY $\Delta\theta$*, into the state estimation stage presented, in section 8.2 as reported in Figure 8.12.

The algorithm has been firstly tested off-line. Data were acquired using the 3Mo.R.D.U.C. robot presented in section 5.2. Tests have been performed using Matlab™ environment running on the onboard laptop which has a Pentium3™ CPU working at 1Ghz.

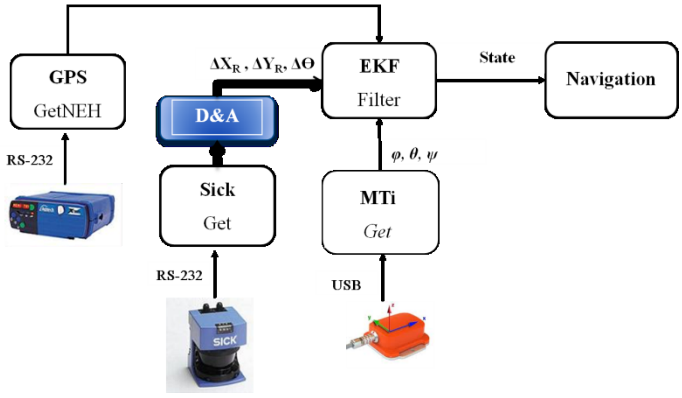


Figure 8.12 State estimation enhanced with D&A Algorithm

Before to move it to other platforms it has been tested in simulations involving the other platform presented such as the U-Go robot or the Segway.

8.4. Self-localization Algorithm through Sonar

One of the possible applications for the U-Go robot, as reported in section 5.3, is to operate inside greenhouses for precision farming applications.

Greenhouse activities can be very dangerous and uncomfortable because of chemicals, high temperature and humidity. With this environmental condition, even common agricultural operation can become heavy and stressful. Moreover, because of high temperature and humidity, operators often do not correctly wear safety clothes, increasing health risks. Well-known alternative could be building robots able to replace humans in greenhouses, for maintenance and production activities as picking of ripe fruits or spraying chemicals [107].

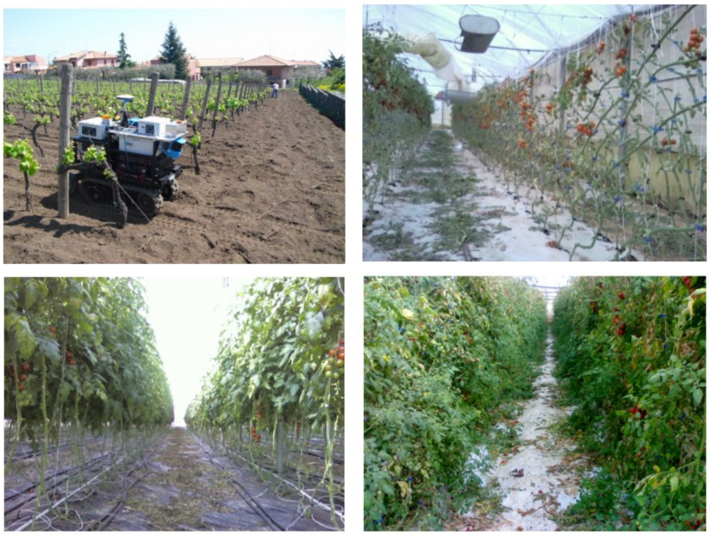


Figure 8.13 Different Environments

This section describes the used algorithm for localization and navigation of the U-Go robot in typical agricultural applications, such as navigation and operation inside vineyards and greenhouses, by means of the US-Ring presented in section 7.4.

The six lateral sensors are used to make the robot navigate, keeping the centre of the corridor. In particular, the proposed algorithm estimates the position and the orientation error of the robot from the ideal trajectory, which is the centre between the two lateral walls. The algorithm works supposing the two lateral walls to be rectilinear. As a matter of this, data coming from sensors on a side of the robot have to face something close to a line even if plants can really strongly change this shape, see Figure 8.13.

Data from each side are used to estimate, using a linear interpolation, parameters of a line, which represent the plant lateral wall. The line equation used is the well known $y = m \cdot x + q$.

Each line is so defined by the couple m and q. The robot displacement from the centre, in robot reference frame, can be evaluated by:

$$y_r = \frac{(q_R + q_L)}{2} \quad \textbf{Eq. 8.9}$$

The orientation error from the ideal trajectory is defined by:

$$\alpha_r = \frac{\tan^{-1} m_R + \tan^{-1} m_L}{2} \quad \textbf{Eq. 8.10}$$

The subscripts R and L stand for right and left. In order to improve the performance an EKF is used to estimate all the parameters of the line.

The filter state X can be expressed by:

$$X = \begin{bmatrix} m \\ q \\ p \end{bmatrix} \quad \textbf{Eq. 8.11}$$

where m and q are the two couples of parameters identifying the two lines. Each sensor needs two parameters to be localized, which means twelve parameters, indicated by *p*. The total number of variables in the state vector of the EKF is sixteen.

The continuous time model can be expressed by means of the following equations:

$$\begin{cases} \dot{m} = 0 \\ \dot{q} = 0 \\ \dot{p} = 0 \end{cases} \qquad \textbf{Eq. 8.12}$$

The filter estimates the position of the sensor, with respect to the robot reference frame, in order to compensate errors in the positioning of the sensors and, of course, line's parameters by means of following six measurements.

$$Z = \begin{bmatrix} m_L \cdot x_{L1} + q_L + y_{L1} \\ m_L \cdot x_{L2} + q_L + y_{L2} \\ m_L \cdot x_{L3} + q_L + y_{L3} \\ m_R \cdot x_{R1} + q_R + y_{R1} \\ m_R \cdot x_{R2} + q_R + y_{R2} \\ m_R \cdot x_{R3} + q_R + y_{R3} \end{bmatrix} \qquad \textbf{Eq. 8.13}$$

where the (x_{Li} y_{Li}) and (x_{Ri} y_{Ri}) pairs are the positions of the sensors, all these parameters have been previously indicated by the vector *p*.

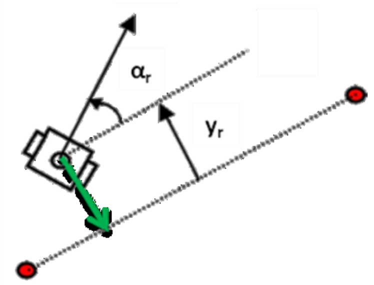


Figure 8.14 Output

Even in this case, this algorithm has been tested with real-time in-field measurements with Matlab™. Due to its mathematical simplicity the algorithm has been rapidly translated into C# code, thus allowing to deploy an MRDS service. This service can be used both during simulation and to perform in-field tests.

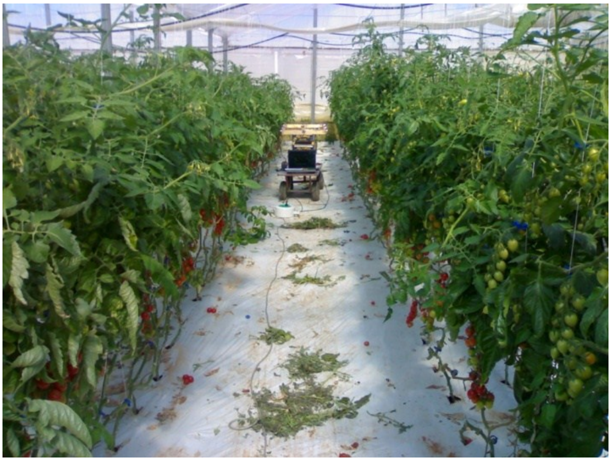


Figure 8.15 Inside Greenhouses test

Several in-field tests, as shown in Figure 8.15, have demonstrated the ability of this approach to guide the robot in between of plant corridors.

