RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ IBN-KHALDOUN DE TIARET

FACULTÉ DES SCIENCES APPLIQUEES

DÉPARTEMENT DE GENIE MECANIQUE

FILIERE DE GENIE MECANIQUE

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du diplôme de Master

Domaine : Sciences & Technologie

Filière : Génie Mécanique

Parcours : Master

Spécialité : Productique

THÈME

Programmation et simulation d'un bras manipulateur pour la tâche de palettisation utilisant le logiciel ROBOGUIDE de FANUC

Préparé par : - Mr AMARA Mourad

Devant le Jury :

Nom et prénoms	Grade	Lieu d'exercice	Qualité
HAMMOU Mahmoud	MAA	UIK Tiaret	Président
ABOSHIGHIBA Hicham	MAA	UIK Tiaret	Examinateur
MECHEKOUR Elhadi	MCB	UIK Tiaret	Encadreur

PROMOTION 2016 / 2017

DEDICATION

I dedicate my work to my family and friends.

A special feeling of gratitude to my parents, brothers and sister whose words of encouragement and push for tenacity ring in my ears.

I also dedicate this work to my closest friends who have supported me throughout the process.

I will always appreciate all they have done.

ACKNOWLEDGMENT

In the Name of Allah, the Most Merciful, the Most Compassionate all praise be to Allah, the Lord of the worlds; and prayers and peace be upon Mohamed His servant and messenger.

First and foremost, I must acknowledge my limitless thanks to Allah, the Ever-Magnificent; the Ever-Thankful, for His help and bless. I am totally sure that this work would have never become possible, without his help.

I owe a deep debt of gratitude to my Supervisor Dr. **Mechekour Elhadi** for letting me involved in this research and appreciate his continuous support and confidence in me. He is a patient and responsible mentor, who not only taught me to learn the new approach, but also gave me enlightement suggestions and optimistic spirit while dealing with failure and disappointments. Furthermore, his encouragement and positive feedback kept me persistently motivated on my research. I am heartily grateful for his help during this year, which enabled me to complete this memoir step by step.

I would like to take this opportunity to say warm thanks to all my beloved friends, who have been so supportive along the way of doing my memoir.

I also would like to thank all of my teachers throughout my whole career, without them I would never have reached this level of knowledge.

Last but not least, I would like to express my wholehearted thanks to my family for their generous support they provided me throughout my entire life and particularly through the process of pursuing the Master degree. Because of their unconditional love and prayers, I have the chance to complete this work.

TABLE OF CONTINENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

DoF	Degree of Freedom
PUMA	Programmable Universal Manipulator for Assembly
ARLA	ABBA Robotics Language
AL	Assembly Language
VAL	Variable Assembly Language
AML	A Manufacturing Language
RCCL	Robot Control C Library
PasRo	Pascal for Robots
HAL	Hybrid Assistive Limb
EoAT	End of Arm Tooling
CAD	Computer-Aided Design

General Introduction

When we talk about robotics, so many ideas come to our minds, we think about machines that look like humans and do any task that human can't do. People may even think about things beyond imagination, however this is not a bad idea because relating the science with science fiction leads to development new technological inventions.

Nowadays, the industries exploded studies about robotic due to their big needs of their advantages in ensuring economical and practical success in automation projects, they consist in many activities that provide continuity of productions, economic power, development …etc. The robot in industries will improve not only the capacity of production but also the quality of products on regarding accuracy, high speed, difficulty and repeatability. Robots of today move in a very high speed with a calculated trajectory in protected and structured environment, without any interaction with the human operators. These robots are designed with respecting the compelled constraints of the industry, such as the repeatability, tasks to be built nicely and respecting the pattern or the way of the production…

In fact, our daily life is virtually affected by robots, the objective of robotic is to create some practical and useful robots that facilitate our daily tasks because of the independency of the robot, they have longer lifetime compared with the humans and are for sure helpful in industry. The need of robots differs by their application, especially in industries there are many applications that may require robots even if they imitate human behaviors but then apply these behaviors to the skill that leads the robots to achieve a certain task, neither do they not get tired, nor they face the commands emotionally, e.g. transporting objects, welding, assembling, painting, sorting, cutting, palletizing…etc. thus palletizing requires robot application more than any other, especially when it comes to heavy loading and repeatability. This is proved when we know that the most sold robots in the word are whose application is palletization

Since robots are designed by human, they can be programmed and expected to obey and perform the desired tasks. So, the most important role of the human here, is to be able to make

the robots do the desired tasks after building them, he has to make the program that controls it in the adequate specifics which can be projected to reality, but to facilitate the task, a simulation program will help the user to do tests before executing the real one, by that we avoid destruction of tools and objects, we avoid the danger to the human around…etc. The programming of robots is really an important subject that has to be developed and taught more than ever, especially the industries of our country have a lack of this development machines. So, I personally think that this is the right time to spread and enhance researches to develop the industries with robots, and why not to compete with other countries with the utilization of robots.

This is why I picked this subject, trying to add a value that I hope it will be applied someday, by creating a work cell with robot and program it to do the desired task, it is somehow the final phase, supposing that robots being available in worldwide market, and for this matter programming and simulating the robot and the task will be the objective of this memoir.

The memoir is axed in three chapters:

First of all, we will learn about robot and robotic in general concept, and know the different applications in which we use robots, knowing the application I chose is Palletization.

Second, we will learn about programming in robotic, the languages we use, then the choice of the robot according to the application and for this case we are going to learn about the software ROBOGUIDE.

Finally, the part comes where we use the software ROBOGUIDE to creating the work cell, the robot, and the application then programming it using simulation method (off line programming) to simulate it and see how it will work in reality.

Chapter 1

Overview on robotics and robots

1.1. Introduction

In this first chapter, we present an overview about robot and robotics that helps the reader to understand our work context, which is the industrial robotization, and the case study that we will see in the last chapter about programming task of “Pick and Place”.

1.2. Generality about robot and robotics

1.2.1. Brief historical review and main definitions of Robot

The word robot comes from the Czech *robota* which means tireless work It was first used in 1921 by the novelist Karel Capek in his novel "Rossum's Universal Robots". Capek's robots *(Figure 1.1)* are tireless working machines that looked like humans and had advanced capabilities even when compared with actual robots [1].

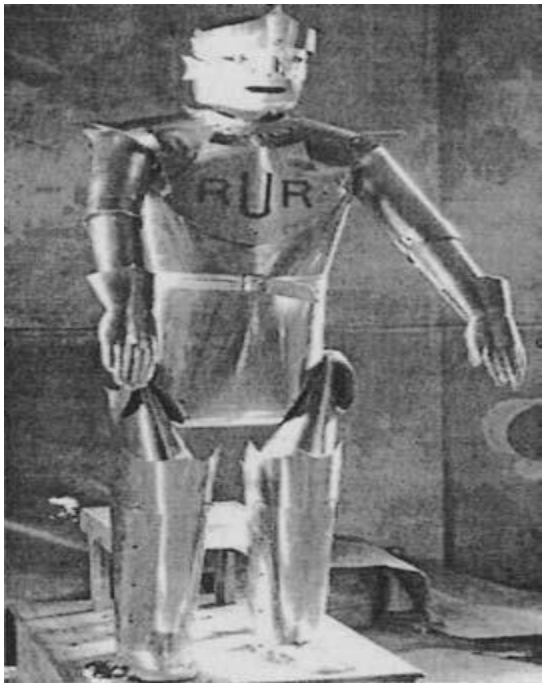


Figure 1- 1 A robot from Karel Capek's novel "Rossum 's Universal Robots"

Robotics was a special concern of the most brilliant minds of our common history, since many of them took time to imagine, design, and build machines that could mimic some human capabilities. It is one of the biggest dreams of man, to build obedient and tireless machines, capable of doing man's boring and repetitive work; an idea very well explained by Nicola Tesla in his diary [2]:

"... / conceived the idea of constructing an automaton which would mechanically represent me, and which would respond, as I do myself, but, of course, in a much more primitive manner, to external influences. Such an automaton evidently had to have motive power, organs for locomotion, directive organs, and one or more sensitive organs so adapted as to be excited by external stimuli...".

Today's challenge is to consider robots as human coworkers and companions, extending human capabilities to achieve more efficient manufacturing and to increase the quality of our lives.

Robotics can be traced back to 350 B.C., in the ancient Greece, to the fabulous philosopher and mathematician Archytas of Tarentum (428-347 B.C.) and a demonstration he made in front of the metropolis senators. A strange machine that he called "the pigeon" was capable of flying more the 200m, using some type of jet propulsion based on steam and compressed air: a great achievement for the time (the invention of the screw and also the pulley are attributed to Archytas).

Figure 1- 2 A Greek design adapted by al-Jazari for a garden hand-wash

In 270 B.C., also in ancient Greece, the civil engineer Ctecibius was capable of building water clocks with moving parts. His work had followers like Phylo of Byzantium author of the book "Mechanical Collection" (200 B.C.), and Hero of Alexandria (85 B.C.), and Marcus

Vitruvius (25 B.C.). In the twelfth century, the Arabian Badias-zaman al-Jazari (1150-1220) studied some of the Greek developments in the book "The Science of the Ingenious Devices" *(Figure 2)*. In those early times the problem was about mechanics, about how to generate and transmit motion. So, it was mainly about mechanisms, ingenious mechanical devices [3,4], but he designed humanoid robot, the first in the history.

And then there was the contribution of Nicola Tesla at the turn of the nineteenth century. He thought of using Henrich Hertz's discovery of radio waves to command an automaton. He built one to demonstrate his ideas and presented it in New York's Madison Square Garden in 1898. The problem then was that machine intelligence was missing. Robots should be able to do pre-programmed operations, and show some degree of autonomy in order to perform the desired tasks. When that became available, robots developed rapidly, and the first industrial one appeared in the early 1970s and spawned a multi-million-dollar business.

In 1974, the first electrical drive trains were available to use as actuators for robot joints. In the same year, the first microprocessor-controlled robots were also available commercially.

Around 1982, things like Cartesian interpolation for path planning were available in robot controllers, and many of them were also capable of communicating with other computer systems using serial and parallel interfaces. In the same year, some manufacturers introduced joystick control for easier programming, and the teach pendant menu interface.

In 1984, vision guidance was introduced as a general feature for tracking, parts identification, and so on.

In 1986, the first digital control loops were implemented enabling better actuator control and enabling the use of AC drives.

Networking is a feature of the 1990s, with several manufacturers implementing networking capabilities and protocols.

In 1991, there was the implementation of digital torque control loops, which enabled, for example, the utilization of full dynamical models; a feature only available in the first robots around 1994.

During the period 1992-1994 several manufacturers introduced features like Windows-based graphical interfaces, virtual robot environments for off-line programming, and fieldbuses. Robot cooperation is a feature introduced from 1995 to 1996.

Figure 1- 3 Actual robot Manipulators

Around 1998, robot manufacturers started introducing collision detection to avoid damaging robots, and load identification to optimize robot performance. Since then other features include fast pick and place, weight reduction, optimized programming languages, object-oriented programming, etc... *(Figure 1-3)* shows some of the robot manipulators available currently on the market.

So how do we define robotics then? Is it a science? Is it a technique or collection of techniques? If the reader opens a robotics book something like this appears:

> *"A robot is a re-programmable multi-functional manipulator designed to move materials, parts, tools, or specialized devices, through variable programmed motions for the performance of a variety of tasks ",* from the book Robotics - Control, Sensing, Vision and Intelligence, Fu, Gonzalez, Lee, McGraw Hill, 1987.

Although correct, despite being restricted to robot manipulators, this definition doesn't give the correct idea. The common-sense image of a robot is usually associated with strong and superb machines, tireless (like Karel Capek's machines), obedient, but nevertheless, fascinating machines that make us dream. And that fascination is not in that definition.

Our problem is that there isn’t yet a clear, efficient, and universally accepted definition of robots. If you ask ten people what the word "robot" means, nine will most likely reply that it means an automatic humanoid creature, or they will describe a device that may be more accurately denned as a manipulator or an automatic arm [5] gives the following definition: "A robot device is an instrumented mechanism used in science or industry to take the place of a human being. It may or may not physically resemble a human or perform its tasks in a human way, and the line separating robot devices from merely automated machinery is not always easy

to define. In general, the more sophisticated and individualized the machine, the more likely it is to be classed as a robot device."

Therefore, we can define robotics as a science of generic, ingenious, precise, mechatronic devices, powered by a permanent power source; a science that is open to new ideas and that stimulates the imagination. A stimulus so strong that it attracted many of the best minds of our common history, i.e., authors of the work that constitutes the legacy that we humans leave for the future.

Other definitions have been proposed in "A Glossary of Terms for Robotics":

"Robot—A mechanical device which can be programmed to perform some task of manipulation or locomotion under automatic control."

"Industrial robot— A programmable, multi-function manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks."

"Pick and place robot—A simple robot, often with only two or three degrees of freedom, which transfers items from place to place by means of point-to-point moves. Little or no trajectory control is available. Often referred to as a 'bangbang' robot."

"Manipulator—A mechanism, usually consisting of a series of segments, jointed or sliding relative to one another, for the purpose of grasping and moving objects usually in several degrees of freedom. It may be remotely controlled by a computer or by a human." [Note: The words "remotely controlled...by a human" indicate that this device is not automatic.]

"Intelligent robot—A robot which can be programmed to make performance choices contingent on sensory inputs."

"Fixed-stop robot—A robot with stop point control but no trajectory control. That is, each of its axes has a fixed limit at each end of its stroke and cannot stop except at one or the other of these limits. Such a robot with N degrees of freedom can therefore stop at no more than 2N locations (where location includes position and orientation).

Some controllers do offer the capability of program selection of one of several mechanical stops to be used. Often very good repeatability can be obtained with a fixed-stop robot."

"Android—A robot which resembles a human in physical appearance."

"Sensory-controlled robot—A robot whose program sequence can be modified as a function of information sensed from its environment.

"Open-loop robot—A robot which incorporates no feedback, i.e., no means of comparing actual output to command input of position or rate."

"Mobile robot—A robot mounted on a movable platform."

"Limited-degree-of-freedom robot—A robot able to position and orient its end effector in fewer than six degrees of freedom."

1.2.2. Feature and Characteristics

There are few essentials characteristics of robots which are required in any of the robot that are also the factors that help in deciding whether a given machine can be categorized as a robot or not.

These robot's characteristics also will help to figure out the features which need to be present in a machine before it can fall into the category of machines which are called as robots.

One of the defining characteristics of real robots is that it is a kind of machine with a potential of interacting with other physical things and it can be provided with the electronic programming to do a particular job or to do a variety of actions. The other important robotic characteristics required in a machine to be classified as a robot is the ability to recognize and understand data on physical objects, or on its local physical environment, or to process data, or to respond to different stimuli. The important point to note down here is that the following discussed characteristics of robots are in contrast to simple mechanical devices like a hydraulic press, a gear or any other appliance which does not possess any of the processing abilities and perform any tasks through purely mechanical processes and motion. The essential characteristics of robots have been classified in the following manner.

Sensing: The very basic characteristic of robots is that it must have the ability to sense its environment or the surroundings. It must sense it in almost similar ways as a human does. There are different types of sensors which can be present in a robot such as light sensors which function as eyes in human, chemical sensors to perform the functions which are done by nose, touch and pressure sensors to act like hands, hearing and sonar sensors to act like ears and taste sensors to act like tongue in human beings. All these sensors provide awareness to a robot about its surroundings.

Movement: Another of the primary robot characteristics required in machine is that it must be able to move around its surroundings. This can be done in different ways like there can be rolling on wheels propelling by thrusters a robot needs to be able to move, to count as a robot. If the entire robot moves, it is like the Sojourner or if only some just parts of the robot moves, it is like the Canada Arm.

Energy: Energy is one of the most important robot characteristics. It is necessary to have an adequate source of energy so that robot is able to power itself on its own. There may be different sources like electrically powered, battery powered or solar powered robots. The means of energy are selected in accordance with the functions which the robot needs to perform.

Intelligence: The most important among all the characteristics of robots is that it must be a kind of smarts. This is the area where programming enters into the scene. Smartness is awarded to a robot by the programmer. A programmer is a person whose job is to provide robot its smartness. The robot receives the program so that it knows what to do and how to do.

1.2.3. Types of Robots

There are many ways how we could possibly define different types of robots. As we have seen the possible divisions vary widely. The main reason of these differences is that different tutors often tend to have different views on what should be taught under "robotics".

There are two possible ways how this could be done. First, we could divide robots into types by their application and second - by the way they move (or don't). It's acknowledged that there are other possible ways how to divide robots into types but let us start with the most common way, which shows and answers one of the most important questions about robotic, what the robot does? Then it comes; how it does it?

1.2.3.1. Types of Robots by Applications

Nowadays, robots do a lot of different tasks in many fields and the number of jobs entrusted to robots is growing steadily. That's why one of the best ways how to divide robots into types is a division by their application. There are:

Industrial robots: Industrial robots are robots used in an industrial manufacturing environment. Usually these are articulated arms specifically developed for such applications as welding, material handling, painting and others. If we judge purely by application this type could also include some automated guided vehicles and other robots

Domestic or household robots: Robots used at home. This type of robots includes many quite different devices such as robotic vacuum cleaners, robotic pool cleaners, sweepers, gutter cleaners and other robots that can do different chores. Also, some surveillance and telepresence robots could be regarded as household robots if used in that environment.

Medical robots: Robots used in medicine and medical institutions. First and foremost - surgery robots. Also, some automated guided vehicles and maybe lifting aides.

Service robots: Robots that don't fall into other types by usage. These could be different data gathering robots, robots made to show off technologies, robots used for research, etc.

Military robots: Robots used in military. This type of robots includes bomb disposal robots, different transportation robots, reconnaissance drones. Often robots initially created for military purposes can be used in law enforcement, search and rescue and other related fields.

Entertainment robots: These are robots used for entertainment. This is a very broad category. It starts with toy robots such as Robosapiens or the running alarm clock and ends with real heavyweights such as articulated robot arms used as motion simulators.

Space robots: I'd like to single out robots used in space as a separate type. This type would include robots used on the International Space Station, *Canadarm* that was used in Shuttles, as well as Mars rovers and other robots used in space.

Hobby and competition robots: Robots that you create. Line followers, sumo-bots, robots made just for fun and robots made for competition.

1.2.3.2. Types of robots by locomotion and kinematics

As we see that the classification by application is not totally sufficient when we talk about a specific robot, that is why we suggest to discuss the main different types of industrial robots:

- ***Cartesian Robot***

A Cartesian coordinate robot (also called linear robot) is an industrial robot whose three principal axis of control are linear as it is shown in (*Figure 1-4)* (i.e. they move in a straight line rather than rotate) and are at right angles to each other. The three sliding joints correspond to moving the wrist up-down, in-out, back-forth. Among other advantages, this mechanical arrangement simplifies the Robot control arm solution. Cartesian coordinate robots with the horizontal member supported at both ends are sometimes called Gantry robots; mechanically, they resemble gantry cranes, although the latter are not generally robots. Gantry robots are often quite large.

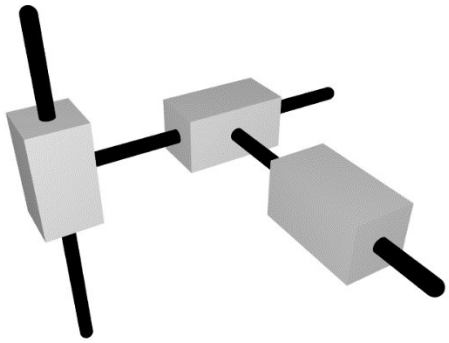


Figure 1- 4 Kinematic diagram of Cartesian (coordinate) robot

- **Cylindrical Robot**

Used for assembly operations, handling at machine tools, spot welding, and handling at diecasting machines. It's a robot whose axes form a cylindrical coordinate system.