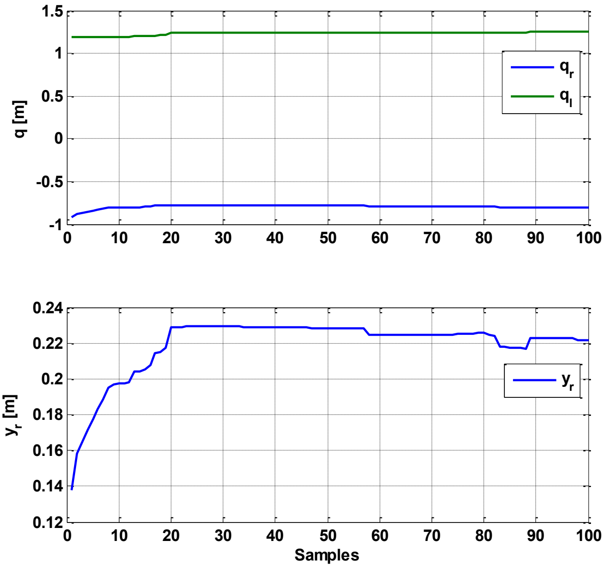


Figure 8.16 Offset

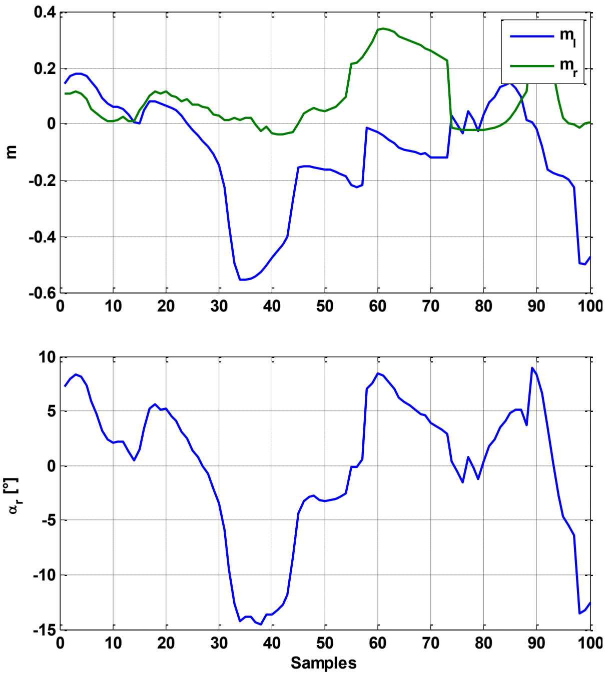


Figure 8.17 Orientation

Figure 8.16 and Figure 8.17 show a set of data collected, during the test phase. The U-Go robot was moving straight inside a corridor with an offset with respect to the center of about twenty centimeters.

Figure 8.16 shows q_l and q_r parameters on the upper side while the displacement of the robot y_r is shown in the lower side. Although the noise, after twenty iterations the algorithm identifies the correct displacement. The value of y_r is evaluated using Eq. 8.9.

Figure 8.17 shows m_l and m_r parameters on the upper side while the orientation of the robot α_r, evaluated using Eq. 8.10, is shown in the lower side. This will be used as feedback to control the navigation of the robot inside the corridor. The control law will be presented in section 8.7.

8.5. Computer Vision for Navigation

An algorithm based on computer vision has been developed in order to allow, during outdoor navigation, terrain morphology detection. This is used both to detect the presence of obstacles in the mission path and to recognize and define a drivable surface in the proximity of the robot. An approach similar to that described in [108] is adopted.

However, while in [108] a simple camera is installed, in our application, the stereo camera pair will be used. Image processing is based on the Intel OpenCV [86] open source library that provides useful, quick and reliable basic functions to build up computer vision algorithms. An auto-calibration tool for stereo camera has been developed, in order to easily and quickly compute cameras intrinsic, extrinsic and distortion parameters.

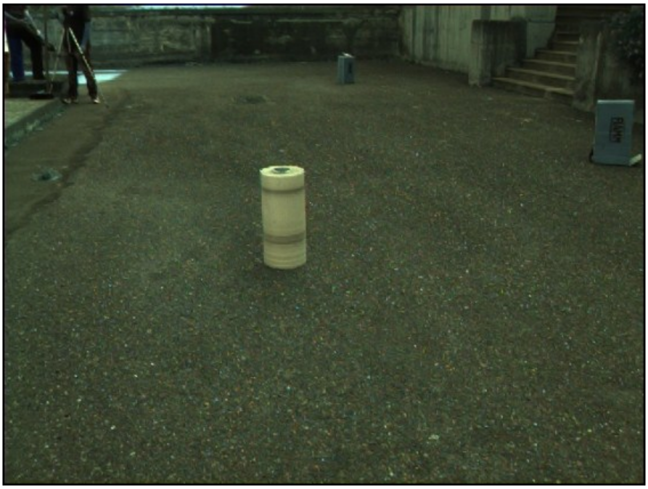


Figure 8.18 Original Image

The computed parameters are used in the real-time image processing procedures, so as to obtain a top down “bird’s eye” view from which information about the path in front of the robot can be extracted. For example, the image shown in Figure 8.18 is transformed in the image shown in Figure 8.19

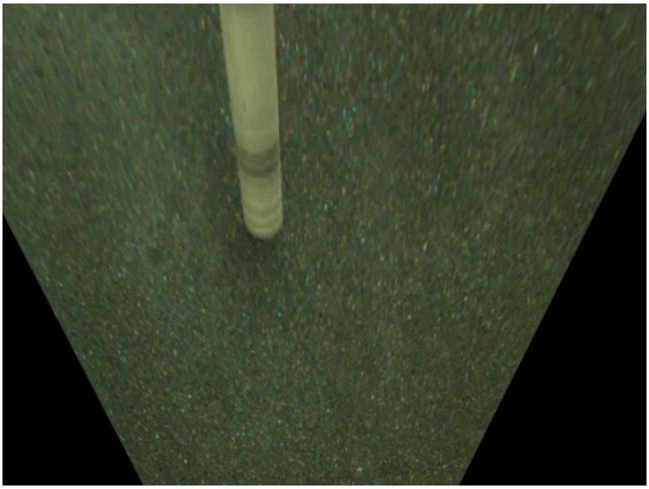


Figure 8.19 bird's eye view

A region of the “bird’s eye” image near the robot, called Kernel, is used to build such a 3D Matrix containing information about its own colors.

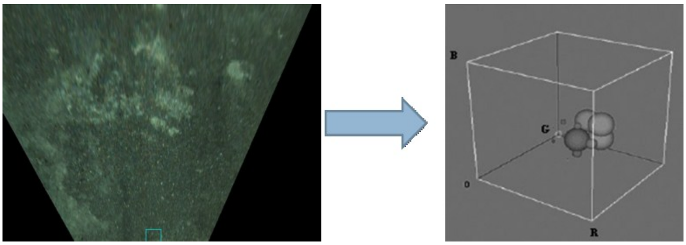


Figure 8.20 Building the 3D Matrix

Building this matrix, each pixels, by means of considering the RGB components, increments by an amount *D* the value of the corresponding element of the matrix and by a smaller amount *d* neighbor elements. Each pixel is mapped in a reduced RGB space where clusters, concentration of high values, are generated.

Once the matrix is built, pixels of the image (Figure 8.21) are analyzed in order to estimate if their color is present in the constructed matrix.