When the arm of the robot has a revolute and two prismatic joints, it can operate in z-axis and each point that can be reached by this robot can be represented by the cylindrical coordinates. As shown in (*Figure 1-5)*, the robot can move in and out in z direction, can elevate in y direction and can rotate in θ direction. The arm can move in directions between the specific upper and lower boundaries. [14]

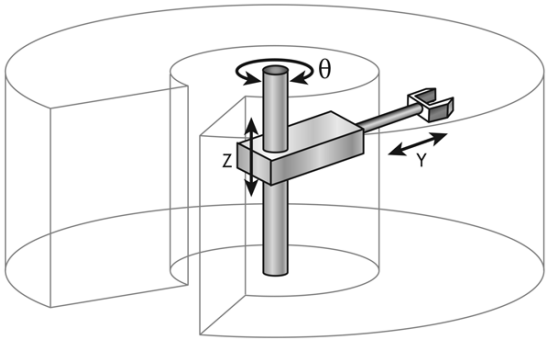


Figure 1- 5 Cylindrical Robot Arm Configuration

- **Spherical Robot**

Or polar robot, it is a robot with two rotary joints and one prismatic joint; in other words, two rotary axes and one linear axis. Spherical robots have an arm which forms a spherical coordinate system, see (*Figure 1-6)* it is possible to have additional joints for the end-effector so that the horizontal arm can extend. An example of this type of robot is the Sphero. Used for handling machine tools, spot welding, diecasting, fettling machines, gas welding and arc welding.

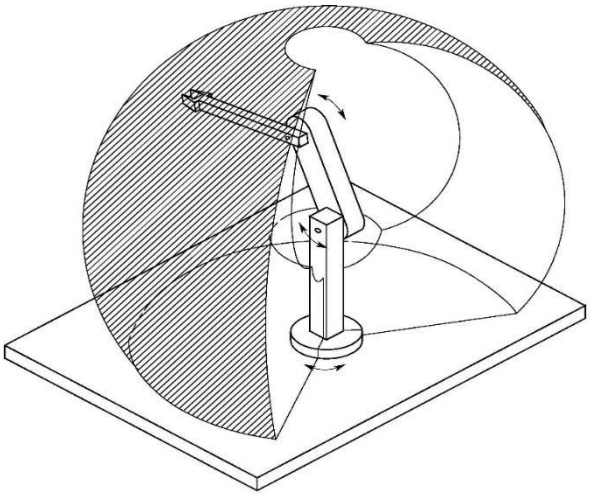


Figure 1- 6 Spherical Robot Arm Configuration

- **Scara**

The robot was called Selective Compliance Assembly Robot Arm, SCARA. It consists of two parallel rotary joints and a prismatic joint as it is shown in (*Figure 1-7)*. Its arm was rigid in the Z-axis and pliable in the XY-axes, which allowed it to adapt to holes in the XY-axes. By virtue of the SCARA's parallel-axis joint layout, the arm is slightly compliant in the X-Y direction but rigid in the 'Z' direction, hence the term: Selective Compliant. This is advantageous for many types of assembly operations, i.e., inserting a round pin in a round hole without binding.

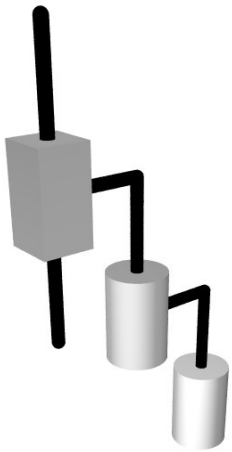


Figure 1- 7 Kinematic diagram of SCARA configuration

The second attribute of the SCARA is the jointed two-link arm layout similar to our human arms, hence the often-used term, Articulated. This feature allows the arm to extend into confined areas and then retract or "fold up" out of the way. This is advantageous for transferring parts from

one cell to another or for loading/unloading process stations that are enclosed. Used for pick and place work, application of sealant, assembly operations and handling machine tools.

- **Articulated Robot**

It is a robot with rotary joints (e.g. a legged robot or an industrial robot). Articulated robots can range from simple two-jointed structures to systems with 10 or more interacting joints (see *Figure 1-8*) They are powered by a variety of means, including electric motors. Used for assembly operations, diecasting, fettling machines, gas welding, arc welding and spray painting.

Almost 80% of the registered robots are articulated and up to 20% are linear robots.

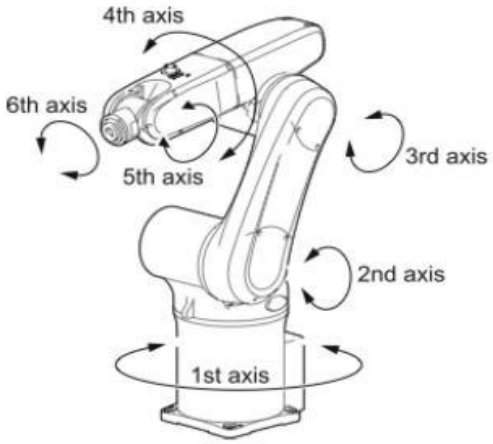


Figure 1- 8 Articulated Robot Arm Configuration

- **Parallel Robot**

Also known generalized Stewart platforms, these systems are articulated robots that use similar mechanisms for the movement of either the robot on its base, or one or more manipulator arms. Their 'parallel' distinction, as opposed to a serial manipulator, is that the end effector (or 'hand') of this linkage (or 'arm') is connected to its base by a number of (usually three or six) separate and independent linkages working in parallel. 'Parallel' is used here in the computer science sense, rather than the geometrical; these linkages act together, but it is not implied that they are aligned as parallel lines; here parallel means that the position of the end point of each linkage is independent of the position of the other linkages. One use is a mobile platform handling cockpit flight simulators.

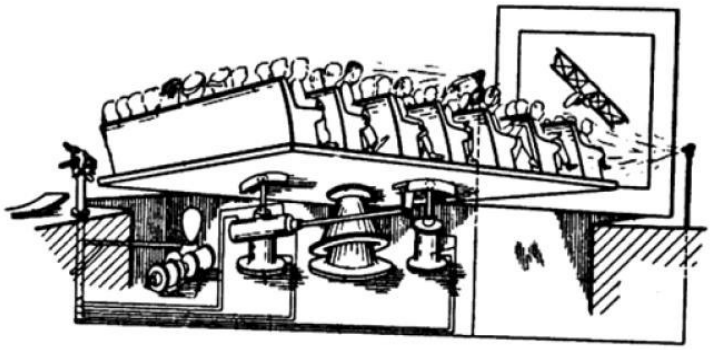


Figure 1- 9 Potentially the first spatial parallel mechanism

- **Anthropomorphic Robot**

It is shaped in a way that resembles a human hand, i.e. with independent fingers and thumbs.

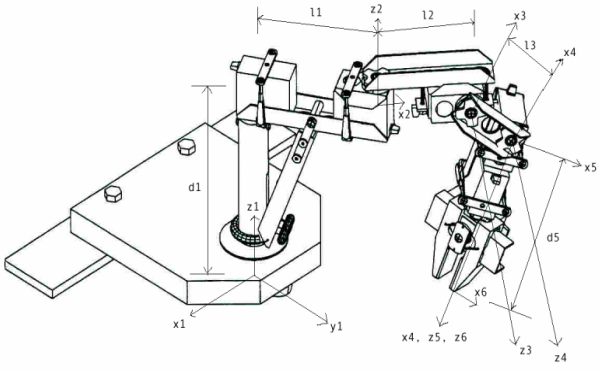


Figure 1- 10 Anthropomorphic Robot Arm Configuration

1.2.3.3. Comparison of fundamental robot arms

Configuration	*Advantages*	*Disadvantages*
Cartesian **(Three linear axes)**	• Easy to visualize • Rigid structure • Easy off-line programming • Easy mechanical stops	• Reach only to front and back • Requires a large floor space • Axes are hard to seal • Expensive
Cylindrical **(1 rotating and two linear axes)**	• Can reach all around • Rigid y,z-axes	• Cannot reach above itself • y-z-axes hard to seal • Won't reach around obstacles • Horizontal motion is circular
Spherical **(2 rotating and one linear axes)**	• Can reach all around • Can reach above or below obstacles • Large work volume	• Cannot reach above itself • Short vertical reach
Articulated **(3 rotating axes)**	• Can reach above or below obstacles • Largest work area for least floor space	• Difficult to program off-line • Two of more ways to reach a point • Most complex robot

Table 1 Comparison of fundamental robot arms

1.2.3.4. Defining parameters

- **Number of axes:** two axes are required to reach any point in a plane; three axes are required to reach any point in space. To fully control the orientation of the end of the arm (i.e. the wrist) three more axes (yaw, pitch, and roll) are required. Some designs (e.g. the SCARA robot) trade limitations in motion possibilities for cost, speed, and accuracy.
- **Degrees of freedom:** this is usually the same as the number of axes limits.
- **Working envelope:** the region of space a robot can reach.
- **Kinematics:** the actual arrangement of rigid members and joints in the robot, which determines the robot's possible motions. Classes of robot kinematics include articulated, Cartesian, parallel and SCARA.
- **Carrying capacity or payload:** how much weight a robot can lift.
- **Speed:** how fast the robot can position the end of its arm. This may be defined in terms of the angular or linear speed of each axis or as a compound speed i.e. the speed of the end of the arm when all axes are moving.
- **Acceleration:** how quickly an axis can accelerate. Since this is a limiting factor a robot may not be able to reach its specified maximum speed for movements over a short distance or a complex path requiring frequent changes of direction.
- **Accuracy:** how closely a robot can reach a commanded position. When the absolute position of the robot is measured and compared to the commanded position the error is a measure of accuracy. Accuracy can be improved with external sensing for example a vision system or Infra-Red. See robot calibration. Accuracy can vary with speed and position within the working envelope and with payload (see compliance).
- **Repeatability:** how well the robot will return to a programmed position. This is not the same as accuracy. It may be that when told to go to a certain X-Y-Z position that it gets only to within 1 mm of that position. This would be its accuracy which may be improved by calibration. But if that position is taught into controller memory and each time it is sent there it returns to within 0.1mm of the taught position then the repeatability will be within 0.1mm.

1.3. Robot Kinematics

From a mechanical point of view, a robot is an assembly of a set of links where any pair of two adjacent links is connected by a one-DOF (Degree of Freedom) joint. Every joint inside a robot's mechanism can be either a prismatic or revolute joint which imposes one degree of

freedom and can be described by one input motion parameter. Since the purpose of a robot's mechanism is to shape the output motion at a particular point on the mechanism itself, it is important to understand the relationship between the robot mechanism's input motion parameters and its output motion parameters [6].

It is worth noting at this point, that the robot mechanism's output motion is specified and described with respect to a Cartesian space which is commonly called task space in robotics. And a three-dimensional Cartesian space can be represented by the orthogonal coordinate system O-XYZ, where a general motion involves six parameters:

- three translations;
- three rotations.

As for the robot mechanism's input motions of a robot's mechanism, a joint imposes one kinematic constraint on the corresponding kinematic-pair. The motion parameter of a one-DOF joint is normally represented by a variable which describes the relative position between the two links in the kinematic pair. The motion parameter of joint i is commonly called a *joint variable* and is denoted by q_i. The joint variable q_i refers to an angle if joint i is a revolute joint. Otherwise, it refers to a linear or circular displacement. All the joint variables in a mechanism form a vector of joint coordinates which define a space commonly called a *joint space*.

Therefore, the robot mechanism's kinematic analysis is simply the study of the mapping between task space and joint space. (see *Figure 1-11*). Mapping from joint space to task space is called forward kinematics, while the reverse mapping (*from task space to joint space*) is called inverse kinematics.

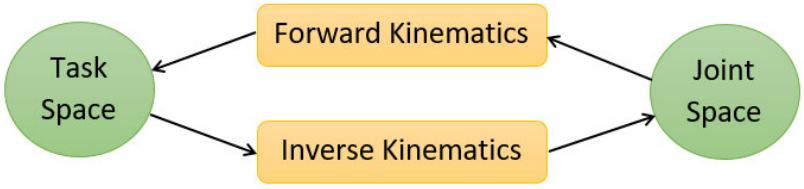


Figure 1- 11 An illustration of kinematic mapping

1.3.1. Kinematics of a link

It is a common practice in science and engineering to develop a systematic approach to describe any complex system. As for kinematic analysis, it is desirable to have a systematic method.

Since a robot's mechanism can be treated as the combination of a set of open kinematic-chains, it is important to develop a theoretical framework for the modelling and analysis of an open kinematic-chain. An open kinematic-chain consists of a set of serially connected links. Thus, the starting point for kinematic analysis is to study the kinematics of a link. [6]

Each joint has one degree of freedom, either translational (a sliding or prismatic joint) or rotational (a revolute joint) and the motion of the joint changes the relative angle or position of its neighboring links.

The definition of mechanisms by means of 4 quantities is a convention usually called the Denavit-Hartenberg notation. It is a systematic way of describing the geometry of a serial chain of links and joints. It was proposed by Denavit and Hartenberg in 1955

According to the Denavit-Hartenberg notation any robot can be described kinematically by giving the values of four quantities for each link; two to describe the link itself, and two to describe the link's connection to a neighboring link. In the usual case of a revolute joint, θ is called the joint variable, and the other three quantities would be fixed link parameters. For prismatic joints, d is the joint variable, and the other three quantities are fixed link parameters. [7]

Joint angle	θ_j	The angle between the x_{j-1} and x_j axes about the z_{j-1} axis	Revolute joint variable
Link offset	d_j	The distance from the origin of frame j-1 to the x_j axis along the z_{j-1} axis	Prismatic joint variable
Link length	a_j	The distance between the z_{j-1} and z_j axis along the x_j axis; for intersecting axes in parallel to $\hat{z}_{j-1} \times \hat{z}_{j-1}$	Constant
Link twist	α_j	The angle from z_{j-1} axis to the z_j axis about the x_j axis	Constant

Table 2 Denavit-Hartenberg Parameters

Given that the Denavit-Hartenberg convention uses exactly four parameters to describe the transformation between link frames, one might naturally wonder if the number of parameters can be reduced even further, by an even more clever set of link frame assignment rules. Denavit and Hartenberg show that this is not possible, and that four is the minimum number of parameters [10].

1.3.2. Forward Kinematics

The forward kinematics of a robot refers to the calculation of the position and orientation of its end-effector frame from its joint values. (*Figure 1-12*) illustrates the forward kinematics problem for the $3R$ planar open chain. Starting from the base link, the link lengths are $L1, L2,$ and $L3$. Choose a fixed frame $\{0\}$ with origin located at the base joint as shown, and assume an end-effector frame $\{4\}$ has been attached to the tip of the third link. The Cartesian position (x, y) and orientation φ of the end-effector frame as a function of the joint angles $(\theta1, \theta2, \theta3)$ are then given by

$$x = L1\cos\theta1 + L2\cos(\theta1 + \theta2) + L3\cos(\theta1 + \theta2 + \theta3) \quad (2.1)$$
$$y = L1\sin\theta1 + L2\sin(\theta1 + \theta2) + L3\sin(\theta1 + \theta2 + \theta3) \quad (2.2)$$
$$\varphi = \theta1 + \theta2 + \theta3. \quad (2.3)$$

If one is only interested in the (x, y) position of the end-effector, the robot's task space is then taken to be the $x - y$ plane, and the forward kinematics would then consist of Equations $(2.1) - (2.2)$ only. If the end-effector's position and orientation both matter, the forward kinematics would then consist of the three equations $(2.1) - (2.3)$.[11]

While the above analysis can be done using only basic trigonometry, it is not difficult to imagine that for more general spatial chains, the analysis can become considerably more complicated. A more systematic method of deriving the forward kinematics would be to first attach reference frames to each of the links; in *Figure 1-13* the three link reference frames are respectively labelled $\{1\}, \{2\}$, and $\{3\}$. The forward kinematics can then be written as a product of four $SE(2)$ matrices:

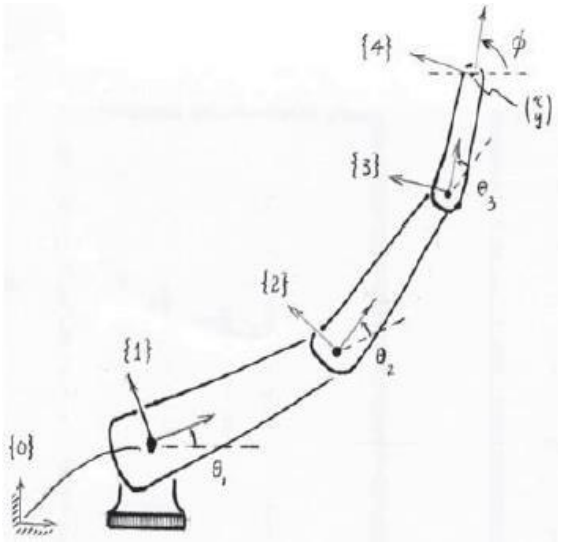


Figure 1- 12 Forward kinematics of a 3R planar open chain

1.3.3. Denavit-Hartenberg

The basic idea underlying the Denavit-Hartenberg approach to forward kinematics is to attach reference frames to each link of the open chain, and to derive the forward kinematics based on knowledge of the relative displacements between adjacent link frames. Assume that a fixed reference frame has been established, and that a reference frame (the end-effector frame) has been attached to some point on the last link of the open chain. For a chain consisting of n one degree of freedom joints, the links are numbered sequentially from 0 to n, in which the ground link is labelled 0, and the end-effector frame is attached to some point on link n. Reference frames attached to the links are also correspondingly labelled from $\{0\}$ (the fixed frame) to $\{n\}$ (the end-effector frame). The joint variable corresponding to the i-th joint is denoted θ_i.[11]

The forward kinematics of the n-link open chain can then be expressed as

$$\boldsymbol{T_0^n(\theta_1\theta_2 \dots \dots \theta_n) = T_0^1(\theta_1)T_1^2(\theta_2) \dots T_{n-1}^n(\theta_n)}$$

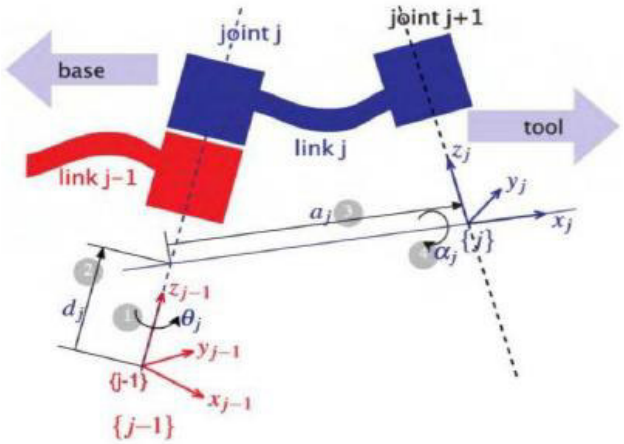


Figure 1- 13 Illustration of Denavit-Hartenberg parameters

1.4. Robot manufacturing applications

1.4.1. Product Design for Robot Automation

Identifying automation opportunities early in product design is important because product design requirements to facilitate robotic manufacturing are often unique and must be integrated early in the product design process. Some overall manufacturing problems for using robots and some design solutions to resolve these problems are listed in (*Table 3*) [12]

Problems in Utilizing Robotics	Design Solutions to Assist Production
Location accuracy and repeatability	Design for vertical assembly; use chamfered edges for mating surfaces; tolerance leeway for mating parts
Part feeding and orientation	Design parts which can be easily fed, provide notches, guide pins, or slots for part orientation; select parts from vendors that will deliver in easy-to-feed packaging
Programming robot and associated equipment	Design simplification; use common parts for different products, part reductions; part families
Application problems with fasteners (screws, washers, and nuts)	Minimize the use of all fasteners; utilize snap fits where possible
Application problems with fasteners (screws, washers, and nuts)	Select vendors that produce high-quality parts

Table 3 Design Solutions for Robots

However, replacing humans with robots to perform processes has often led to failure. The reason is that the robots are often mechanically capable of the manipulation while being incapable of process planning and control. Thousands of robot installations have failed because replacing the manual method with the automatic method lacked adaptability to process related variation. The human operators had been using their cognitive abilities to do the job. A vast majority of successful robot implementations past and present have a very important common aspect: repeated execution of fixed programs with little or no on-line modification of path or position. [13]

Process robot planning and programming still usually require the efforts of highly skilled technicians. Often, complex programs take too long. Continuously controlling and varying path manipulation parameters for real-time process control is difficult.

Also, robotic material handling and process applications are presented from an automation system perspective focusing on the robot's manipulation functions. Programming and control are viewed as the means of integrating robot manipulation as part of the manufacturing process.

1.4.2. Industrial Applications of Process Robots

1.4.2.1. Palletizing and Depalletizing

Many products are packaged in boxes of regular shape and stacked on standard pallets for shipping. Robots are commonly used to palletize and depalletize boxes because they can be programmed to move through the array of box positions layer after layer. Although palletizing is more common than depalletizing, there is no major functional difference in the manipulation requirements. Transport distances of several feet are common. Stack heights usually do not exceed 5 ft. (1.5 m) Payload weight can be in excess of 100 lb. (45 kg) When standard servo-driven joint actuators are used accuracy and repeatability will usually be far better than the required box positioning precision [13].

Palletizing typically requires four axes of controlled motion — three for translation and a fourth for yaw to orient the box. Cylindrical coordinate robots are favored in palletizing because they have large vertical lift and a compact footprint allowing more of the floor area in the workspace for conveyors and pallets. When larger workspace is needed gantry, robots must be used. Continuous duty cycles are not uncommon and robot power is important for maximizing throughput. The most technically demanding aspect of system design is the gripper. Vacuum grippers are popular for lifting boxes by their tops, but other more complex gripping methods are sometimes needed. Payloads must be carefully positioned with respect to the robot's wrist and other links to balance gravitational and dynamic loading. Load shifting during high acceleration moves can result in dropping or misallocating the box.

Palletizing position arrays are usually taught or programmed relative to a corner or keystone box position as a reference so that the entire array can be shifted by redefining that one position. Programs are simple and easily modified to adapt to changes in box dimensions. Monitoring is done by checking the state of discrete proximity and vacuum sensors. A proximity sensor mounted on a gripper will indicate if an object is at an expected location; or the same simple proximity sensor may be used to stop the robot in the correct location to pick up a box from a stack of unknown height when the top of the box is encountered. Vacuum pressure

switches are often used to verify acquisition by suction cup. A simple proximity switch can be used to signal the presence of an expected package at the pick-up point. With careful timing and additional sensor inputs, items can be transported to and from moving conveyors.

1.4.2.2. Packaging

Packaging is often a combination of palletizing and assembly-type actions. A collection of objects which may not be identical are inserted into a box or other container. The robot may also be required to assemble, place dunnage, seal, or mark the package. Insertion may simply require positioning the pack item over the opening of the package and dropping it. Boxes most often are supplied partially assembled, printed, and folded flat. Usually human operators or a special machine will open and prepare the box for packing; rarely will the robot be used for this purpose. Often the robot can be used to place cardboard layer separators, foam, or cardboard holding forms and other protective dunnage in the box. Finally, sealing and marking operations may be performed by the robot. Pack items may require complicated assembly-type motions such as rotations and curved moves to clear other packed items [13].

Three to six axes of motion may be needed. Packing items with a variety of sizes, shapes, and other varying physical properties into one package have the potential to complicate motion and tooling requirements. Grippers can be designed with multiple functions or they can be designed to be exchanged by the robot at tool storage racks. When material throughput is high, a single robot may be dedicated to each pack item. Simple programming methods are employed such as teach programming.

1.4.2.3. Sorting

Discrete parts are often sorted during production, usually as a condition of transfer to the next production station. The sort characteristics are usually distributed in some unpredictable manner so that individual inspection and handling are required. The difference between sorting and other transfer or loading robot applications is that the disposition of the part is based on information gained during the sort. The robot must have the programming functions to support multiple preprogrammed path execution triggered by the logical sort outcome conditions.

1.4.2.4. Part Dipping

Many processes require controlled manipulation of parts temporarily submerged in some working fluid or coating material. Some common part dipping processes are; Investment casting, Solder pretinning, conformal protective coating and quenching.

Dipping processes require precision of part insertion and withdrawal so velocity and acceleration must be programmable and repeatable. The stirring requirements may require the use of a two- or three-axis wrist in addition to the translation motion axes. Grippers may require special cleaning or cooling capability either on board or at service stations located conveniently in the robot's reach [13].

1.4.2.5. Resistance Spot Welding

Robotic spot welding is the most pervasive robot application in the automotive industry. Resistance spot welds are formed by tightly clamping steel pieces together with opposing contact electrodes and then passing a large amount of current through the joint, welding the metal while producing a spray of molten sparks along with loud noise. Then the joint is held momentarily until the weld solidifies. Welds are made at discrete positions by moving the robot-mounted gun to pretaught poses. The spot welding process parameters, pressure and temperature, are controlled with the separate gun controller. Weld location and therefore positioning of the gun are critical.

Dexterity, payload, and quickness are critical operational requirements for spot welding robots. Gun pose repeatability is critical for consistently locating weld joints. Access to joint locations is limited because both electrodes must reach the weld site while maintaining clearance between gun frame and workpiece edges [13].

1.4.2.6. Drilling

Hole drilling is a precision machining process. Most robots cannot hold a drill spindle rigidly enough to overcome the drilling reactions and most robots cannot generally move in a precise enough straight line to feed the drill. Drilling robots use special drilling end effectors which locate and dock onto the work piece or a fixture. The robot wrist and arm must be compliant and forceful enough to hold the drilling end effector firmly into location against the fixture or workpiece. Drilling end effectors have a spindle motor and a feed mechanism which execute a separately controlled drilling cycle while the robot holds the end effector in position.

The robot's only contribution to the process is to move the drilling end effector into its docking or holding place. Drilling robots have been used most successfully in the aerospace

industry because airframe structures require thousands of holes to be placed precisely and in complex orientations. Manipulability requirements for drilling are similar to those for spot welding. The drilling end effector weight will tend to be less than a welding gun but tool holding force and reach usually impose the requirement for large robots.

1.4.2.7. Paint and Compound Spraying

Paint spraying is a major application in the automotive industry. Painting booths are hazardous because the paint material is often toxic, carcinogenic, and flammable. Human painters often wear required protective clothing and breathing equipment. The paint fan projecting from the arm-mounted spray gun must be manipulated smoothly along paths that are often curved and complex [13].

Most robots used for painting are especially designed for that purpose. They usually have large reach, small payloads, and repeatability is usually larger than that of other types of robots and may exceed ± 0.01 in. (0.2 mm) Painting robots are typically six DOF articulated arms, often with supplemental axes to pitch the paint gun and to traverse alongside a moving line.

1.4.2.8. Cutting

Many engineering materials are produced and supplied as stacked or rolled flat plates or sheets. Further fabrication can require forming and/or cutting these materials into precise shapes. Robots are frequently used to manipulate a variety of cutting tools along paths that are often complex and curved. Many cutting processes are also used to produce fine features such as holes and slots. Common robotically manipulated cutting processes are; Laser, water-jet, abrasive-jet, plasma arc, router and knife.

The cutting tool path and pose must be precisely controlled to achieve accurately patterned piece parts. The demands on manipulator performance are primarily determined by the interaction of desired part geometry, material thickness, and material properties. Feed rate, tool stand-off, beam, jet, and arc angle are all cutting process control variables which must be adjusted to material characteristics for good results.

Still there are more applications or applications derived from those mentioned previously to be considered as important as the first ones like machine tending (loading and unloading), fastening, inspection, arc welding, finish machining…etc.

Chapter 2

Robot Programming

2.1. Introduction

We have seen in the previous chapter an overview on robots and robotics, more particularly the robot arm that is the main subject of the memoir. Now we have to continue with the most important about it, the programming of a Fanuc Robot, using the software ROBOGUIDE.

2.2. Work objective

The work consists in realizing the programming of a manipulator arm which performs a task of "palletizing" according to different configurations, according to the trajectory of the arm and the obstacles in its environment. The manipulator arm used in this study is the FANUC "M410iB/140H", which will be described in detail in this chapter and the software used is "ROBOGUIDE", the tool dedicated to programming the manipulator arms of the specialized company In the FANUC industrial robots.

I have chosen to take this topic, as a result of the lack of practical training in industrial robotics in our department which I hope this is going to be taken in consideration and help in the future practical work. So, I made it as a didactic tutorial to facilitate to anyone who wants to set up such training about it.

2.3. Work approach

In order to achieve our objective mentioned previously, we chose to use the ROBOGUIDE software, which allows us to program the manipulating arms of the FANUC company. This dedicated software integrates several functionalities facilitating the task of programming robots of the same company.

In my approach, I propose a training-oriented environment that can be easily modified in an industrial environment if it is required.

To facilitate our didactic approach, I summarize the steps of programming a manipulator arm as follows:

- Creation of the "working cell" or the working environment;
- Choice of the manipulator arm to be used;
- Choice of tool or gripper, in my case it is a Vacuum;
- Creation of the manipulated parts or the import of their 3D model;
- Application of the “Palletization” task using the teach tool/simulation program

- Simulation of the task for verification.

Before continuing this work, I want to emphasize the fact that each constructor tries to impose its own programming language of manipulator arms. Which is rarely possible to find standardized languages that work with all types of arms, which is why I will introduce you to some types of programming in robotics as well as programming languages in robotics.

2.4. Robot programming

As we know, the programming of the robot comes as the last procedure, the robot is manipulated by a series of commands and functions stored in file a so-called "robot program". The path of the robot must be programmed before execution. There is more than one way to program a robot, and in this passage, I am about to mention the most three famous methods of programming a robot. Robot programming has largely moved away from low level coding to more intuitive methods, this move has partly been fueled by a desire to make programming easier for operators.

2.4.1. Teaching Pendant

The most popular method of robot programming is probably the teach pendant. Over 90% of robots are programmed using this method. The robot teaching pendant has changed a lot throughout its lifetime, but often consists of, what looks like, a giant handheld calculator as it is shown in *(Figure 2-1).* Early pendants were large, grey boxes with magnetic tape storage. The modern teach pendants are more like a touchscreen tablet, as the technology has developed to suit the ever-evolving users. To program the robot, the operator moves it from point-to-point, using the buttons on the pendant to move it around and save each position individually. When the whole program has been learned, the robot can play back the points at full speed.

Figure 2- 1 Teach Pendant device

➢ **Advantages of Teaching Pendant**

- Most traditional industrial robots come with a teach pendant, which makes them familiar to technicians. Such as Fanuc Robots.
- They allow precise positioning, as the robot can be programmed using numerical coordinates, in either world coordinates, robot coordinates or another coordinate system.
- Teach pendants are great for simple movements, such as painting in a straight line or over a large flat surface.

➢ **Disadvantages of teaching pendant**

- Disruptive to the whole system due to robot downtime. The robot must be put into "teach mode" and all operations using the robot halted until it has been programmed.
- Requires training to learn and program.
- Might be difficult for skilled craftspeople who are unfamiliar with programming.

2.4.2. Simulation / Offline Programming

Offline programming, or simulation, is most often used in robotics research to ensure that advanced control algorithms are operating correctly before moving them onto a real robot. However, it is also used in industry to reduce downtime and improve efficiency. Programming offline means that this does not interfere with production too much. Offline programming allows the robot to be programmed using a virtual mockup of the robot and task. If the simulation software is intuitive to use, this can be a quick way to test an idea before moving it to the robot. Some simulators also allow to enter a CAD part and the system will automatically generate the robot trajectories. This can improve the efficiency of programming even further. *(see Figure 2-2)*

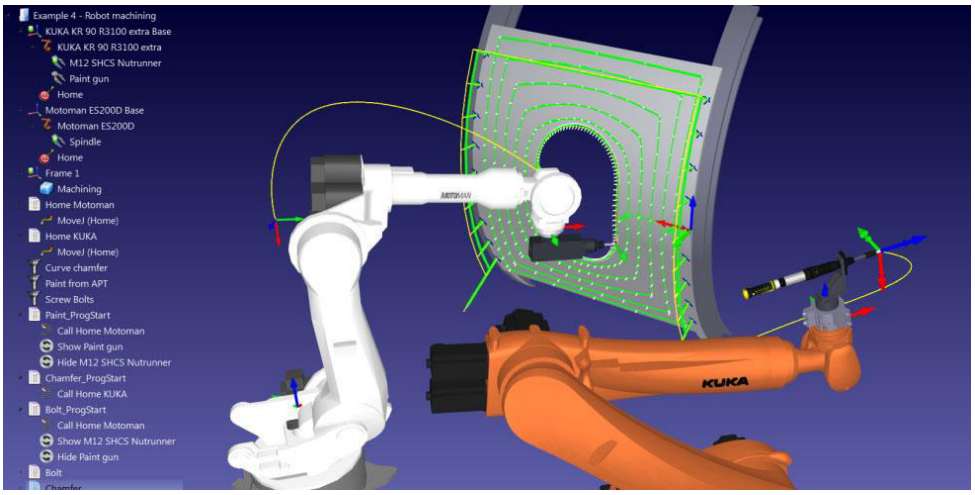


Figure 2- 2 Simulation Programming of Kuka robot

➢ **Advantages of Offline Programming**

- Reduces downtime required for robot programming. Programs are developed offline, so the robot only has to be halted while the new program is being downloaded and tested.
- Can be quite intuitive, especially if the robot can be moved around in a 3D CAD environment with drag and drop techniques.
- Easy to test many different approaches to the same problem, which would be inefficient for online programming methods.

➢ **Disadvantages of Offline Programming**

- Virtual models will (probably) never be able to represent the real world with 100% accuracy. Programs may still need to be altered after they are applied to the real robot.
- Might take longer overall. Although offline programming reduces the downtime of the robot, it means that someone has to spend extra time developing the simulation, as well as testing it on the robot.
- Can sometimes end up wasting time sorting out simulator issues instead of solving production challenges. This could be related to the quality of the simulator.

2.4.3. Teaching by Demonstration

Teaching by demonstration (and more specific methods like Kinetic teaching) offers an intuitive addition to the classic teach pendant. These methods involve moving the robot around, either by manipulation a force sensor or a joystick attached to the robot wrist just above the end effector. As with the teach pendant, the operator stores each position in the robot computer. Many collaborative robots have incorporated this programming method into their robots, as it is easy for operators to get started immediately using the robot with their applications. *(See Figure 2-3)*

Figure 2- 3 Teaching by Demonstration

➢ **Advantages of Teaching by Demonstration**

- Quicker than traditional teach pendants. It removes the need for multiple button pressing, allowing the operator to simply move the robot to the desired position.
- More intuitive than both traditional teach pendants and simulation programs, as the task is programmed in almost the same way a human operator would perform it. This makes it simple for operators to learn. Generally, this method requires no knowledge of programming concepts or being familiar with 3D CAD environments (as simulation does).
- Very good for detailed tasks which would require many lines of code to achieve the same effect, such as welding or painting of intricate shapes.

➢ **Disadvantages of Teaching by Demonstration**

- As with traditional a teach pendant, this method uses the physical robot for programming. This means that it does not reduce downtime, as much as offline programming.
- Moving the robot to precise coordinates is not as straightforward as with the other methods. This is especially true with some joystick based systems, where there is no way of entering a numerical value. Kinetic teaching combines these features by allowing for the entering of exact numerical coordinates along with positioning based coordinates.
- Not so good for tasks which are "algorithmic" in nature. For example, if a robot had to paint a flat surface by moving horizontally along the surface, then move down an inch, move horizontally in the opposite direction, etc. Moving the robot by hand would be arduous and inaccurate for such a task.

2.5. Languages of Robot programming

An industrial robot as such is basically not dedicated for a particular task or application (even if some types of robots are preferably used in certain applications). That distinguishes robots from other types of machinery. Originally, however, the individual joints of a robot were commanded and controlled as for any other multi-axes servo-controlled machine. This means that motions were specified numerically by sequences of simple motion commands. Interpretation of these commands results in calls to move-procedures provided by an interface to the servo control algorithm, which controls the physical system via sensors and actuators, as shown in (*Figure 2-4*). This means that rudimentary robot control is similar to standard servo control, and is easily incorporated into any of today's programmable control systems.

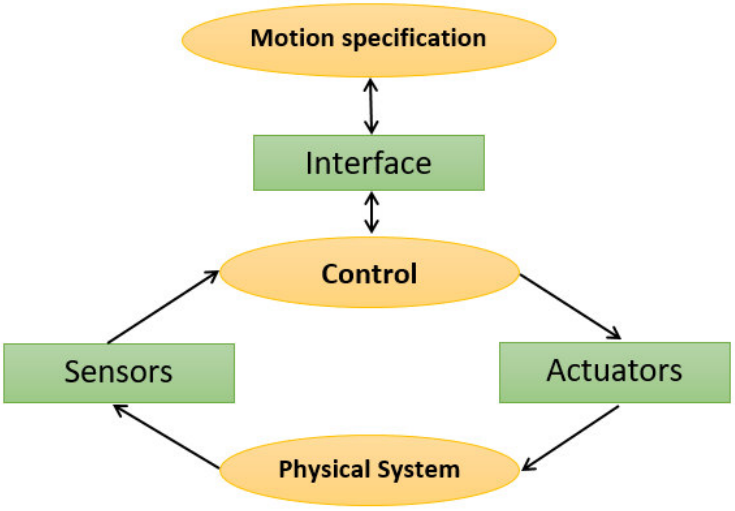


Figure 2- 4 Basic components of motion control.

To make robots more useful, the development during the last 20 years has resulted in more sophisticated specification of motions, both in terms of how motions and computations are specified/programmed, and in terms of the tools used for the programming. For example, the abstraction level and ease of use of the system were increased by having a kinematic model of the robot built into the control system, and special programming tools and languages were developed to aid the robot programmer [8]. Robot programming was still manipulator oriented, i.e. the manipulator motions were specified rather than the task to be performed.

A recent trend has been to include more knowledge about the physical system into the controller. For the control algorithms, this means that dynamic models are utilized in order to improve performance. Performance demands come from required utilization of the relatively expensive mechanical part of the system. The interface models the physical environment on some level, just like a reference signal to a simple control loop can be viewed as a model of the controlled output. The development for the interface and for the motion description has aimed at an increased level of abstraction to make robots easier to use. One example is motion commands specified as relations to the objects being manipulated. Another example is application specific task level programming, allowing the programmer to specify motions (or online adjustments of motions) in terms of production data that he/she is used to, like arc-welding parameters [9] etc.

2.5.1. Level of programming abstraction

Robot programs consist of statements for motions and for information processing. Either motion statements are built into the language, or they are achieved by use of a robot software

library. In the former case, we have a special manipulator language, e.g., languages like ARLA, AL, or AML.

It has turned out that major parts of typical robot programs consist of information processing, i.e., robot programs resemble computer programs. This implies that robot programming includes all the aspects and problems of computer programming, plus some additional ones. This is also the reason why recently introduced robot programming languages either belongs to existing computer programming languages (with a robot library), or are a new general-purpose programming languages with some special robot programming support (types, syntax, etc.). Examples of the former are RCCL, PASRO, and HAL, whereas Karel and RAPID are examples of the latter [9]

2.5.2. Most common languages programing of the Robots

2.5.2.1. RAPID

RAPID is a programming language for writing such a program. It uses some English words (like IF and FOR) to make it understandable for humans. The advantage with RAPID is that, except for having most functionality found in other high level programming languages, it is specially designed to control robots.[15] RAPID is a high-level programming language used to control ABB industrial robots. RAPID was introduced along with S4 Control System in 1994 by ABB, superseding the ARLA programming language.