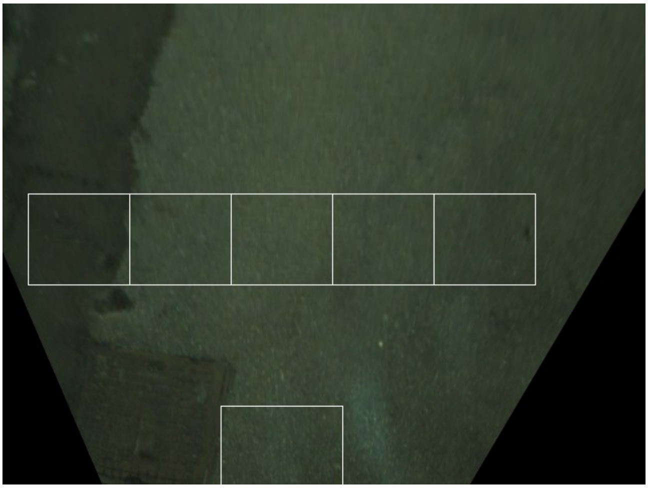


Figure 8.21 Initial bird's eye image

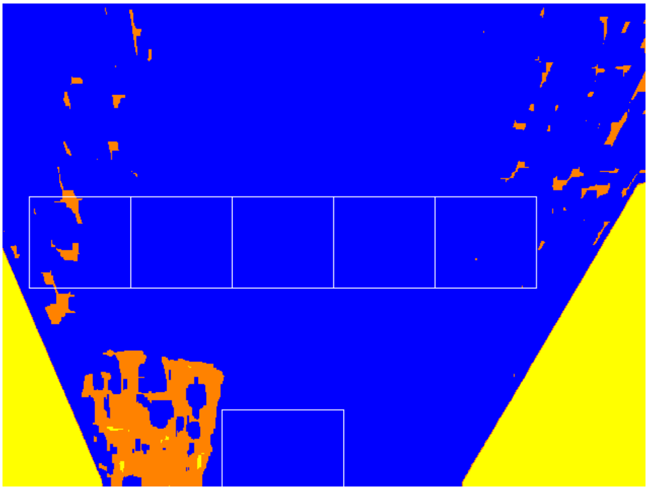


Figure 8.22 Output image

By means of the value of the 3D Matrix at the coordinates connected to the pixel RGB color, three sets of pixels are created (Figure 8.22):

- Blue pixels represent Fully Drivable Surfaces
- Yellow ones represent Un-Drivable Surfaces
- Orange ones are in between the previous two cases

As said before, the adoption of stereo cameras also allows detecting obstacles and getting out elevation data from the images, in order to find drivable corridors.

To do so, both left and right images (Figure 8.23 a and b), transformed in "bird's eye" view (Figure 8.23 c and d), are used to create a "disparity image" (Figure 8.23 e).

The disparity image is the difference between left and right images. In presence of obstacles non-zero pixels are present. Non-zero means clearer colors which define borders of obstacles.

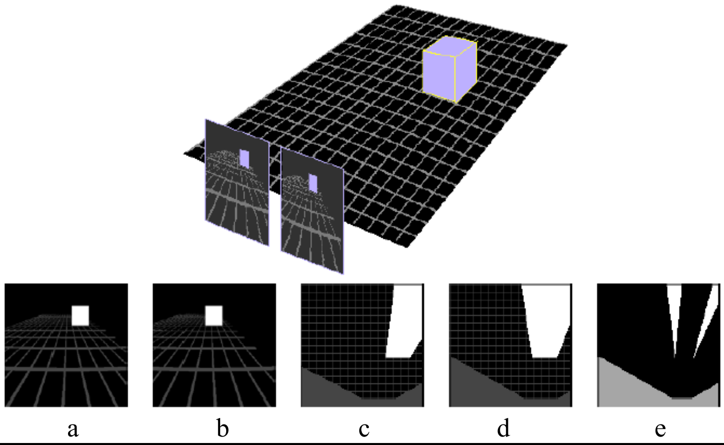


Figure 8.23 Obstacle Detection Overview

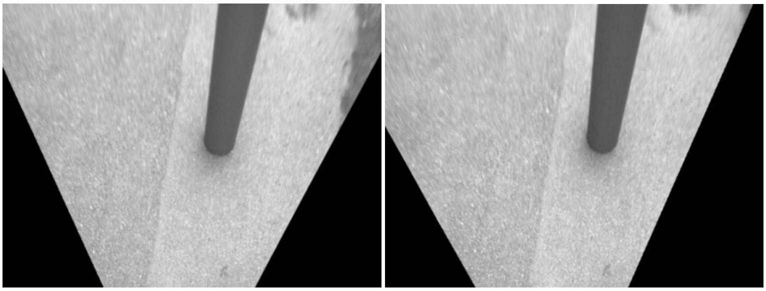


Figure 8.24 Left and right images

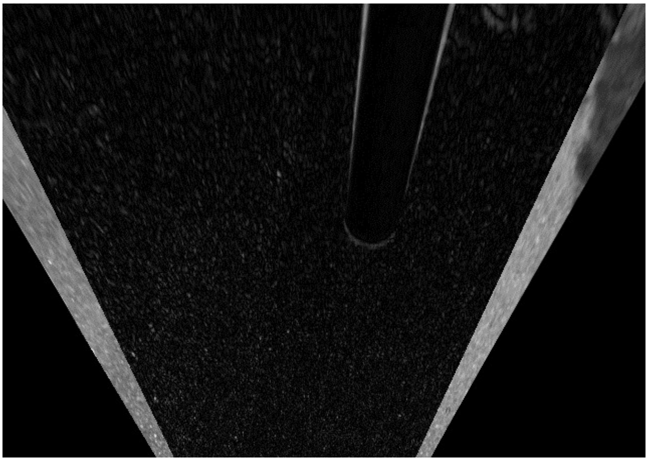


Figure 8.25 Output

Figure 8.24 and Figure 8.25 show some of the in-field tests performed in order to tune the vision algorithm. Some other computation tasks allow to find the location of obstacles.

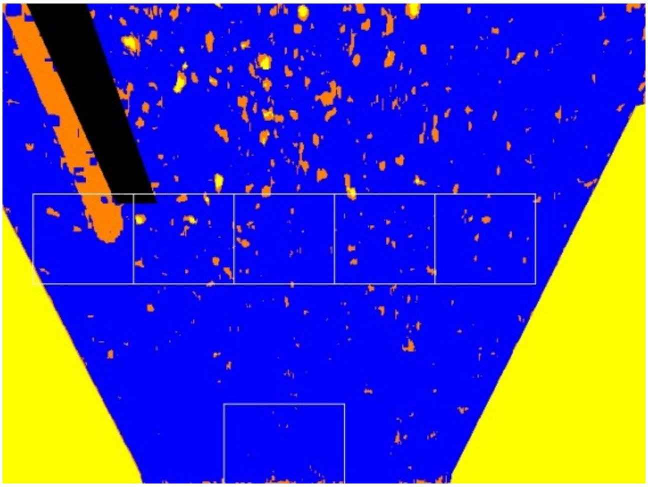


Figure 8.26 Overall output

In detail, the Figure 8.26 shows the output. Drivable surface are painted in blue while not-safe surfaces are painted in yellow. In presence of an obstacle a black region is shown.

The white bordered squares, in the base of the image shown in Figure 8.22 and in Figure 8.26, are the so called kernel, while the other five are particular region of interest. As it will be clearer later in next section, they are used to control robot movements.

This algorithm has been developed and tested both with the U-Go robot and the Robovolc. These two robots are usually used outdoor. The application involved in these two tasks, Drivable Surface detection and Obstacles detection, has been developed as a standalone application written, as said before, using C++ programming language.

Information coming from those algorithms is provided to a dedicated MRDS service through simple UDP communication.

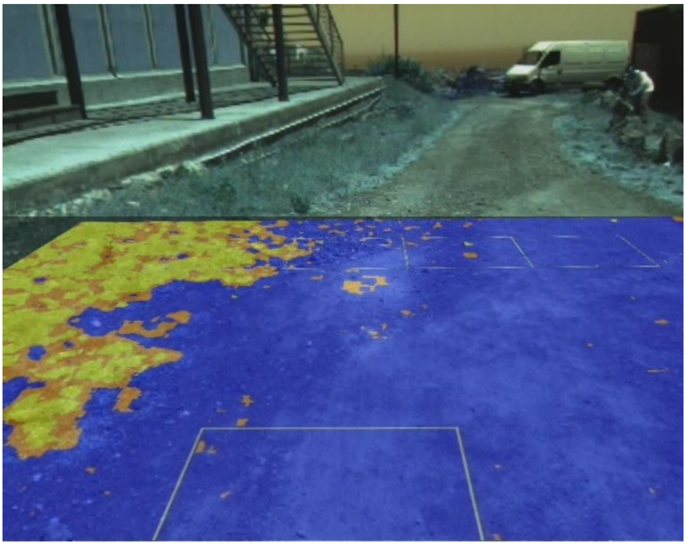


Figure 8.27 Drivable Surface

Figure 8.27 and Figure 8.28 highlight the results of this algorithm. They are both obtained transforming the output image of the algorithm, like the ones shown before. Knowing all the camera parameters, it is possible to fuse it with the original image acquired by the camera.