- **The structure of the Language**

The program consists of a number of instructions which describe the work of the robot. Thus, there are specific instructions for the various commands, such as one to move the robot, one to set an output, etc. [16]

The instructions generally have a number of associated arguments which define what is to take place in a specific instruction. For example, the instruction for resetting an output contains an argument which defines which output is to be reset; e.g. Reset do5. These arguments can be specified in one of the following ways:

- as a numeric value, e.g. *5* or *4.6*
- as a reference to data, e.g. *reg1*
- as an expression, e.g. *5+reg1*2*
- as a function call, e.g. *Abs(reg1)*
- as a string value, e.g. *"Producing part A"*

There are three types of routines – *procedures, functions* and *trap routines*.

- A procedure is used as a subprogram.
- A function returns a value of a specific type and is used as an argument of an instruction.
- Trap routines provide a means of responding to interrupts. A trap routine can be associated with a specific interrupt; e.g. when an input is set, it is automatically executed if that particular interrupt occurs.

Information can also be stored in data, that is grouped into different data types which describe different types of information, such as tools, positions and loads. There are three kinds of data – *constants, variables* and *persistent.*

- A constant represents a static value and can only be assigned a new value manually.
- A variable can also be assigned a new value during program execution.
- A persistent can be described as a "persistent" variable. When a program is saved the initialization, value reflects the current value of the persistent.

Other features in the language are:

- Routine parameters
- Arithmetic and logical expressions
- Automatic error handling
- Modular programs
- Multi-tasking.

- **Programming Principles**

The program flow can be controlled according to five different principles:

- By calling another routine (procedure) and, when that routine has been executed, continuing execution with the instruction following the routine call.
- By executing different instructions depending on whether or not a given condition is satisfied.
- By repeating a sequence of instructions, a number of times or until a given condition is satisfied.
- By going to a label within the same routine.
- By stopping program execution.

2.5.2.2. KAREL

Karel is an educational programming language for beginners, created by *Richard E. Pattis* in his book Karel The Robot: *A Gentle Introduction to the Art of Programming* [17], it is a lower-level language very similar to Pascal. It features strongly typed variables, constants, custom types, procedures, functions, and gives you access to all sorts of useful built-ins.

KAREL is a compiled language; the source must be translated from a KAREL source file (.kl) into p-code (.pc) before it can be loaded and executed on the controller[18].

- **Structure of the Program**

The PROGRAM statement followed by the name of the program must be the first statement in any KAREL program. The END statement must use the same identifier.

Any statements within the BEGIN and END statements are executed by the KAREL program. You can probably guess what the WRITE statement does. You can optionally provide it with a FILE to write to, but by default it will write to the default TPDISPLAY, the USER screen. Each item you want to write can be separated by commas, and the CR stands for 'carriage return.' Any subsequent WRITE statements will take place on the next line [18].

- **Principle**

A program in Karel is used to control a simple robot named Karel that lives in an environment consisting of a grid of streets (left-right) and avenues (up-down). Karel understands five basic instructions [17]:

- *Move:* Karel moves by one square in the direction it is facing.
- *TurnLeft:* Karel turns 90 ° left
- *PutBeeper:* Karel puts a beeper on the square, where it is standing.
- *PickBeeper:* Karel lifts a beeper off the square, where it is standing.
- *TurnOff*: Karel switches itself off, the program ends.

Karel can also perform Boolean queries about his immediate environment, asking whether there is a beeper where he is standing, whether there are barriers next to him, and about the direction he is facing.

A programmer can create additional instructions by defining them in terms of the five basic instructions, and by using conditional control flow statements *if* and *while* with environment queries, and by using the iterate construct.

2.6. Presenting M410iB/140H arm of FANUC

The robot that we are going to use in our case, and we will present its characteristics is the M410iB/140H robot manufactured by FANUC *(see Figure 2-5).*

Figure 2- 5 Isometric view of the robot M410iB/140H

The choice of this M-420iB/140H robot is done due to its high speed and because it carries 140 Kg payload. It is in my opinion the perfect one for palletizing when the task needs speed and repeatability. This robot is cylindrical

2.6.1. The robot M410iB/140H diagram

In this section, we will see the diagram of the robot in *(Figure 2-6)*, and the *(Table 4)* contains the specifications of the robot.

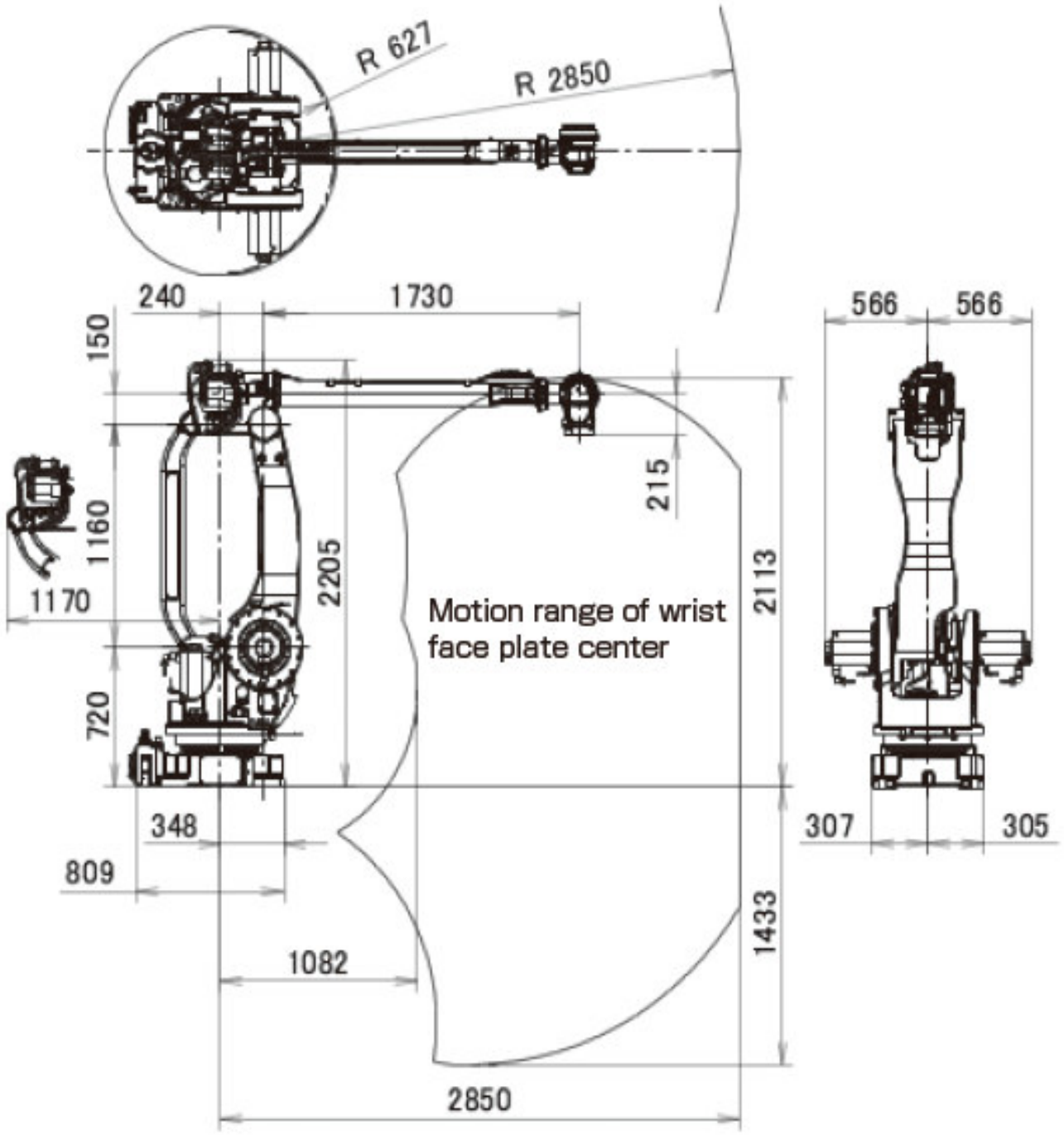


Figure 2- 6 M410iB/140H Robot dimensions

Item		Specifications
		M410iB/140H
Type		Articulated type
Controlled axes		5 axes (J1, J2, J3, J4, J5)
Reach		2850mm
installation		Floor
Motion range (maximum speed)	J1 axes rotation	360° (140°/s)
	J2 axes rotation	155° (115°/s)
	J3 axes rotation	112° (135°/s)
	J4 axes wrist swing	20° (135°/s)
	J5 axes wrist rotation	720° (420°/s)
Max. load capacity at wrist		140 Kg
Max. load capacity at J2 base		550 Kg
Max. load capacity at J3 arm		140 Kg
Allowable load inertia at wrist	J4 axis	147 $Kg.m^2$
	J5 axis	53 $Kg.m^2$
Throughput		1900 cycles/hour
Drive method		Electric servo drive by AC servo motor
Repeatability		$\pm$0.2 mm
Mass		1200 kg

Table 4 The robot specifications

2.7. The ROBOGUIDE Software

The ROBOGUIDE software is specially designed to control FANUC robots. Within this software, it is possible to simulate a working cell and to program the trajectory of a robot. The data compiled by the simulation software can then be transferred to the actual robot in order to perform a real task.

Before presenting the notions for the design of a working cell and the notions of programming, the environment and the working structure of the ROBOGUIDE software will be presented. So that the rest of what is coming in the next chapter will be easy to understand.

2.7.1. The windows of the program

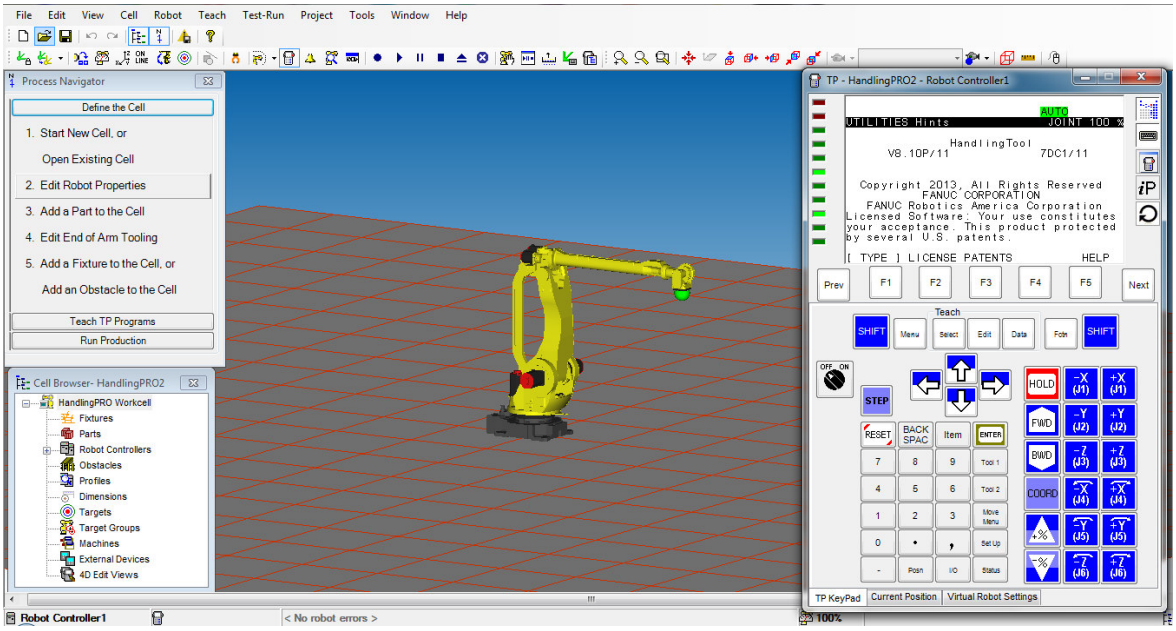


Figure 2- 7 The Window of the Program

2.7.2. The working cell structures

The ROBOGUIDE software organizes the elements according to different categories:

- The robot
- End of Arm Tooling (The tool at the end of the arm of the robot)
- Parts
- Fixture
- Obstacle

2.7.2.1. The Robot

The main element of the (WorkCell) is the robot, because it is all about it and also the ultimate goal of the software is to be able to control a robot. Inside the software library, it is possible to choose all FANUC robots on the market. Generally, the robots are fixed at the base and are equipped with pivoting arms in order to perform manipulations.

2.7.2.2. End of Arm Tooling

Another essential element is the tool at the end of the arm of the robot, also called "EoAT". The tool makes the link between the robot and the parts, it allows to take or release a part.

There are several types of tools, for example:

- Grippers
- Vacuum
- Welding torch (weldtorch)

2.7.2.3. The Parts

The rooms are the elements that we want to manipulate with the robot. The parts can be modeled on a 3D software like SOLIDWORKS and imported inside ROBOGUIDE.

2.7.2.4. Fixture

To be able to take or deposit a part in the ROBOGUIDE software, the part must always be tied with a fixture, it is an object on which the parts are based.

2.7.2.5. Obstacle

Obstacles are the elements that represent the different objects that could interfere with the robot's trajectory. It is important to incorporate obstacles that are present in the actual working environment within the simulation software to ensure that the robot in reality does not collide with objects.

2.7.3. Commanding instructions

2.7.3.1. Create the tool

- **Process navigator**

It is a menu containing the different stages of design of a working cell. The steps are arranged in the logical order of creation.

To access the process navigator, you must go to the "View" tab of the main menu and click "Navigator". You can also access the process navigator by clicking directly on the button

The process navigator is divided into three parts:

- *Define the Cell:* Creating elements to fill the work cell, for example:
  - rooms
  - Robot tools
  - Bindings
- *Teach TP Programs*: Creating a program
- *Run Production:* Roll the program and analyze its profile.

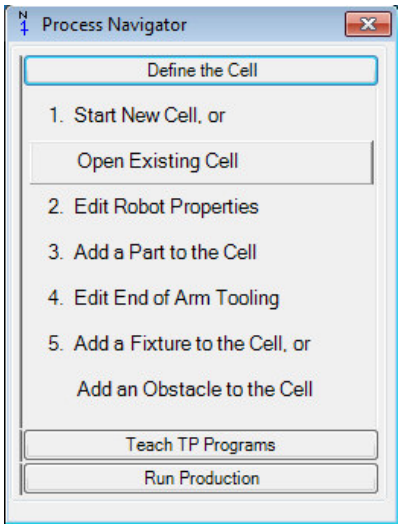


Figure 2- 8 Process Navigator

- **Cell browser**

It Is a window where you can find the different elements and the different programs of the work cell. The tree regroups the elements as it's shown in the image below.

To access the cell browser, you need to go to the "View" tab of the main menu and click "Cell Browser". You can also access to it by clicking directly on the button.

In addition, you can add or delete an item by choosing one of the options when you right click on an item.

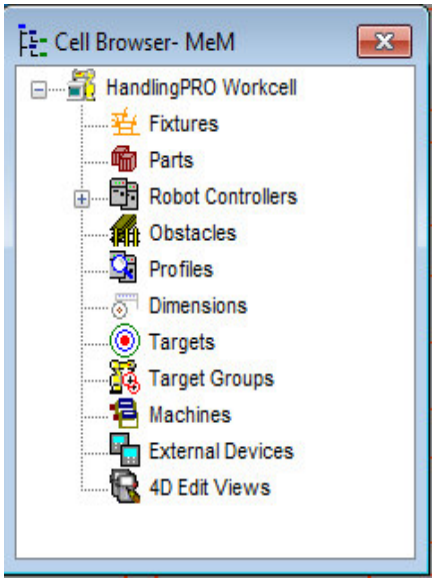


Figure 2- 9 Cell Browser

- **Teach Pendant**

The control box is breeding a few details of the actual remote-control robot. With the remote control, it is possible to move the robot and to program a trajectory.

To access the virtual remote-control window, click the button.

The use of "Teach Pendant" will be explained later.

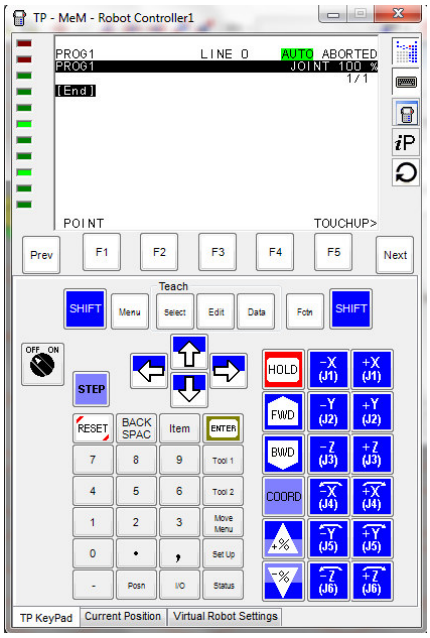


Figure 2- 10 Teach Pendant

2.7.3.2. Move the Robot

- **Move the robot using the "Teach Pendant" commanding box**

To move the robot using the command box, you must first activate it by clicking the button "ON". then, you must click on one of the boxed buttons below while holding down the "shift" key. When you move the robot using these commands you will see that you rotate the different joints of the robot.

To change the coordinate system, click on "COORD". The coordinate system choices are: JOINT, S / JGFRM, S / WORLD, S / TOOL, S / USER.

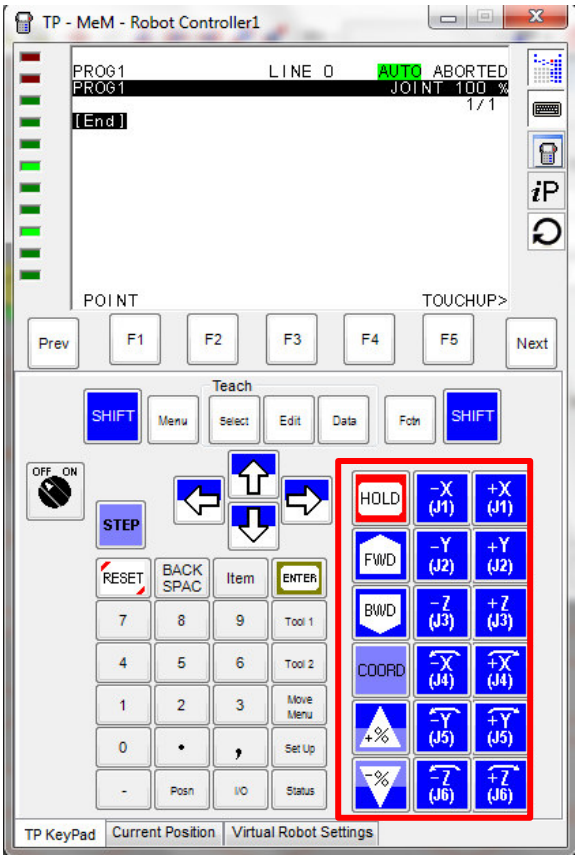


Figure 2- 11 Movement buttons on the control box

To avoid always pressing the "shift" key to change the position of the robot, it is possible to change the options so that the "shift" key is activated automatically by clicking on one of the buttons shown above. To activate this option, you must go to "Tools" in the main menu and click "Option". A window will appear and you must check the option "Jog keys automatically apply the SHIFT key" as shown below.

You can know the orientation of the different joints when you click the "Current Position" tab at the bottom of the remote. If you select "X, Y, Z" instead of "Joint" you will get the Cartesian coordinates of the "TeachTool" relative to the origin of the robot. If you select "USER" you will get the coordinates of "TeachTool" related to "UFrame" that is selected, as shown in the figure below.

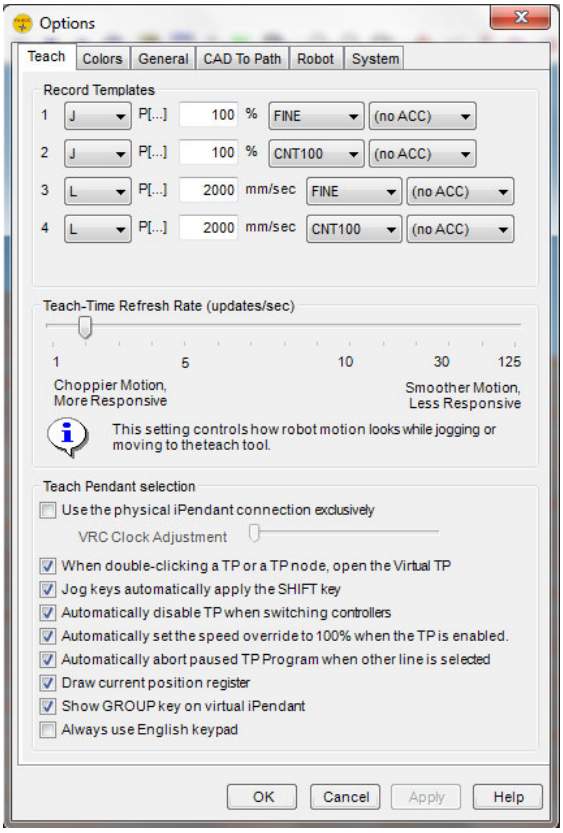


Figure 2- 12 Advanced option for moving the robot onto ROBOGUIDE

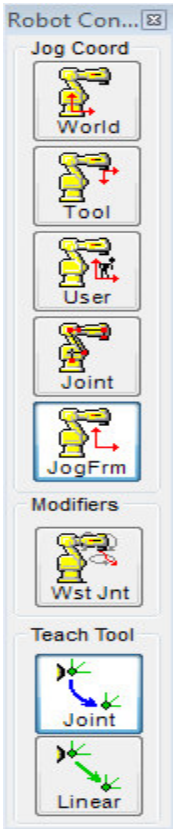


Figure 2- 13 "TeachTool" Tool bar

You can also move the robot relatively to the current position by clicking the button "Relative". If you choose the option "Tool", you change the location of the "TeachTool" relatively to the current positioning of the tool. If you choose the option "Joint", you can decrease or increase the angle of the joints. If you choose the option "X, Y, Z", you change the location of the "TeachTool" relative to the origin of the robot. Finally, if you choose "USER", you change the location of the "UTool" relative to a selected "UFrame".

- **Moving the robot using the tool "TeachTool"**

You can move the robot by sliding manually the "TeachTool", represented by a green ball. To do this, you must click on the button # to display the "TeachTool" axes. Then you can move the green ball by dragging the mouse over its axes. It is also possible to change the orientation of the "TeachTool" by holding down the "SHIFT" key and dragging the mouse on its axes. In addition, it is also possible to move the green ball freely by holding down the "Ctrl" key while clicking on the "TeachTool" and moving the mouse.

You can change the motion reference point by clicking the button #. A window will appear as shown below. This will give you the option to choose the origin (World), the tool (Tool), the user (User), joints (joints) as a reference point for the movement.

WARNING. If the robot does not move to where you moved the "TeachTool" it may be because the point is out of the reach of the robot, or that your axis "Z" is oriented in the wrong direction.

- **Moving the Robot using the Joints**

You can move the robot by changing manually the orientation of the joints. To do this, you must click on the button and then drag your mouse over the different joints of the robot as shown below.

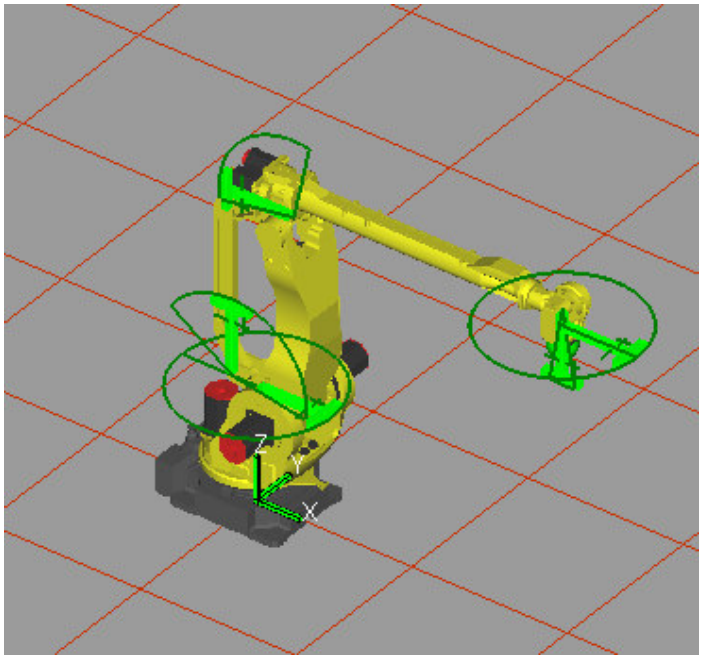


Figure 2- 14 Manipulation of robot joints on ROBOGUIDE

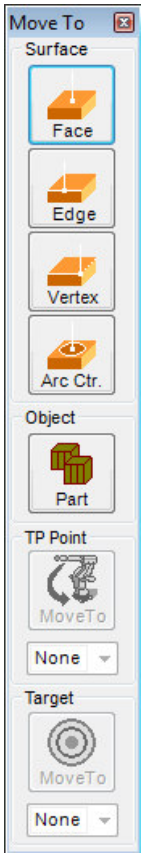


Figure 2- 15 "Move to" tool bar

- **Moving the Robot automatically using the tool "Move to"**

The robot can be moved automatically using the tool "Move to". To do this, you must click on the button #. A window, image below, will appear. In this window, we have in this window

the option of directing the "TeachTool" on a front (face), a side (Edge), a corner (Vertex), an arc (Arc Ctr) or a part (Part).

To use this tool, you must first select one of the options mentioned above, then click on a region of your work cell that corresponds to where you want to move the tip of the robot arm.

2.7.3.3. Using the Measurement Tool

To access the measurement tool, we must click on the button. A window containing the measurement options as shown in the figure below will open.

To measure the distance between two objects or the length of a part, first click the button "From" and then select the type of surface you want to select: face, edge, vertex, face center or the edge center. Next, you must select where you want to position the measurement start point.

To choose the end point of the measurement, you must repeat the above steps by choosing the desired characteristics for the end point and then click on the position of the end point.

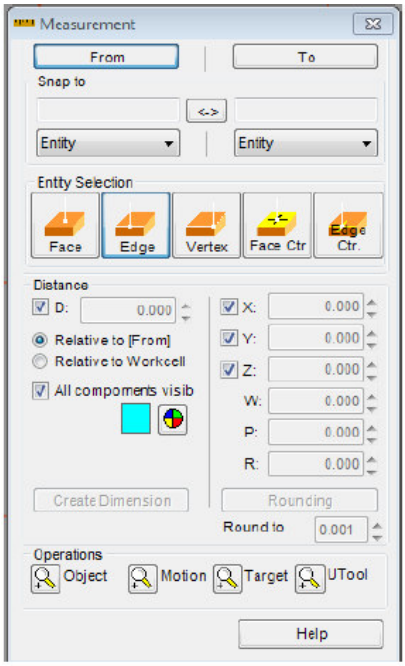


Figure 2- 16 Measurement Tool Window

2.7.3.4. How to export the program to the Robot

To export a file to the actual robot, first click on the file and then click on "Export / To" Robot as shown in the *(Figure 2-12)*.

A window, as shown in the (*Figure 2-13)*, will appear. If you have not already done so, you must select the robot to which you want to export the robot. Finally, you must click on "Export" and the file should normally be sent to the robot.

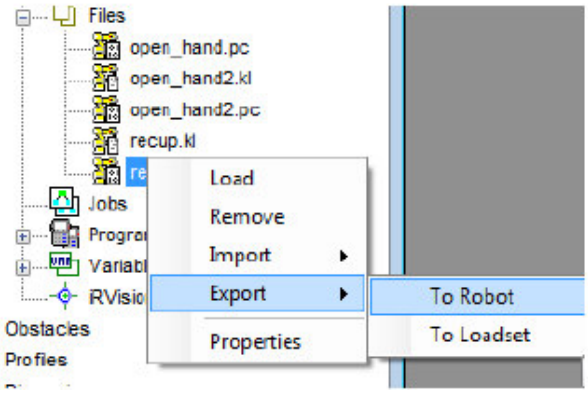


Figure 2- 17 Exporting a file from the computer to the robot step 1

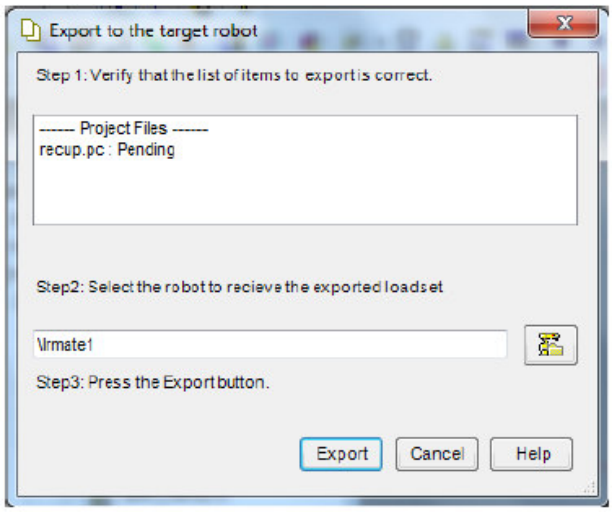


Figure 2- 18 Exporting a file from the computer to the robot step 2

Chapter 3

Robot Simulation (Palletizing)

3.1. Introduction

The software ROBGUIDE allows us to create the robot with its environment, to simulate the actions and the motions of the robot, to let us choose the adequate tool and edit the basic specifics, such as weight and height, and this is what is exactly I'm going to do in this chapter, I am going to create a work cell, as in work shop, to make it close to reality, and the creation of the robot is included with the creation of the tool and the parts, considering the application is Palletizing, after all that the program should be done and executed to simulate how the robot will act and move.

3.2. The creation of the robot

The choice of the robot M140iB/140H and the main reason have been discussed in the previous chapter, to choose the wanted one, we need to check on the brochure by FANUC which represents all the robots with all of their settings and parameters *(see figure 3-1).* now all that is left to us is find this robot and create is it in the work cell of this software, thus it will be followed by creating all the rest of the objects.

Figure 3- 1 Fanuc Brochure of M410iB robots

First of all, after opening new file of ROBOGUIDE, it requires to set some parameters and choose the Robot (as there are plenty of them according to their types, design and their mass as well) Then, we select the robot on the list, and it will show up in the work cell. *(See figure 3-2)*

About the type of the work cell, the software offers 6 types; HandlingPRO, WeldPRO, PalletPRO, PickPRO, PaintPRO and MotionPRO.

Our subject is about the robot to handle carrying parts and displace them, so the work will be with handlingPRO

HandlingPRO is an offline robot simulation software product, built on the virtual robot controller, it allows users to simulate a robot process in 3-D space or conduct feasibility of the robot application without the physical need. with HandlingPRO, sales, proposal and application engineers can import unique CAD model of parts, create a work cell including machines, part transfer devices and obstacles, and teach robot path to simulate the operation and performance of a multi-robot work cell.

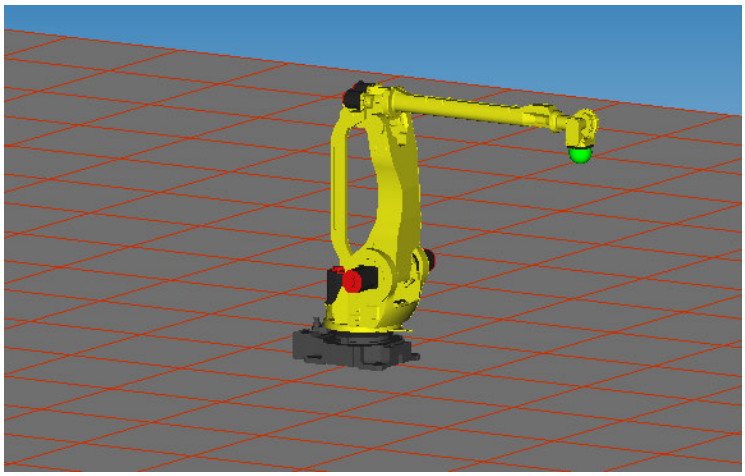


Figure 3- 2 M410iB/140H robot in the work cell

3.3. The choice of the part and the handling tool

3.3.1. Creating the part

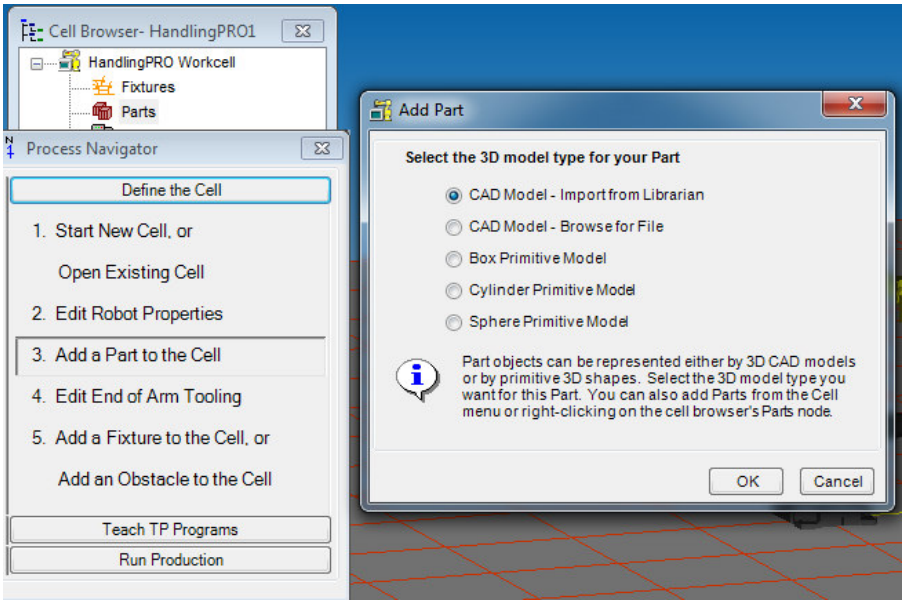
The main objective of the process is that the robot handles a part, and for this the part has to be created, the reason why we should create the part in first place stops in editing the parameters of the part according to the other fixtures when they are created.

The software offers us three types of parts (box, cylinder and sphere), but it is also possible to import any part from any other CAD software like SOLIDWORKS.

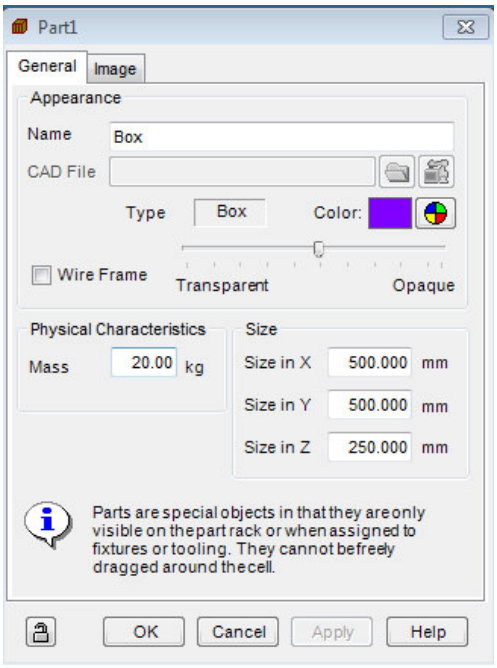
I chose a box as shape of the part, because generally the palletizing consists on palletizing boxes or objects packaged in boxes.

The instructions of the creation are as follow:

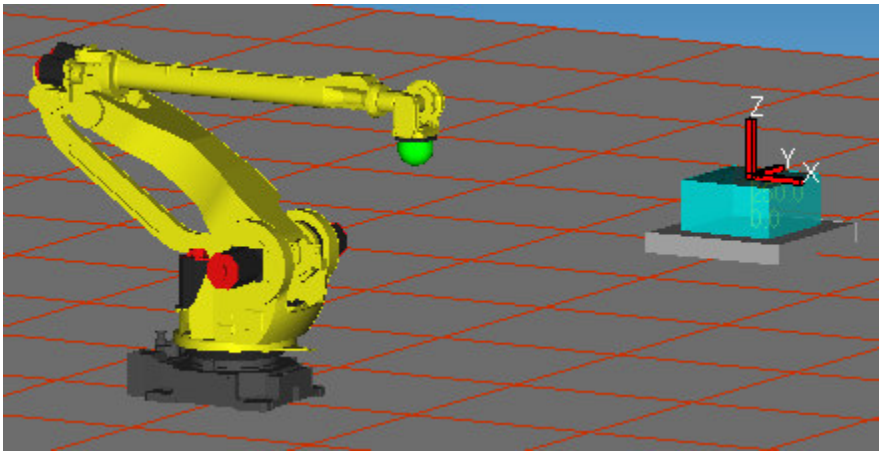
- Inside the section "Define the Cell" of the "Process Navigator" click on "Add a Part to the Cell", a window with a different option concerning the nature of the part will open.
    - As part of this topic, select "Box Primitive Model" and click OK.



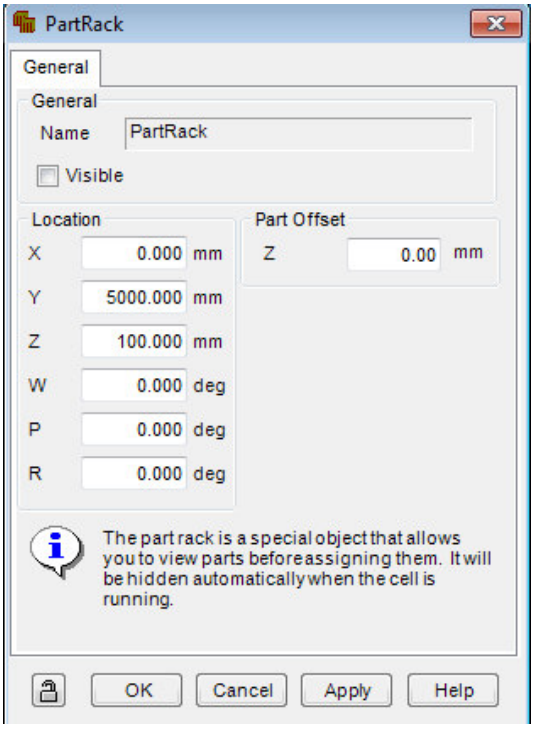
A window with the properties of the part will appear.



- Give the part a name, for example: box
- We can change the color of the box by clicking on the color palette.
- In the field "Mass", change the mass to 20 kg, as the robot can withstand a maximum of 140 kg including the mass of the end effector.
- In the field "Size", change the dimensions for 500 – 500 – 250.
- Click OK, we should have a platform (Part Rack) topped by the part next to the robot.