On the left part of Figure 8.27 it is possible to see how the grass at the border of the road is painted in yellow and orange, meaning that this is an undrivable surface. Figure 8.28 shows how an obstacle is identified: its borders are marked with two darker rectangles.

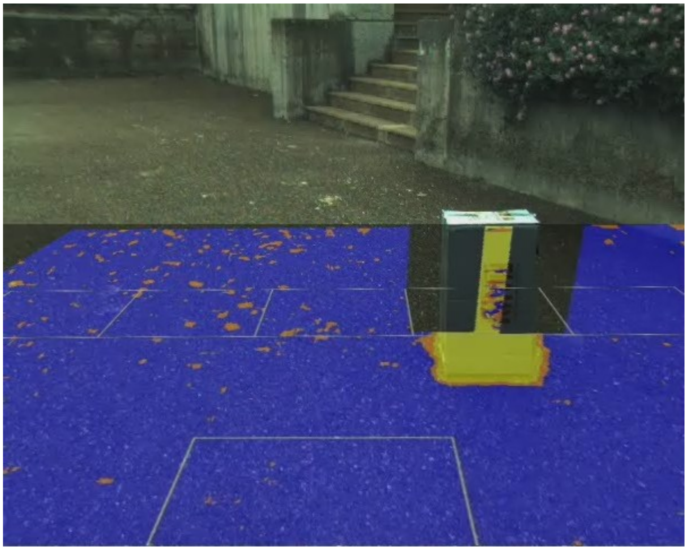


Figure 8.28 Obstacles Detection

8.6. Trajectory attractive Potential Field Method

The Potential Field Method (PFM) [109] has been used for the motion control and obstacle avoidance. PFM is not a novelty but allows to build the software and to operate with high modularity.

As is well known, PFM has been first introduced by Khatib in the far 1979 [109]. Since the beginning till nowadays researchers have deeply studied PFM and applied it in fields like mobile robot navigation and robotic manipulator control. The popularity of this method is due to its simplicity and elegance. Moreover PFM can be quickly implemented providing good results.

The key idea of PFM are virtual forces acting on a robot. Obstacles exert repulsive forces onto the robot while the target applies an attractive force. In the case of mobile robots navigation, the sum of all forces, called resultant force, determines the consequent direction of motion.

Theoretically speaking, a potential field associates coordinates within a space with scalar or vector potential values. In the real-world, the space is a 3D or 2D representation of the environment and the potential values are often energy levels, in case of a scalar field, or physical forces, in case of a vector field.

Usually it is assumed that a high potential value means a highly undesirable position and so the lowest point on the field, a point of global minimum potential, is considered as the optimal solution.

As a matter of this, obstacles are defined in a way to increase the potential around themselves. However PFMs are affected by the well known problem of local minima. Indeed this is not the only problem in using PFM, but during the years researchers have faced and solved different lack of this method and they have often proposed some improved variants of it. See [111] for further details.

However, despite this intensive study, very few works have been focused on concrete trajectory-following using PFM.

One common way to mobile robot navigation is to acquire waypoints during an exploration stage and use them to plan future movements through the previously explored environment. This strategy works really close as having a map of the environment. Anyway, theoretically speaking, one of the aims of SLAM techniques for a mobile robot is to build a map which has to be used for navigation. In [49] and [50] for example, to achieve a goal identified by a GPS waypoint, a sequence of movements through several GPS waypoints, which act as sub-goals, allow the robot to safely move towards the goal. Usually a waypoint is considered as reached if the distance between the robot and the target itself is less than a given threshold. This threshold is called ρ_G. In other words, the robot must be within a circle, we will call

Confidence Circle (CC), which is centered in the associated waypoint and has radius ρ_G.

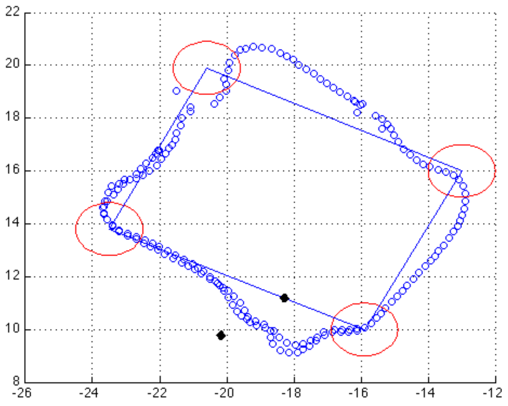


Figure 8.29 Square path test with obstacles and multiple sub-goals. Distances in meter.

Figure 8.29 shows a test reported also in [49]. It is the classical square path test through four waypoints. The red circles represent the four CCs, the blue lines are the supposed trajectories through the waypoints while the black dots are obstacles.

This strategy can be followed anytime a path has to be followed by a robot. The global path can be split in a sequence of sub-goals connecting the starting point with the global goal. Between each sub-goal pair, the robot should follow, even if obstacles could be found, the straight-line trajectory between the two sub-goals.

Working this way, any path, even if complex, could be seen as sequence of waypoints connected by a line, as shown in Figure 8.30.

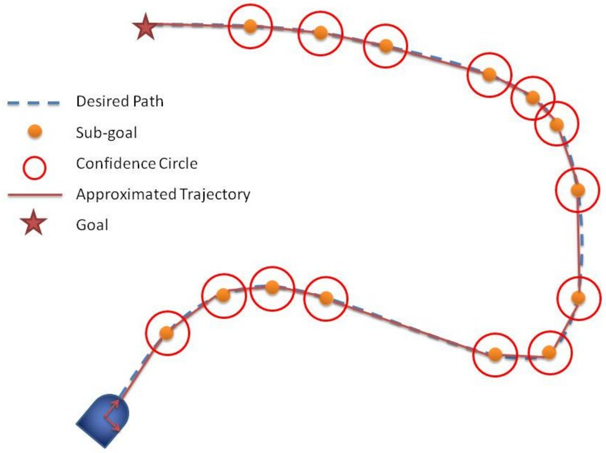


Figure 8.30 Complex path definition

As reported in [92], [112] and [113], this force is used to drive the robot.

In each position of the space the total virtual force acting on the robot can be obtained using the following equation, as is shown Figure 8.31.

$$\overrightarrow{F_T} = K_A\overrightarrow{F_A} + K_R\overrightarrow{F_R} \qquad \textbf{Eq. 8.14}$$

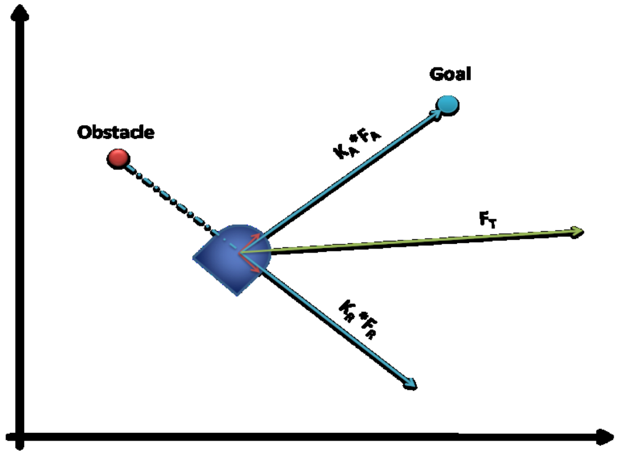


Figure 8.31 The force acting on the robot

What presented till now is well known in the literature. Almost all the references that can be found dealing with PFM are focused on PF definitions, local minima problem and some other peculiarity of such a method. Even in [92], PFM is presented as a way for obstacle avoidance (OA). During the years PFM has been widely used just for OA. In [114], for example, a system for trajectory tracking is presented but PFM is used only for the OA task.