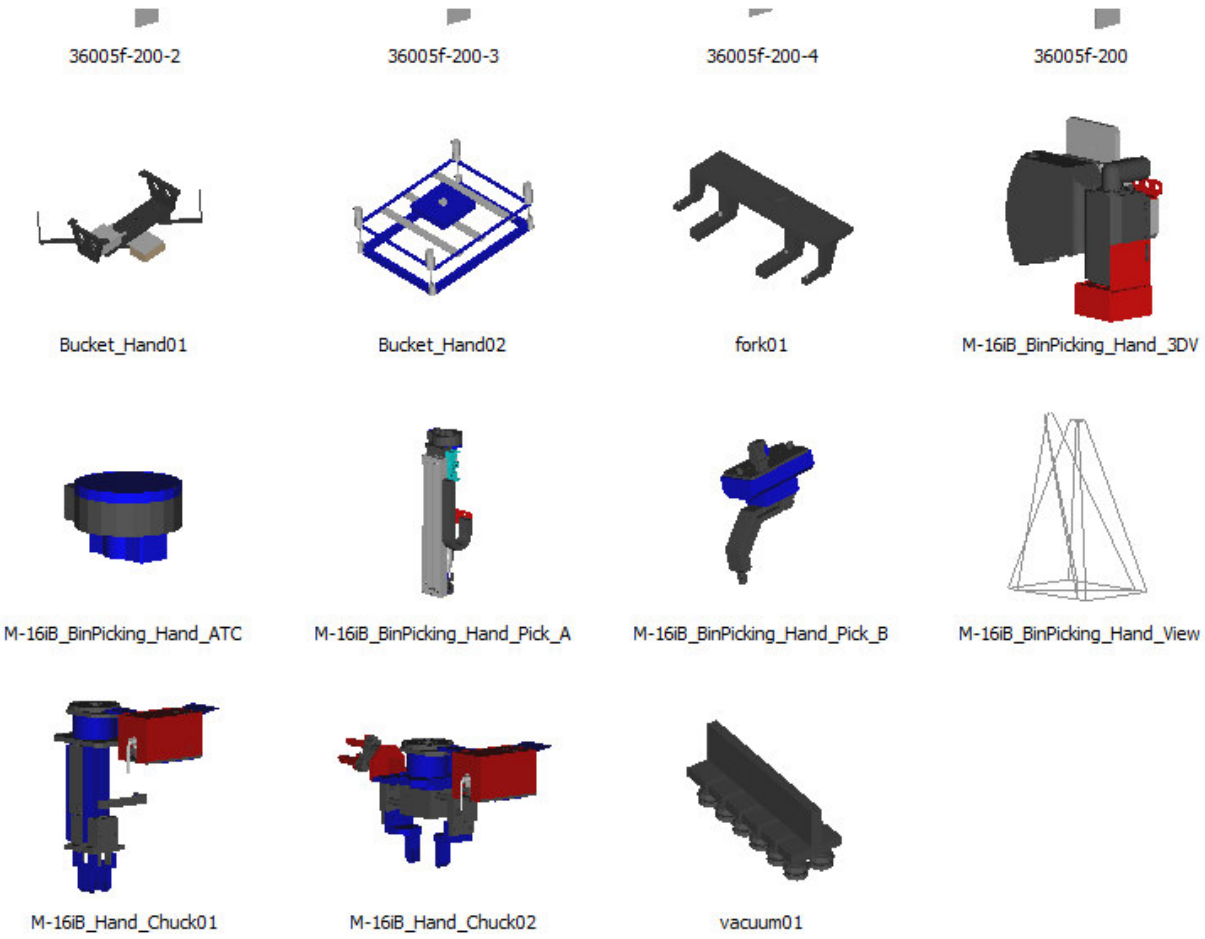


- Click by right on the parts in the cell browser, a tab of options will appear, choose PartRack properties, then a window will open, in this one we have to uncheck the field “visible”



3.3.2. Creating the tool

The creation of the end effector also known as End-of-Arm-Tooling is also as mandatory as the creation of the part, because there will be no picking up parts if there is no tool to do it, the robot we have got has not an end effector (tool in its extremity), it is up to us to create it, as there is plenty of tools, each one of them is supposed to do a specified task, also the type of robot and the application must be taken in consideration.



The application that I made for this robot is palletization and the adequate tool to be used in the process is a vacuum tool, the vacuum is a gripper that does the same task as all of the grippers, it picks up a part and take it somewhere else then drop it, and what made it better for this task is due to its benefits:

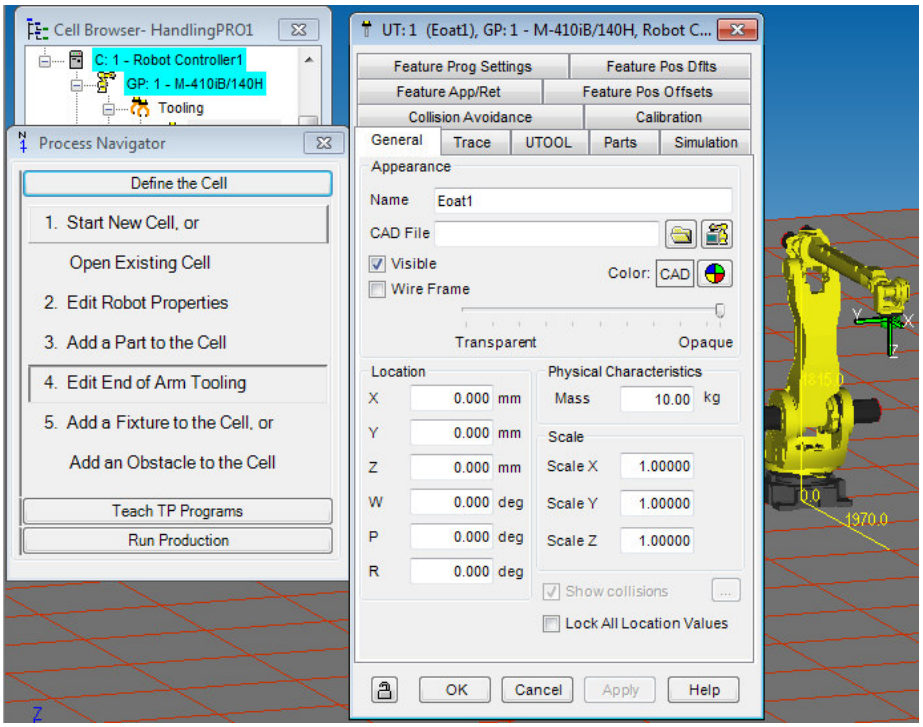
- ✓ Integrated vacuum generation.
- ✓ Energy saving function as intelligent valve technology closes unused suction openings.
- ✓ Contamination-proof design ensures high level of availability and process safety.
- ✓ Suitable for short cycle times.

✓ Simple installations and operations.

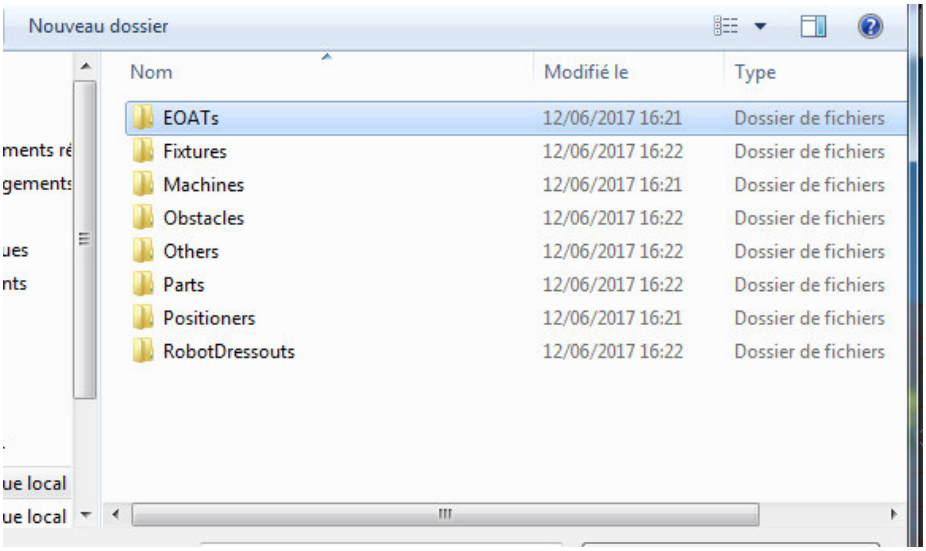
✓ Wide range of standard products and fast development of customized solutions.

The software offers us different tools and among them this suitable one, and to create it attached to the robot we have to follow these instructions:

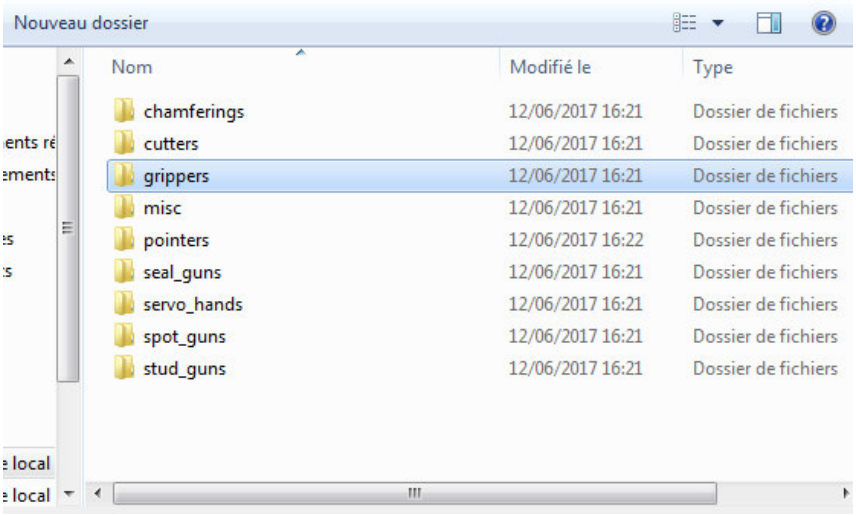
- Inside the section "Define the Cell" of the "Process Navigator" click on "Edit EoAT", a tool properties window will open.



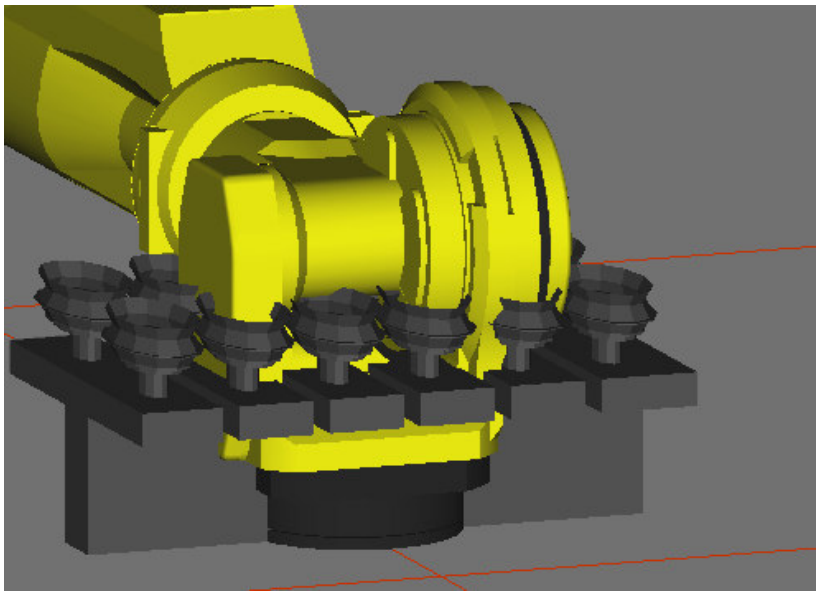
- Import an existing tool by clicking on the folder next to the field "CAD File", a window to browse the file will open.



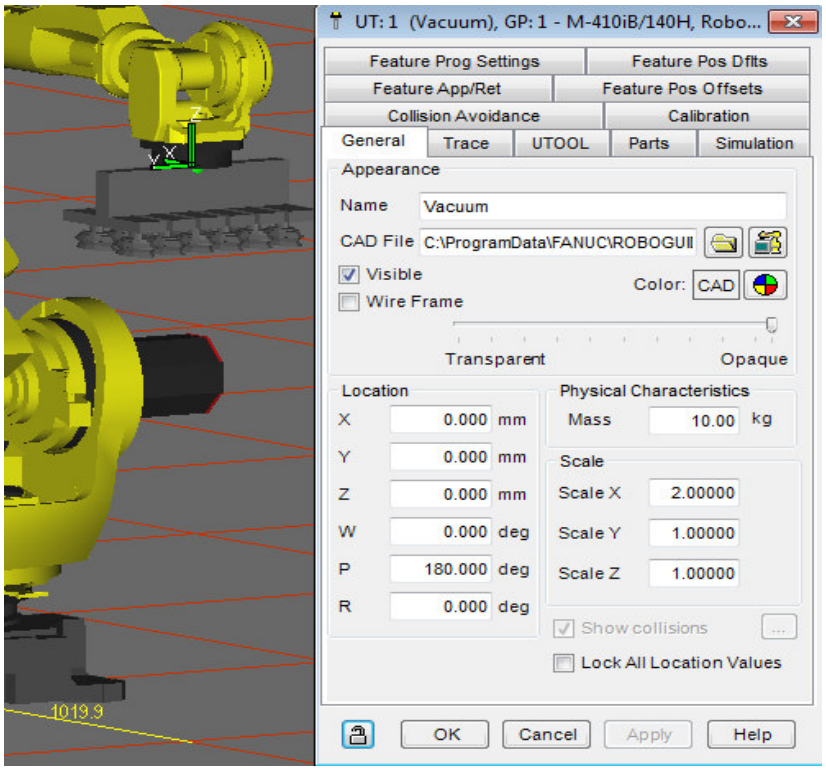
- Go to EoAT / grippers and select the file "Vacuum01" and click open.



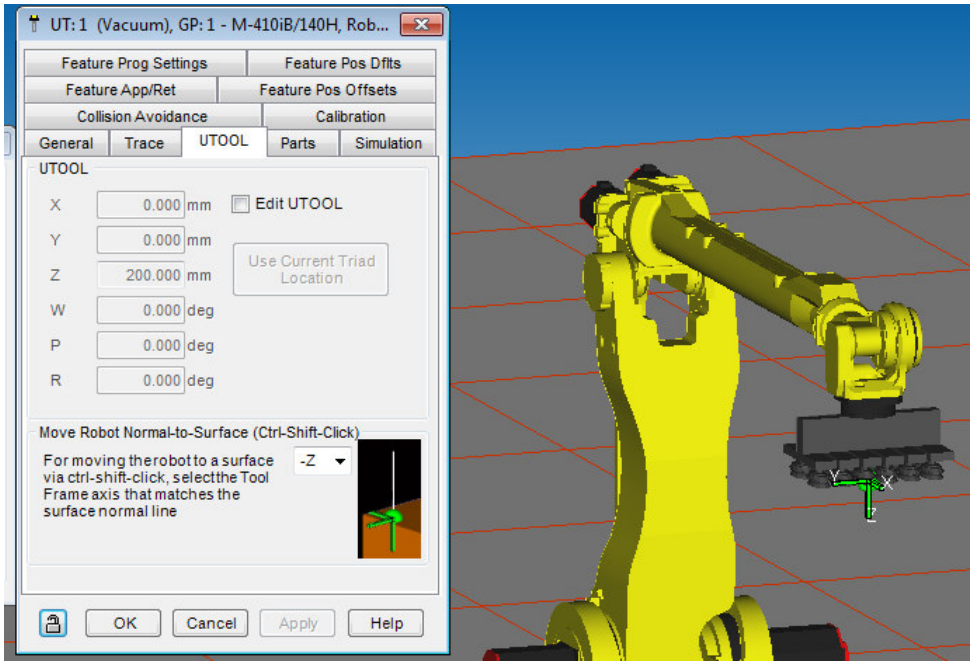
- Change the weight of the gripper to 10 kg by changing the field "Mass"
- Change the room scale to X=2 to give a proxy size proportional to the robot. Then click OK.



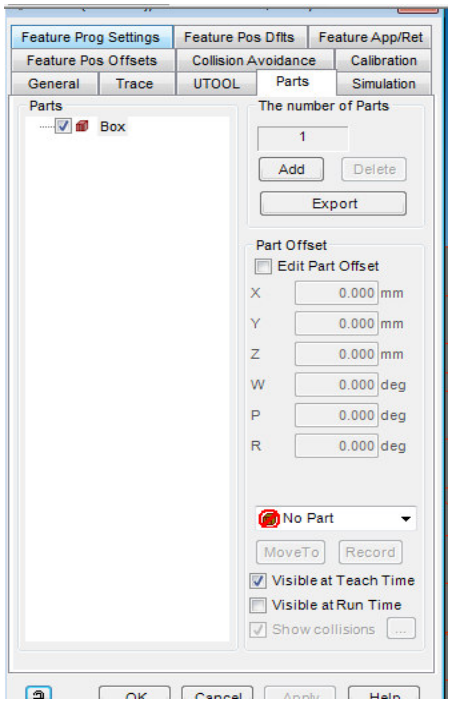
- We will notice that the gripper is not installed correctly. To correct the situation, double-click on the gripper to reveal the properties of the tool and change the value of the "P" field to 180. This will rotate around Y axis of 180° as it is shown. Then click Apply.



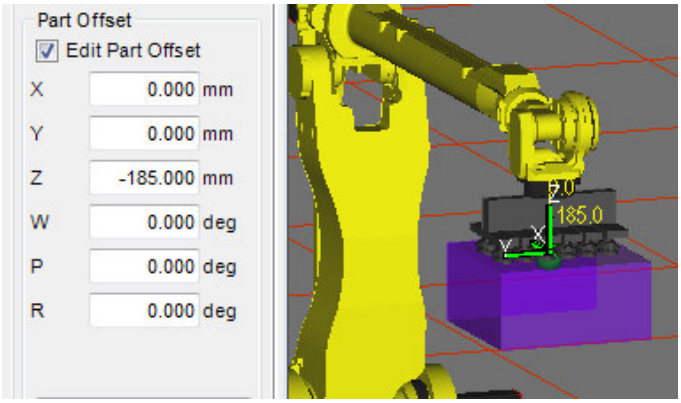
- It is now necessary to modify the point of contact of the robot with the part. To do this, click on the tab "UTool" of the gripper properties and check "Edit UTool".
- we can now enter values for the position of the contact point relative to the repository at the end of the robot arm. In this example, enter the value 200 in the field "Z" and click Apply. The green sphere should now be at the point of contact of the gripper.



- It is now necessary to determine the location of the part when the robot takes the part. To do this, go to the tab "Parts" of the gripper properties and select it on the desired part (box) and click Apply.



- In the same tab, select "Edit Part Offset". You can now change the position of the part relative to the repository at the end of the robot arm. In this example, enter the values X = 0, Y = 0, Z = -185, W = P = R = 0. Then click Apply.
- Now we must determine the interaction of the vacuum gripper during the simulation. To do this, go to the tab "Simulation" of the gripper properties and select the desired part. Then select "Material Handling - Vacuum" for the field "function" and choose the file vacuum01.igs in EoAT / grippers trajectory by clicking on the file next to the field "Actuated CAD". Click OK.

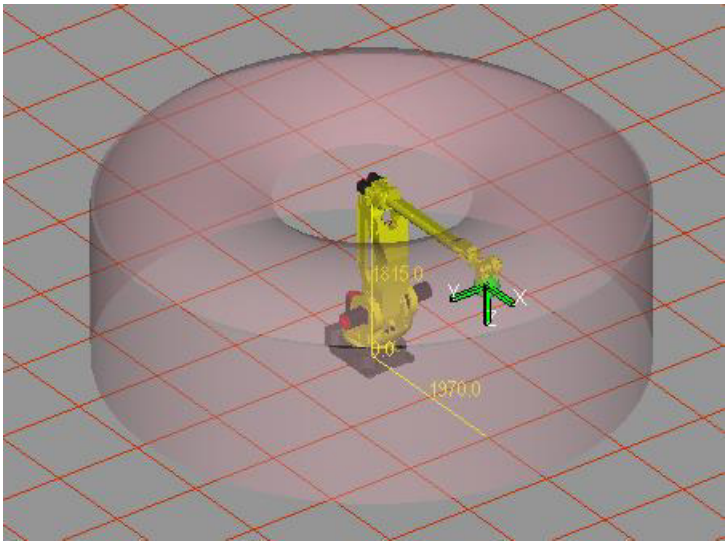


3.4. The creation of the work Cell

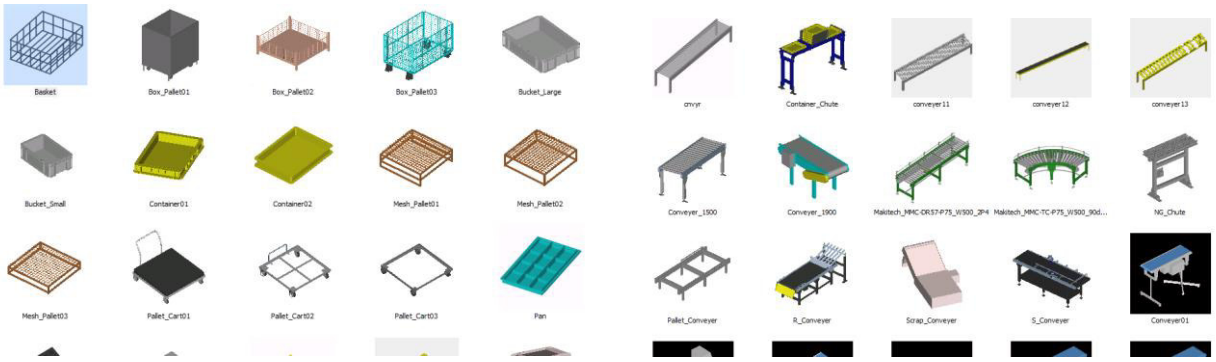
Nevertheless, the robot by itself won't be sufficient to realize the needed task, so it is mandatory to add equipment and fixture materials, i.e. the robot is aimed to work in a well-conditioned environment that contains all the necessary objects, one more reason is to facilitate to whole process.

The most important thing to be rigorously respected here is the work envelope of the robot. Before creating any fixture or object, make sure that it will be placed in the right place, and for those that carry the part and the robot need to reach them should be placed inside the work envelope. For this robot, the work envelope is a cylinder indeed.

To know how to determine the work envelope of the robot; go to "Robot" in the taskbar and choose "Show work envelope".



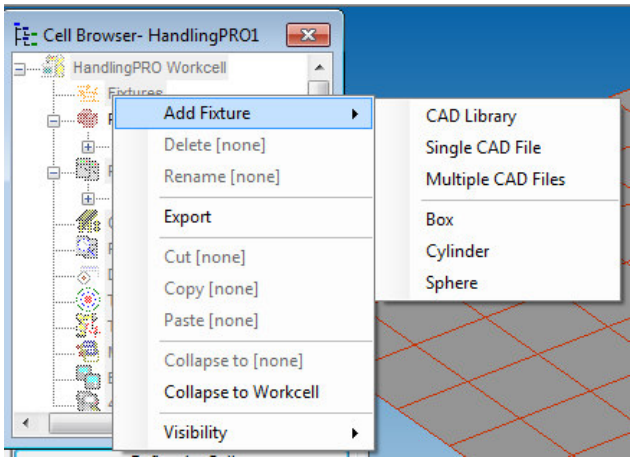
While the robot should be installed to the floor so do the other fixtures, fortunately the software we use contains almost all what we need in workshop.



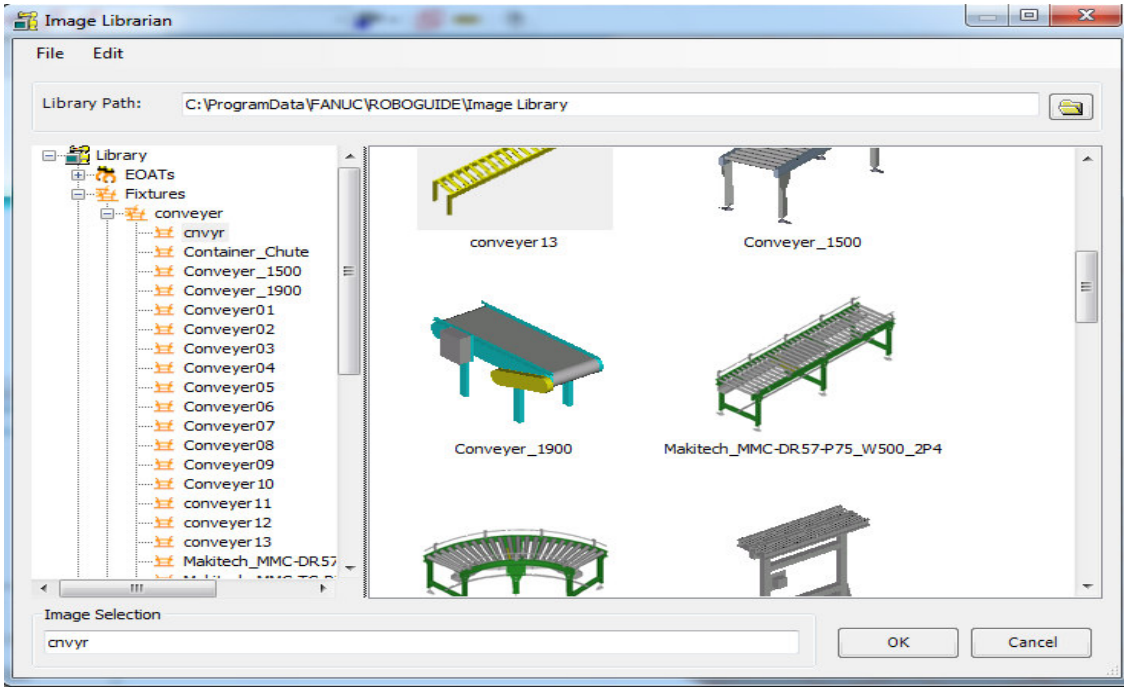
In first place, according to my work cell design, I am going to use a conveyor that brings parts to the robot, when the robot receives them, it will pick them up then take them to a pallet and drops them one by one until it fills it with nine parts (then two workers will take that pallet to be stocked and so on)

So, let us create the conveyor where the parts have to be picked, to do so, we need to follow the following instructions:

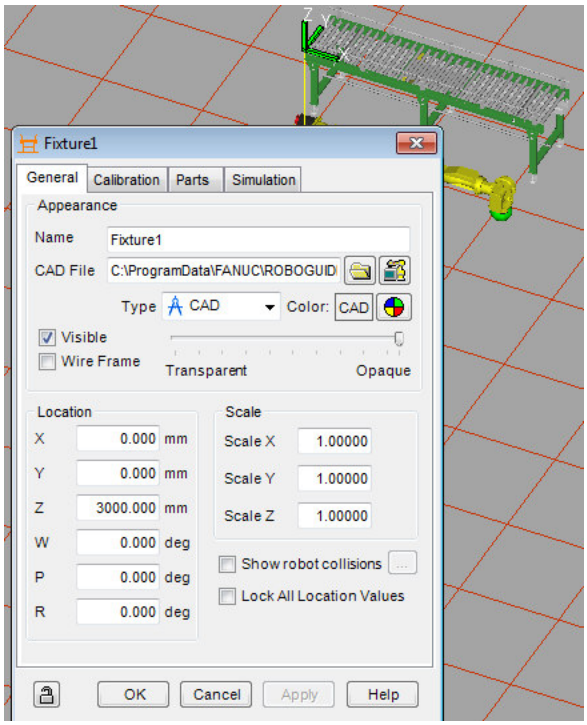
- Click on fixture in the "Cell Browser", go to "Add Fixture" and choose an option, e.g. choose "CAD Library". A window with the fixtures library will open.



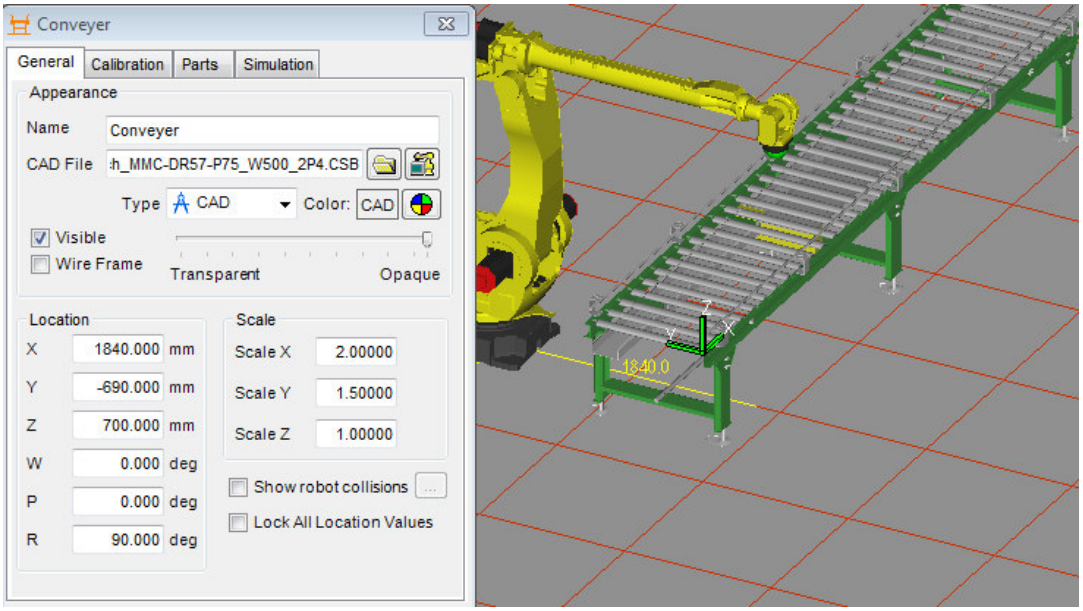
- The fixtures we get: conveyer, jig, machine, misc., motors, pallets, part feeder, racks, shelfs, table…etc.



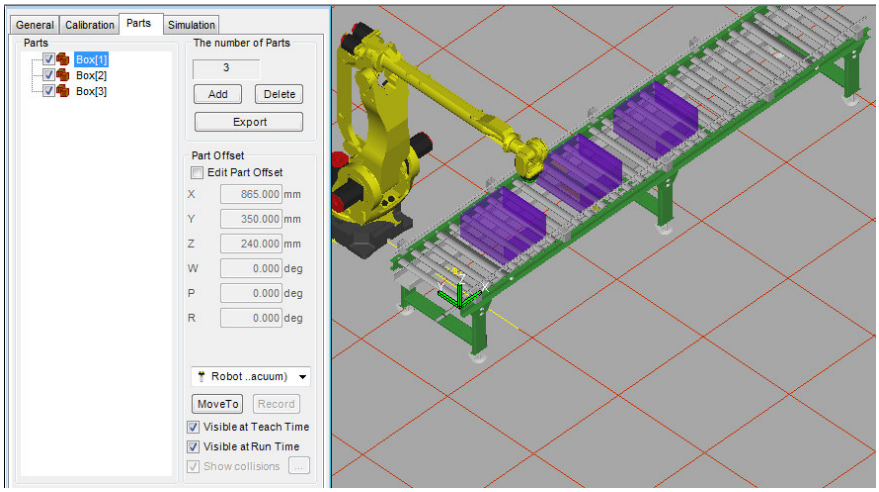
- Click the small "+" to the left of "Fixture" for a list of all fixing types. Choose the file "conveyer" and then “Makitech_MMC-DR57…” and click OK. A window will open with the properties of the fixture.



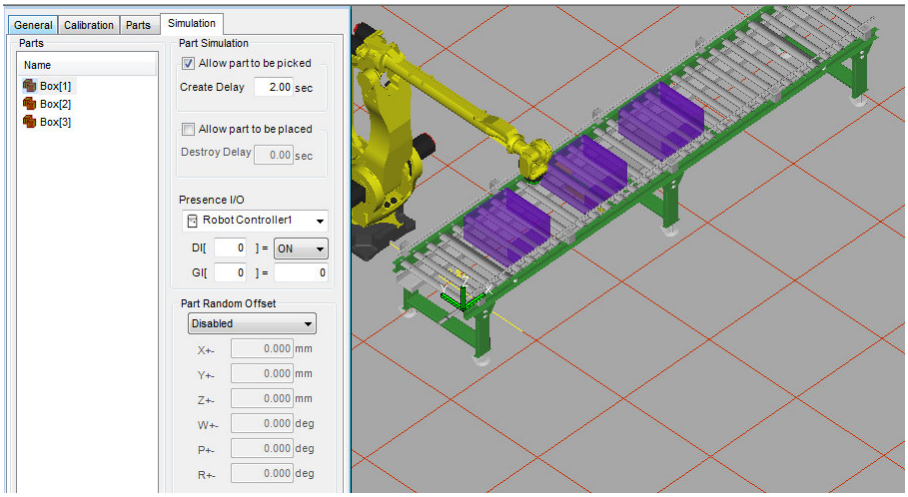
- Name the fixture by changing the "Name" field, for example: Conveyor.
- Change the scale of the fixture to X = 2, Y = 1.5, Z = 1 and change the location of the object for X = 1840, Y = -690, Z = 700, W = P = 0, R = 90. Then click Apply.

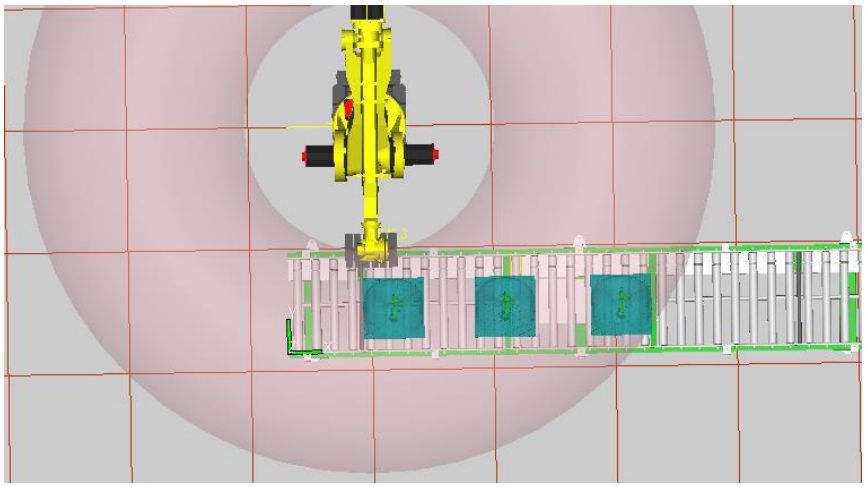


- Configure the part on the fixture by going to the "Parts" tab of the fixture properties, select the desired part, e.g. "box" and click Apply.
- In the same tab, check "Edit Part Offset" to change the location of the part on the fixture. Enter the values X = 865, Y = 350, Z = 240, W = P = R = 0. Then check in the toolbar, in the show work envelope if the object is still inside the work envelope, so the robot can reach it when it works.
- Then in the same tab click on add parts and choose the number by axis, X=3, Y=1, Z=1. Then click Apply



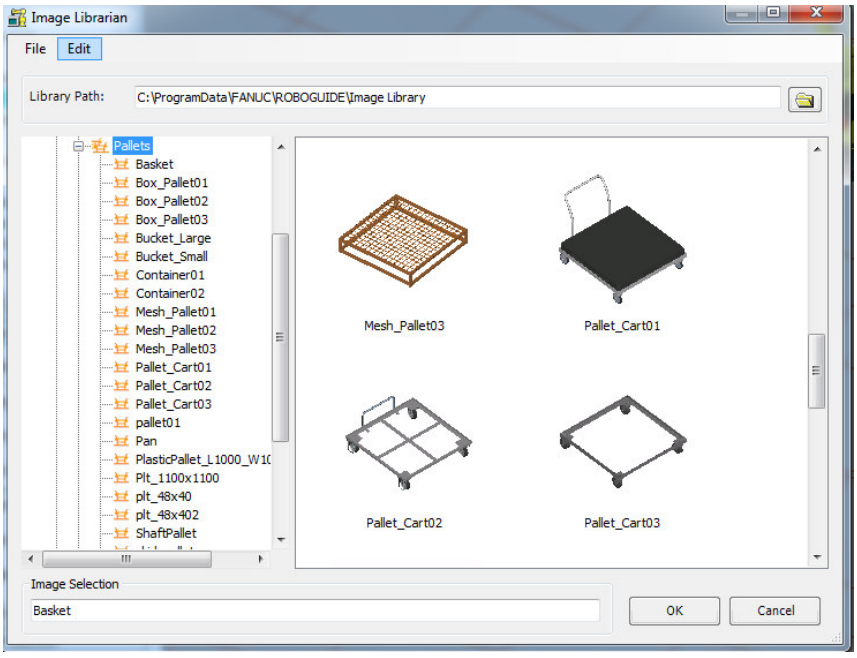
- Modify the simulation parameters by going to the tab "Simulation" of the fixture properties and click on the desired part "box". Then, check the field "Allow part to be picked" to allow the program to take this part on the fixture and change also the time to 2 sec. Then click Apply and then OK.



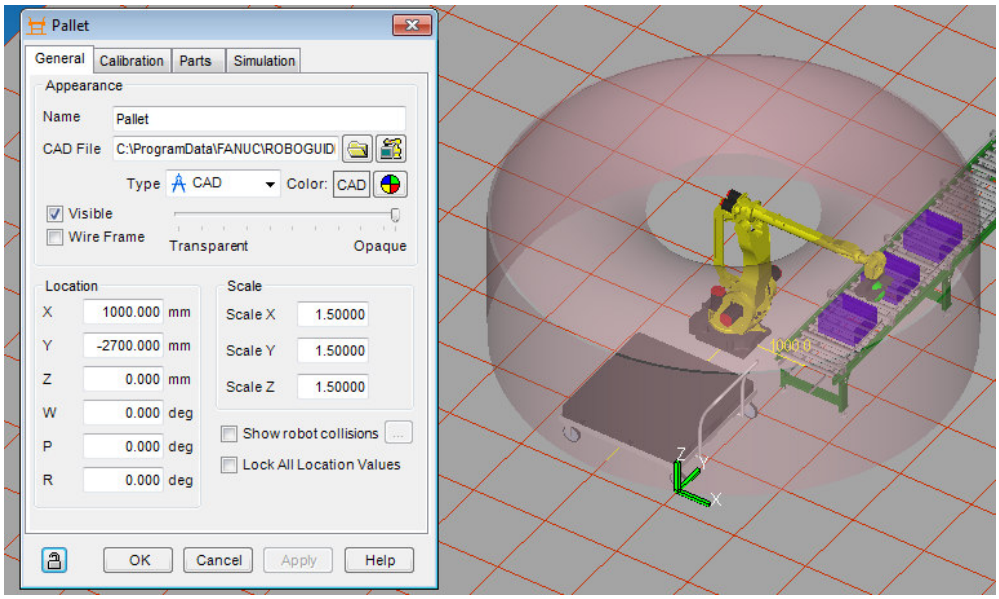


Then, we need to create a fixture (Pallet) to place the parts on it, by following the following instructions:

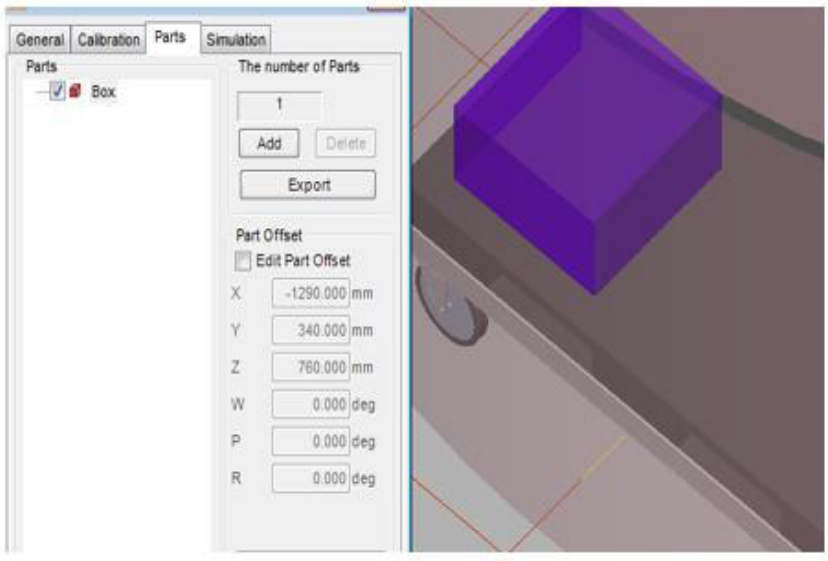
- The steps are substantially the same as creating the previous fixture. On the other hand, this fixture will be specially configured to allow the deposit of the parts.
- Create a new fixture but this time choose the file ‘Pallet” and then choose “Pallet_Cart01” and click OK. A window will open with the properties of the fixture as the previous one.