Virtual Obstacle Potential Field (VOPF) control algorithm is presented in [109]. This method was built to force the robot to approach a given path. This algorithm uses two more virtual obstacles whose location is changed in order to force the robot to move along the desired path.

The algorithm we propose here is a PFM algorithm which intrinsically drives a mobile robot towards a goal, following a desired trajectory. The trajectory chosen is the one connecting two sub-goals.

Thus one more attractive force is used, as shown in Figure 8.32. This force is connected to the distance between the robot and the trajectory. For this reason, the algorithm presented here is called *Trajectory-attractive PFM* (TA-PFM).

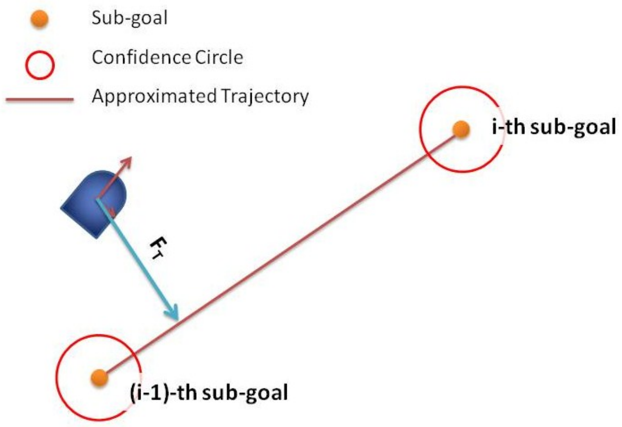


Figure 8.32 The effect of the trajectory on the robot

Working this way both the trajectory-following and the obstacle avoidance tasks are faced at the same time so that the robot can be driven the same way done with classical PFM.

$$\overrightarrow{F_T} = K_A\overrightarrow{F_A} + K_{AT}\overrightarrow{F_{AT}} + K_R\overrightarrow{F_R} \qquad \textbf{Eq. 8.15}$$

Eq. 8.15 is an extension of Eq. 8.14 and the result is shown in Figure 8.33.

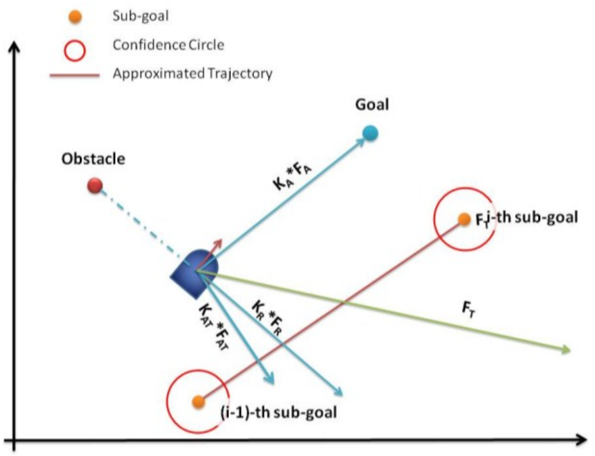


Figure 8.33 Forces acting on the robot with TA-PFM

8.7. Control of the robots

The control algorithm for the robot has been also developed using MRDS.

In the control diagram, therefore, services receive and process sensors information. Data from all these sensors are, first of all, processed by the EKF shown in 8.2, in order to evaluate robot state. The navigation algorithm takes care of generating the control reference for the used robot. The TA-PFM algorithm has been used for the motion control to avoid collisions, with the obstacles detected by sensors, during the motion itself, without losing the main task.

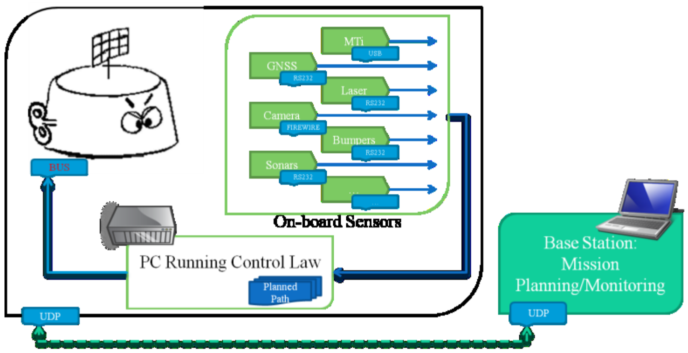


Figure 8.34 Navigation Overview

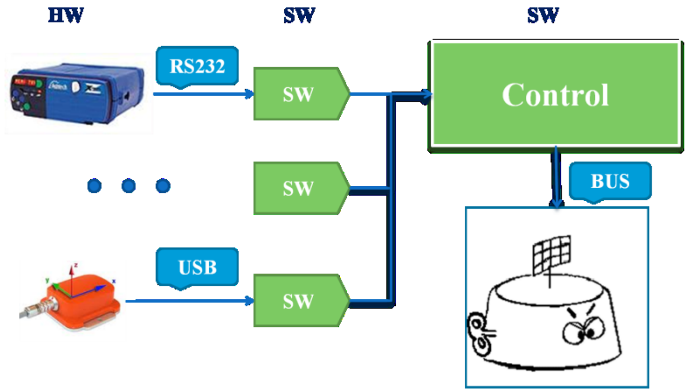


Figure 8.35 Multisensor Data Fusion

Figure 8.34 and Figure 8.35 show how building blocks are connected and highlight how hardware devices are projected into software equivalents, which are MRDS services.

Figure 8.36 shows how information from computer vision is transformed into virtual forces, suitable to be used for the PFM. Obstacles obviously apply repulsive forces on the robot, so un-drivable surface do. Small attractive forces are also generated by these drivable surfaces.

The white bordered squares, in the top of image shown in Figure 8.22 and in Figure 8.26, are used to evaluate five forces, acting on the robot. A mean off all the pixels in the square weighted by their color, is used to calculate the force value, while centre of the square is used to evaluate forces directions.

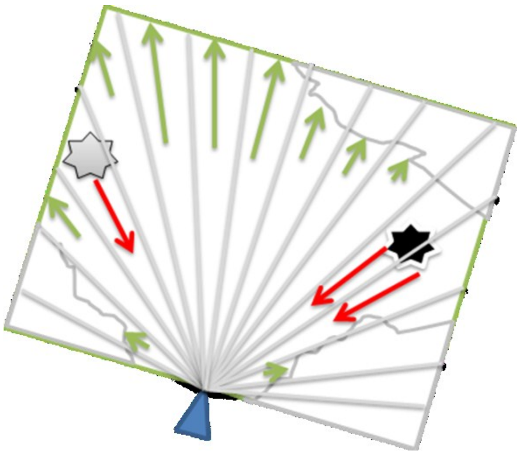


Figure 8.36 Computer Vision virtual forces

The US-sonars act on the robot in an identical way. The force generated on the robot is proportional to the distance from the ideal trajectory and oriented toward it. The resulting force is shown as a green arrow in Figure 8.37.

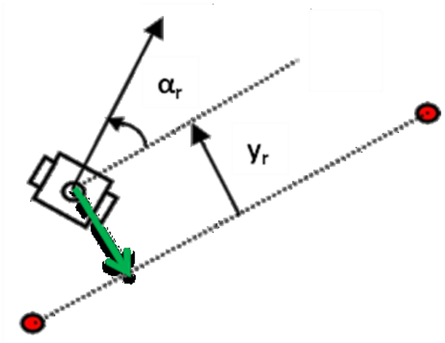


Figure 8.37 US Sonar forces

The way all the information coming from sensors, transformed into forces, acts on the global robot behaviour, is shown in Figure 8.33. This approach has shown a key-aspect in its high modularity. In case a sensor is not used or, in the worst case, it fails, the result is the absence of a force, while the global architecture, shown in Figure 8.38, is always the same.

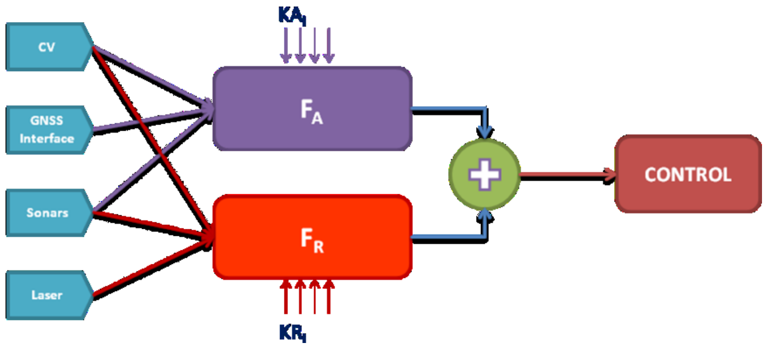


Figure 8.38 Overview of the PFM

The control inputs for the robot have been chosen as the linear and angular velocities *v* and *w*. The control strategy adopted is close to the one reported in [113]. In details the linear speed is simply inverse proportional to the repulsive force F_R, while the angular speed is controlled by the following equation:

$$w = K_W(\delta - \theta) \qquad \textbf{Eq. 8.16}$$

where K_W is a proportional constant, δ is the orientation of the total force F_T, while θ is the actual robot orientation.

8.8. Conclusions

If there could be a key word for all the research done in the field of mobile robot it is: *Modularity*.

From sensor point of view, different setup involving different sensor suite can be arranged both in simulation and in real in field tests. From an application point of view, all presented robots can be used with the same control algorithms; this is possible because, at an higher level, the robots interact with the controller by means of the same inputs, as stated in section 8.1.

The key aspect of the control algorithm is its simplicity.

Appendices

Appendix A **List of Figures**

Appendix B **List of Tables**

[1] RAPOLAC Project home page. [Online] http://www.RAPOLAC.eu.

[2] AMR home page. [Online] http://www.amr.co.uk.

[3] Chua, C. K., Leong, K. F., Lim, C. S.; *Rapid Prototyping: Principles and Applications (3rd Edition)*, World of Scientific Publishing (2010).

[4] Koren, Y.; *Computer Control of Manufacturing Systems*, McGraw-Hill Singapore (1983).

[5] Hecht, J.; *The Laser Guidebook (2nd Edition)*, McGraw-Hill NewYork (1992).

[6] Taraman, K.; *CAD/CAM Meeting Today's Productivity Challenge*, Computer and Automated Systems Association of SME (1982).

[7] Bonaccorso, F., Cantelli, L., Muscato, G.; *An Arc Welding Robot Control for a Shaped Metal Deposition Plant: Modular Software Interface and Sensors*, Industrial Electronics, IEEE Transactions on , vol.58, no.8, pp.3126-3132, Aug. 2011.

[8] Zhang, A. Y., et al.; *Weld deposition based rapid prototyping: a preliminary study*, Journal of Materials Processing Technology, Vol.135, pp. 347 – 357, 2003.

[9] Messer, R. W.; *Principles of Welding: Processes, Physics, Chemistry and Metallurgy*, Wiley-Interscience (1999).

[10] Baufeld, A. B., Van der Biest, O., Gault, RS.; *Additive manufacturing of Ti–6Al–4V components by shaped metal deposition: Microstructure and mechanical properties*, Journal of Materials and Design, Vol. 31 pp 106-111, 2010.

[11] Moore, D.S., Naidu, S., Ozcelik, K.L.; *Modeling, Sensing and Control of Gas Metal Arc Welding*, Elsevier (2003).

[12] Eagar, T. W., Tsai, N. S.; *Temperature fields produced by traveling distributed heat sources*, 64th Annual AWS Convention, 1983.

[13] Eagar, T. W., Tsai, N. S.; *Changes of welding pool shape by variation in the distribution of heat source in arc welding*, Modeling of casting and Welding Processes, AIME New York (1984).

[14] Bonaccorso, F., Bruno, C., Cantelli, L., Longo D., Muscato, G.; *Control of a Shaped Metal Deposition Process*, 4th International Scientific Conference on Physics and Control, PHYSCON 09, Catania, Italy, 1-4 September 2009.

[15] Rosenthal, D.; *The Theory of Moving Source of Heat and its Application to Metal Transfer*, Transactions ASME, Vol 43(11), pp.849-866.

[16] Yagishita, S., Kanda, M.; *Arc Welding Robot Systems for Large Steel Constructions*, IEEE Transactions on Industrial Electronics, Vol. IE-30, No. 3, 1983.

[17] Muscato, G., Spampinato, G., Cantelli, L.; *A Closed Loop Welding Controller for a Rapid Manufacturing Process*, Proceedings of the EFTA 2008, 13th IEEE Conference on Emerging Technologies & Factory Automation, 2008.

[18] Koseeyaporn, P., Cook, G.E., Strauss, A.M.; *Adaptive voltage control in fusion arc welding*, Industry Applications, IEEE Transactions on , vol.36, no.5, pp.1300-1307, Sep/Oct 2000.

[19] Bjorgvinsson, J.B., Cook, G.E., Andersen, K.; *Microprocessor-based arc voltage control for gas tungsten arc welding using gain scheduling*, Industry Applications, IEEE Transactions on, vol.29, no.2, pp.250-255, Mar/Apr 1993.

[20] Zhang, P. J., Li, Y. M.; *Precision Sensing of Arc Length in GTAW Based on Arc Light Spectrum,* Transactions of the ASME, p. Vol. 123, February 2001.

[21] Sciaky, C.A., Johnson, A.M.; *System For Controlling Length Of Welding, 3236997* US, 1966.

[22] Ralchenko, Yu., Kramida, A.E., Reader, J., and NIST ASD Team; NIST Atomic Spectra Database (version 3.1.5). [Online] National Institute of Standard and Technology. http://physics.nist.gov/asd3.

[23] Tham, J.; *Methods of Characterizing Gas-Metal Arc Welding Acoustics for Process Automation,* PhD Thesis, University of Waterloo (2005).

[24] Jolly, W. D.; *Acoustic emission exposes cracks during welding*, Vol. 48, Welding Journal, 1969.

[25] Drouet, M. G., Nadeau, F.; *Pressure waves due to arcing faults in a substation,* IEEE Transactions on Power Apparatus and Systems, Vol. PAS-98, 1979.

[26] Ozcelik, S., Moore, K., Naidu, D.S.; *Modeling, Sensing and Control of Gas Metal Arc Welding*, Elsevier Science, 1th Edition (2003).

[27] Tzyh-Jong, T., Shan-Ben, C., Changjiu, Z.; *Robotic Welding, Intelligence and Automation,* Lecture Notes in Control and Information Sciences (2007).

[28] Cheni, S. B., Zhang, Y., Qiu, T., Lin. T.; *Robotic Welding Systems with Vision-Sensing and Self-learning Neuron Control of Arc Welding Dynamic Process*, Journal of Intelligent and Robotic Systems, 2003.

[29] Wu, C.S., Gao, J.Q., Liu X.F., Zhao, Y.H.; *Vision–based measurement of weld pool geometry in constant–current gas tungsten arc welding*, MOE Key Lab of LSMH, Institute of Materials Joining, Shandong University, Jinan, People's Republic of China, IMechE, (2003).

[30] Cook, G.E., *Robotic Arc Welding: Research in Sensory Feedback Control,* Industrial Electronics, IEEE Transactions on, Vol. IE-20, 1983.

[31] Cook, G.E., et al.; *PC-Based Arc Ignition and Arc Length Control System for GTAW,* Industry Applications, IEEE Transactions on, Vol. 28, 1992.

[32] Amson, J.C.; *An analysis of the Gas shielded consumable electrode metal arc welding system*, British Welding Research Association, London, (1964).

[33] Hensinger, D. M., Ames, A. L., Kuhlmann, J.L.; *Motion Planning for a Direct Metal Deposition Rapid Prototyping System*, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000.

[34] Bonaccorso, F., Bruno, C., Cantelli, L., Longo, D., Muscato, G.; *A Modular Software Interface for the Control System of an Arc Welding Robot*, Proceedings of the 2nd International Conference on Human System Interaction HSI'09, pp 450-455, May 21-23 2009, Catania, Italy. ISBN 978-1-4244-3960-7.

[35] Bonaccorso, F., Bruno, C., Cantelli, L., Longo, D., Muscato, G., Rapisarda, S.; *A Closed Loop Welding Controller for a Rapid Manufacturing Process*, Proceedings of the 2009 International Conference on Trends in Aerospace Manufacturing TRAM 2009, 9-10 September 2009, Sheffield U.K.

[36] MathWorks Symbolic Math Toolbox [Online] http:// www.mathworks.it/products/symbolic/index.html.

[37] Samcef Mecano™ [Online] http://www.samtech.com/ product/product.asp?idP=91&pid=11.

[38] Robovolc project home page, [Online]. http:// www.robovolc.dees.unict.it

[39] Caltabiano, D., Muscato, G.; *A Comparison between different traction methods for a field robot*, IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2002, Lausanne Switzerland, Sep.30-Oct.4 2002.

[40] Caltabiano, D., Ciancitto, D., Muscato, G.; *Experimental Results on a Traction Control Algorithm for Mobile Robots in Volcano Environment*, Proceedings of the 2004 IEEE International Conference on Robotics & Automation, New Orleans, LA, pp. 4375-4380, April 2004.

[41] Caltabiano, D., Longo, D., Muscato, G.; *A New Traction Control Architecture for Planetary Exploration Robots*, 8th International Conference on Climbing and Walking Robots, CLAWAR 2005, London, U.K., September 13-15, 2005

[42] Muscato, G., Caltabiano, D., Guccione, S., Longo, D., Coltelli, M., Cristaldi, A., Pecora, E., Sacco, V., Sim, P., Virk, G.S., Briole, P., Semerano, A., White, T.; *ROBOVOLC: A Robot for volcano exploration – Result of first test campaign*, Industrial Robot: An International Journal, Vol. 30, N.3,pp.231-242, 2003.

[43] Caltabiano, D., Muscato, G.; *A Robotic System for Volcano Exploration*, pp. 499-519, in “Cutting Edge Robotics”, Advanced Robotic Systems Scientific Book, ISBN:3-86611-038-3, (July 2005).

[44] Caltabiano, D., Muscato, G., Russo, F.; *Localization and Self Calibration of a Robot for Volcano Exploration*, Proceedings of the 2004 IEEE International Conference on Robotics & Automation, New Orleans, LA, pp. 586-591, April 2004.

[45] Service Robotic Group, Università di Catania http://www.ro botic.diees.unict.it/.

[46] Guccione, S., Muscato, G.; *Control Strategies Computing architectures and Experimental Results of the Hybrid Robot Wheeleg*, IEEE Robotics and Automation Magazine, Vol.10, N.4, pp.33-43, December 2003.

[47] Lacagnina, M., Muscato, G., Sinatra, R.; *Kinematics, Dynamics and Control of a Hybrid Robot Wheeleg*, Journal of Robotics and Autonomous Systems, Vol 45, pp. 161-180, Elsevier Oxford, UK, 2003.

[48] Lacagnina, M., Guccione, S., Muscato, G., Sinatra, R.; *Modelling and Simulation of Multibody Mobile Robot for Volcanic Environment Explorations*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2002, Lausanne Switzerland, Sep.30-Oct.4 2002.

[49] Aranzulla, P., Bonaccorso, F., Bruno, C., Cantelli, L., Lanteri, G., Longo, D., Muscato, G., Pennisi, A., Prestifilippo, M.; *An innovative autonomous outdoor vehicle based on Microsoft Robotic Studio*, Proceedings of CLAWAR2010: 13th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, Nagoya, Japan, 31 August - 03 September 2010; p. 97, 104; ISBN-10: 9814327972; ISBN-13:978-9814327978; World Scientific Publishing Co Pte Ltd.

[50] Bonaccorso, F., Cantelli, L., Longo, D., Melita, D., Muscato, G., Prestifilippo, M.; *The U-Go Robot, a multifunction rough terrain outdoor tracked vehicle for R&D on autonomous navigation algorithms*, Proceedings of the 5th IARP Workshop on Robots for Risky Interventions and Environmental Surveillance-Maintenance, Leuven (B), 20-22 June 2011.

[51] Videredesign, User's manual STH-MDCS3-Var Stereo Head, [Online] http://www.videredesign.com.

[52] XSens Mti MT software development kit [Online] http://www.xsens.com/images/stories/Icons/pdf_icon.gif.

[53] Sick LMS200/211/221/291 Laser Measurement Sensors Datasheet, [Online] http://www.sick.com.

[54] Sick Lms2xx Serial Protocol Datasheet, [Online] http://www.sick.com.

[55] Devantech SRF08 Ultra sonic range finder Technical Specification, [Online] http://www.robot-electronics.co.uk/htm/srf08tech.shtml.

[56] Magellan Corporation Ashtech Precision Products Z-Xtreme™ Technical Reference Manual [Online] http://www.ashtech.com.

[57] Lynxmotion AL5A, [Online] http://www.lynxmotion.com/c-124-al5a.aspx.

[58] Maxon DC Motor F2260 Datasheet, [Online] www.treffer.com.br/produtos/maxon/motores/pdf/95.pdf.

[59] Maxon Encoder HEDS 5540 Datasheet, [Online] http://test.maxonmotor.com/docsx/Download/catalog_2005/Pdf/05_243_e.pdf.

[60] Altera. Getting Started with Altera's DE1 Board. [Online] ftp://ftp.altera.com/up/pub/Tutorials/DE1/Digital_Logic/tut_initialDE1.pdf.

[61] Quartus II Introduction Using Schematic design. [Online] ftp://ftp.altera.com/up/pub/Tutorials/DE2/Digital_Logic/tut_quartus_intro_schem.pdf.

[62] DE1 Development and Education Board . [Online] http://cseweb.ucsd.edu/classes/fa10/cse140L/resources/DE1_UserManual_v1017.pdf.

[63] Cyclone II Device Handbook. [Online] http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf.

[64] Configuration Handbook. [Online] http://www.altera.com/literature/hb/cfg/config_handbook.pdf.

[65] NIOS II Processor Reference, Handbook. [Online] http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf.

[66] NIOS II Flash Programmer, User Guide. [Online] http://www.altera.com/literature/ug/ug_nios2_flash_programmer.pdf

[67] Livatino, S., Muscato, G., Sessa, S., Koffel, C., Arena, C., Pennisi, A., Di Mauro, D., Malkondu, E.; *Depth-Enhanced Mobile Robot Teleguide based on Video Images*, IEEE Robotic and Automation Magazine, Special issue on New Vistas and Challenges in Telerobotics, Vol15, N.4, pp.58-67, December 2008.

[68] Livatino, S., Muscato, G., Privitera, F.; *Stereo Viewing and Virtual Reality Technologies in Mobile Robot Teleguide*, IEEE Transactions on Robotics,Vol.25, No.6, December 2009.

[69] Segway Robotic Mobility Platform User Guide [Online] http://wikiri.upc.es/images/6/6a/RMP_User_Guide_2.pdf.

[70] Segway Robotic Mobility Platform (RMP) Interface Guide Version 2.0 [Online] http://wikiri.upc.es/images/3/3c/Interface_Guide_for_Segway_RMP.pdf.

[71] Segway Robotic Mobility Platform (RMP) Interface Guide Version Page 11 errata [Online] http://rmp.segway.com.

[72] Chella, A., Pagello, E., Menegatti, E., Sorbello, R., Anzalone, S.M., Cinquegrani, F., Tonin, L., Piccione, F., Prifitis, K., Blanda, C., Buttita, E., Tranchina, E.; *A BCI Teleoperated Museum Robotic Guide*, International Conference on Complex, Intelligent and Software Intensive Systems.

[73] Nourbakhsh, I., Hamner, E., Porter, E., Dunlavey, B., Ayoob, E., Hsiu, T., Lotter, M., Shelly, S.; *The Design of a Highly Reliable Robot for Unmediated Museum Interaction*, International Conference on Robotics and Automation, Barcelona, Spain, April 2005.

[74] Few, D.A., Roman, C.M., Bruemmer, D.J., Smart, W.D.; *What Does it Do?*, HRI Studies with the General Public, 16th IEEE International Conference on Robot & Human Interactive Communication, Jeju, Korea, August 26 - 29, 2007.

[75] Kuno, Y., Sadazuka, K., Kawashima, M., Tsuruta, S., Yamazakil, K., Yamazaki, A., *Effective Head Gestures for Museum Guide Robots in Interaction with Humans*, 16th IEEE International

Conference on Robot & Human Interactive Communication, / Jeju, Korea, August 26 - 29, 2007.

[76] iRobot Roomba [Online] www.irobot.com.

[77] Lego MINDSTORMS [Online] http://mindstorms.lego.com/en-us/Default.aspx.

[78] Pioneer P3-DX [Online] http://www.mobilerobots.com/researchrobots/researchrobots/pioneerp3dx.aspx.

[79] Morgan S.; *Programming Microsoft Robotics Studio*, Microsoft Press, ISBN 0-7356-2432-1, (2008).

[80] Johns K., Taylor T.; *Professional Microsoft Robotics Developer Studio*, Wiley Publishing, Inc 2008, ISBN 978-0-470-14107-6

[81] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R.; "ROS: an open-source Robot Operating System". [Online] http://www.ros.org.

[82] Gerkey, B., Vaughan, R., and Howard, A.; *The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems*, Proceedings of the International Conference on Advanced Robotics.

[83] Blanco, J.L., Gonzalez, J., Fernández-Madrigal, J.A.; *Consistent observation grouping for generating metric-topological maps that improves robot localization*, IEEE International Conference on Robotics and Automation (ICRA), pp. 818-823, (2006).

[84] Elliott, D., Kaplan, C., Hegarty, J.; *Understanding GPS: Principles and Applications* 2nd Edition.

[85] Ben Axelrod web page [Online] http://www.benaxelrod.com/.

[86] OpenCV (Open Source Computer Vision) libraries [Online] http://opencv.willowgarage.com/wiki/.

[87] Google SketchUp Pro [Online]- http://sketchup.google.com/.

[88] Kalman, R. E.; *A New Approach to Linear Filtering and Prediction Problems*. Transaction of the ASME—Journal of Basic Engineering, 82(Series D), 35-45, 1960.

[89] Bishop G., Welch G.; *An Introduction to the Kalman Filter*, [Online] http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html.

[90] Hartikainen, J., Särkkä, S.; *EKF/UKF Toolbox for Matlab V1.3*, [Online] http://www.lce.hut.fi/research/mm/ekfukf/.

[91] Sciavicco L., Siciliano B.; Modelling and control of robot manipulators, Springer ISBN-10: 1852332212.

[92] Siciliano B., Khatib O.; *Springer Handbook of robotics*, ISBN 978-3-540-30301-5, (2008).

[93] Horn, B. K. P., *Closed-form solution of absolute orientation using unit quaternions*, In Journal of the Optical Society of America, vol.4, num.4, pp. 629-642, 1987.

[94] Moon, I.B., You, B.J., Kiol, H., Oh, S.R., *Fast Markov Localization using Angle-Histogram*, In Proc. of the IEEE International. Conference on. Advanced. Robotics (ICAR), vol.1, pp. 411-416, June 2003.

[95] Besl, P.J., McKay, N.D.; *A method for registration of 3-D shapes,* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992.

[96] Lu, F., Milios, E.; *Globally Consistent Range Scan Alignment for Environment Mapping*, In Autonomous Robots, vol. 4, pp. 333-349, 1997.

[97] Lu, F., Milios, E.; *Robot Pose Estimation in Unknown Environment by Matching 2D Range Scans*, In Proc. of IEEE CVPR, pp. 935-938, 1994.

[98] Będkowski, J., Masłowski, A.; *GPGPU implementation of On-Line point to plane 3D data registration*, Proceedings of the 2011 International Conference on Electrical Engineering and Informatics Volume 2, 17-19, Bandung, Indonesia, pp. 931-936, July 2011.

[99] Cox, I.J.; *Blanche - An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle*, IEEE Transactions on Robotics and Automation, vol. 7, no. 2, pp. 193-204, April 1991.

[100] Bonaccorso, F., Catania, F., Muscato, G.; *Evaluation of algorithms for indoor mobile robot self-localization through laser range finders data*, In Proc. 7th IFAC Symposium on Intelligent Autonomous Vehicles IAV2010, 2010.

[101] MathWorks Matlab Builder NE [Online] http://www.mathworks.com/products/netbuilder/demos.html.

[102] Kay, S.M., *Fundamentals of Statistical Signal Processing*, vol. 2. Estimation Theory, Prentice Hall, New Jersey, (1993).

[103] Nunez, P., Vazquez-Martin, R., del Toro, J.C., Bandera, A., Sandoval, F.; *Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation*, Robotics and Autonomous Systems vol. 56, pp. 247–264, 2008.

[104] Lingemann, K., Surmann, H., Nuchter, A., Hertzberg, J.; *Indoor and Outdoor Localization for Fast Mobile Robots*, In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.

[105] Bailey, T., Nebot. E.M.; *Localisation in large-scale environments*, Robotics and Autonomous Systems, vol. 37, pp. 261–281, 2001.

[106] Bailey, T., Nebot, E.M., Rosenblatt, J.K., Durrant-Whyte, H.F.; *Data Association for Mobile Robot Navigation: A Graph Theoretic Approach*, In Proc. of IEEE International Conference on Robotics & Automation, April 2000.

[107] Sammons P.J., Furukawa T., Bulgin, A.; *Autonomous Pesticide Spraying Robot for Use in a Greenhouse*, Australian Conference on Robotics and Automation (CD-ROM), Sydney, pp. 1-8, December 5-7, 2005.

[108] Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S., Bradski, G.; *Self-supervised monocular road detection in desert terrain*, In Proceedings of the Robotics Science and Systems Conference, Philadelphia, PA, 2006.

[109] Cheng, G., Gu, J., Bai, T., Majdalawieh, O.; *A new efficient control algorithm using potential field: extension to robot path tracking*, Electrical and Computer Engineering, Canadian Conference on, vol.4, no., pp. 2035- 2040 Vol.4, 2-5 May 2004.

[110] Khatib, O.; *Commande dynamique dans l'espace operationnel des robots manipulateurs en presence d'obstacles*, PhD thesis, L'Ecole Nationale Superieure de l'Aeronautique et de l'Espace, 1979.

[111] Bell, G.; *Forward Chaining for Potential Field Based Navigation*, PhD thesis School of Computer Science University of St Andrews, September 2005.

[112] Tilove, R.B.; *Local obstacle avoidance for mobile robots based on the method of artificial potentials*, IEEE Int. Conf. Robot. Autom., pp. 566—571, 1990.

[113] Nelson, D.R., Barber, D.B., McLain, T.W., Beard, R.W.; *Vector Field Path Following for Miniature Air Vehicles*, Robotics, IEEE Transactions on, vol.23, no.3, pp.519-529, June 2007.

[114] Grech, R., Fabri, S.G.; *Trajectory Tracking in the Presence of Obstacles Using the Limit Cycle Navigation Method*, Intelligent Control, Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation, 27-29 June 2005.