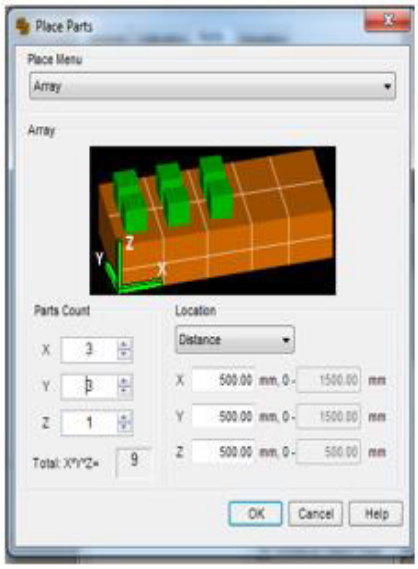


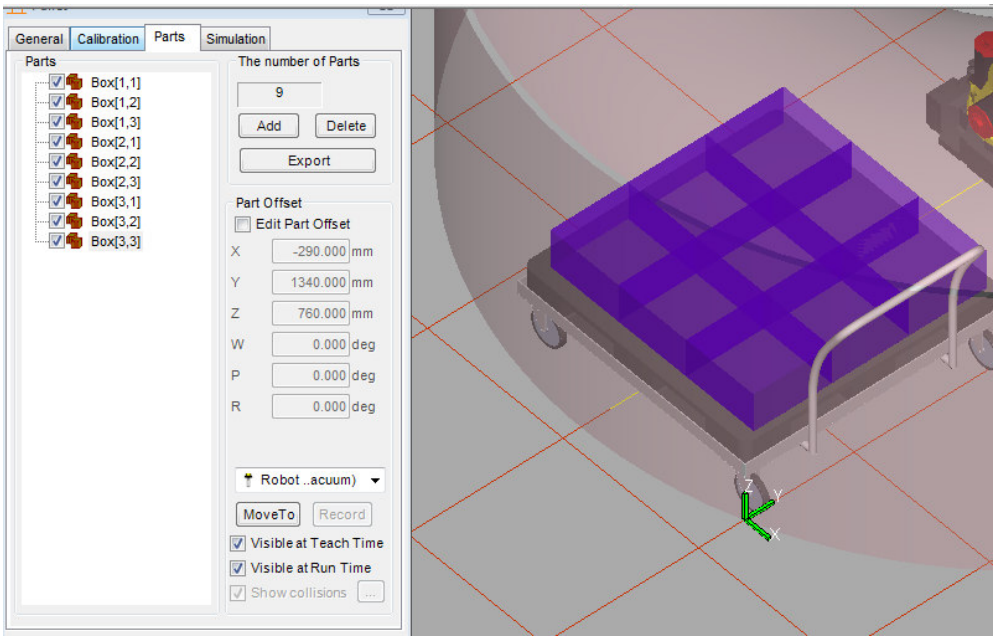
- Name the fixture by changing the "Name" field, for example: Pallet.
- Change the scale of the fixture to 1.5 and change the location of the object for X = 1000, Y = -2700, Z = W = P = R = 0. Then click Apply.



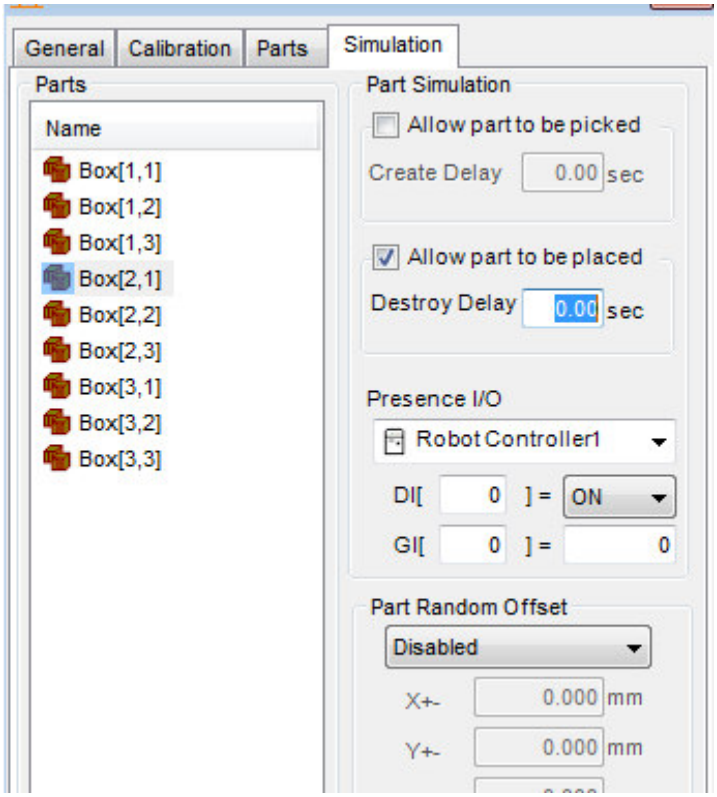
- Configure the part on the fixture by going to the "Parts" tab of the fixture properties, select the desired part, e.g. "box" and click Apply.
- In the same tab, check "Edit Part Offset" to change the location of the part on the fixture. Enter the values X = -1290, Y = 340, Z = 760, W = P = R = 0. Then see if the object is still inside the work envelope, so the robot can reach it when it works as it has been said beforehand.
- Then in the same tab click on add parts and choose the number by axis, X=3, Y=3, Z=1. Then click Apply



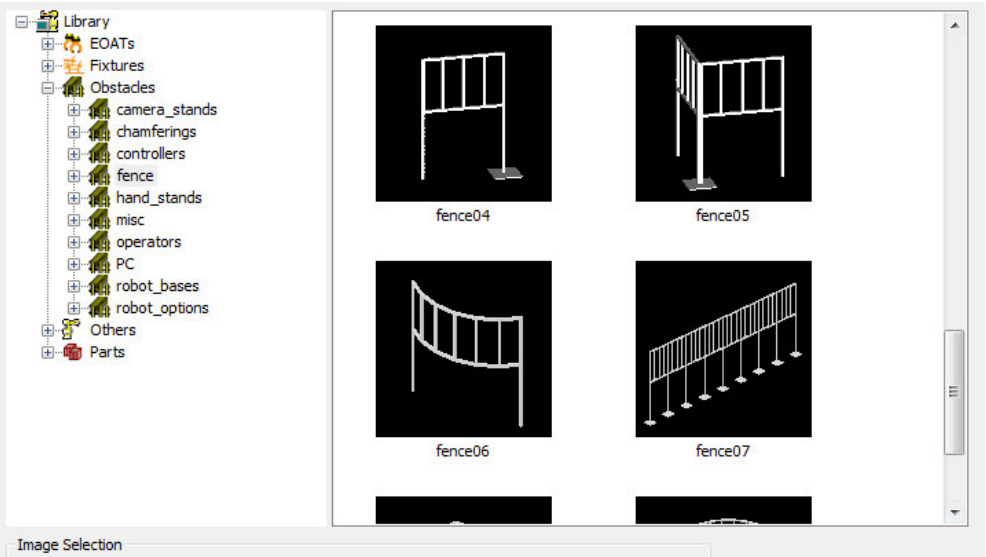




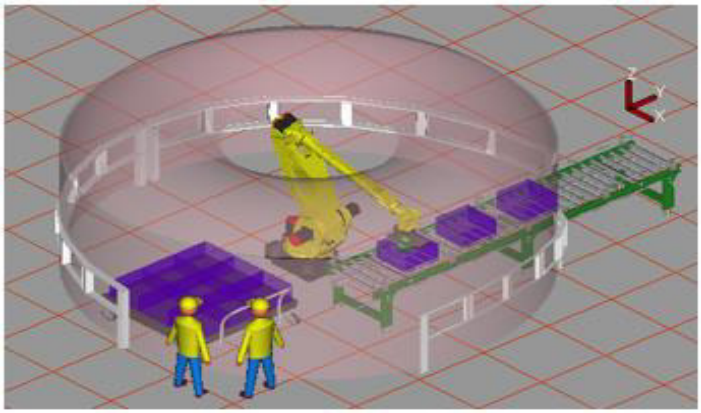
- Modify the simulation parameters by going to the tab "Simulation" of the fixture properties and click on the desired part "box". Then, check the field "Allow part to be placed" to all of them, to allow the program to place these parts on the fixture and change also the time to 2 sec. Then click Apply and then OK.

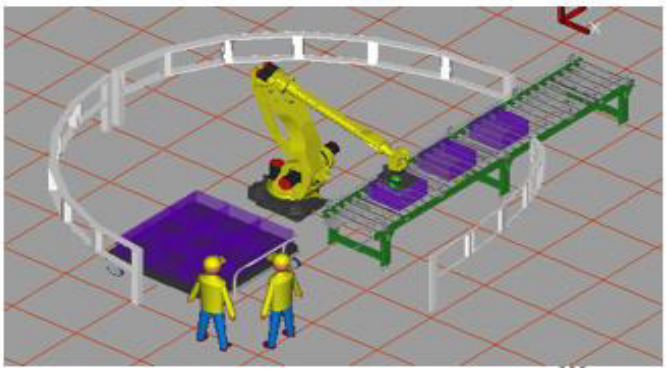


Those are the principal fixtures of the studied subject, but we can't neglect the security of the environment, so we need to secure it by creating obstacle that are going to surround the cell according to the robot's work envelope.



By that we go to obstacle, and choose to add obstacle, a window of all the obstacles will open, we choose the list "fence" then we choose the file "fence06" and place it by changing the properties, the value and the coordinates.



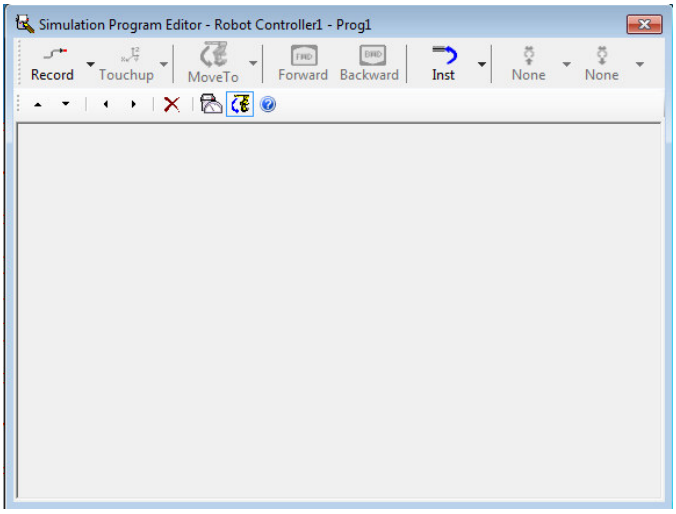


3.5. Simulation and programming

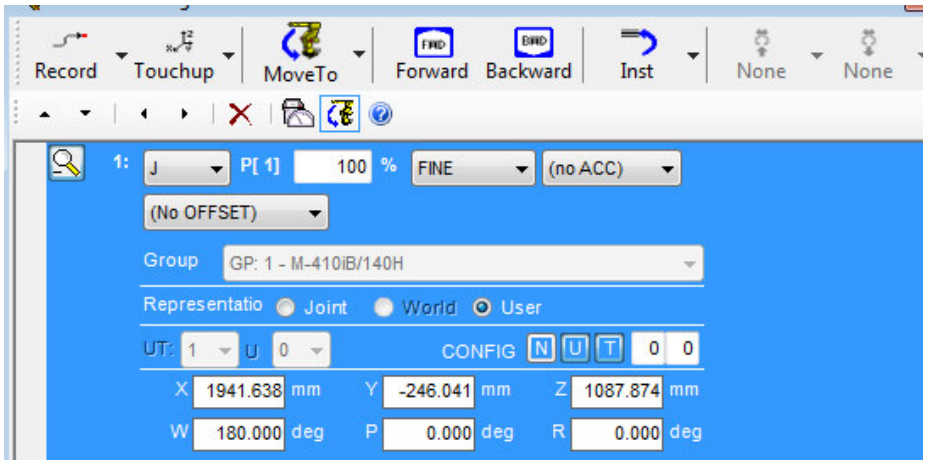
All that previous studied parts led to this very one, the part that decide the functionality of the entire process, the programming, to recall that this part in robotic has been very well discussed in the previous chapter or at least in my opinion it was.

This program that we use provides a very easy method to let us create the program, in a way that any beginner in programming could learn very fast, I did not find any difficulty about the basis, and this is because of the very easy language "KAREL". So, in a nutshell;

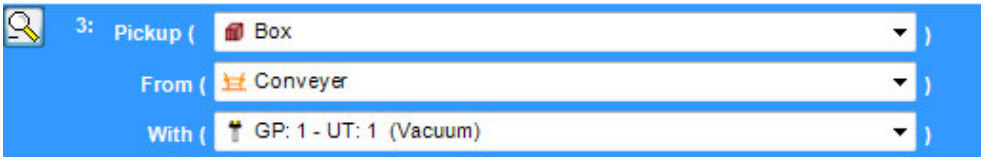
To create a simulation program, go to "Teach in the taskbar" and choose "Add Simulation Program", a window will appear as blank page, this one is going to be filled with the desired commands as follow:



- First, we have to choose a position of the robot where it should be at the rest time (before moving), to do so, click on "record list" and choose "J P […] 100% Fine".



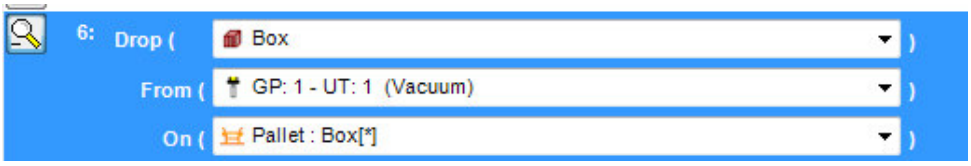
- Then, to pick the part from the conveyer, click again on “Record List” and this time choose “L P […] 2000mm/s Fine”. Then go to “MoveTo list” and choose “Box1@conveyer”, we notice the robot moved to the box at the conveyer.
- In this case, we must pick this box, to do so, go to “Inst list” and choose “Pick”, the line of pick task will open and it requires to be filled with the settings desired, and then we notice that the robot picked the part from the conveyer.



- The robot now has to go back to its first position, to do so, we need to click on the first line of the program and then on the “MoveTo button”.

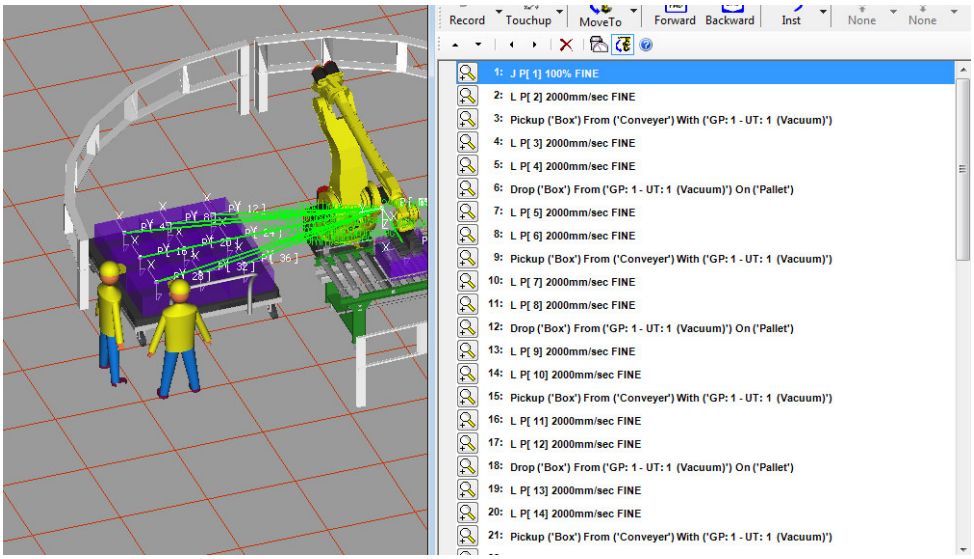
After picking the part from the conveyer we need to place it at the pallet, so we have to do as follow:

- Click on “Record List” then choose “L P […] 2000mm/s Fine”. Now go to “MoveTo” list and choose “Box[11]@pallet”, we notice the robot moved to a box at the pallet.
- Now all we have left to do is to place it, as before, we have to click first at “L P […] 2000mm/s Fine”. Then we go to “Inst list” to choose “drop”, and fill the parameters we have, by choosing the part and the fixture given, we notice here that the box has been placed.



- The robot has to come back at its initial position to repeat the task, we do that by clicking on first line and then clicking on the “MoveTo” button

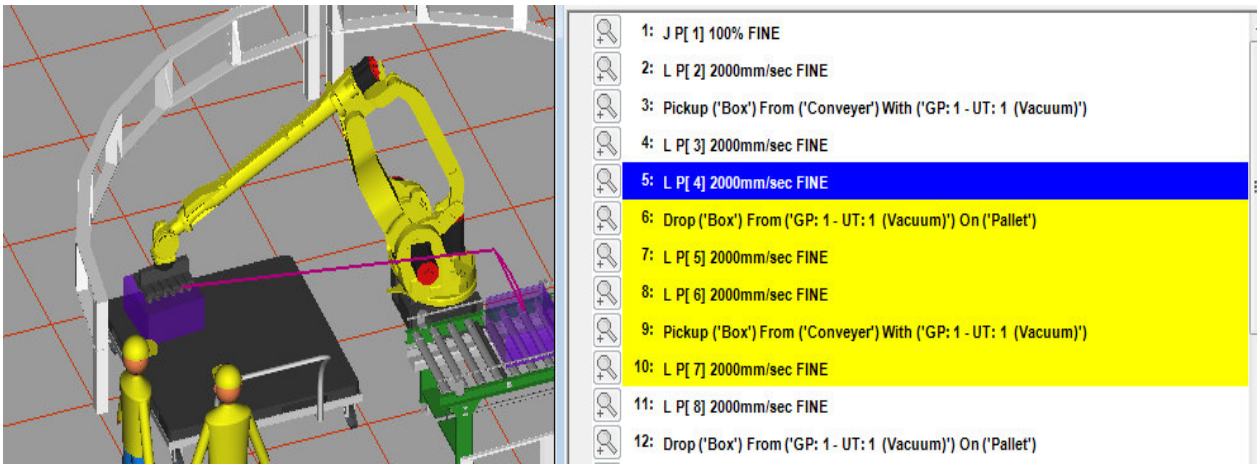
All these steps should be done repeatedly with changing of the place task configuration, each time we move to the other box as long as we have 9 boxes at the pallet, [11] [12] [13] [21] [22] [23] [31] [32] [33].



By finishing the program, we have to run it:

- Go to the bar as it is shown below
- Click on the button run.

- Now the robot is doing exactly what is commended.



Conclusion

Conclusion and future work

In this memoir, the concept of robotics is discussed, and what is related to it is studied, so do the types that exist and the applications where we need the use of the robot. The most important in robot programming is a good knowledge of its kinematics as it is said in the first chapter, but to do so, the programming needs a software that achieves it. According to what is mentioned previously, there are lot of robots in the world, and the number of robot trademarks is also growing up, most of them create their own software to program their robots.

In this memoir, we have chosen a FANUC robot that uses its own software "ROBOGUIDE". It helps to simulate a work cell as a workshop in reality, to create a robot and the tool with which we want to work, and the most necessary is to help create the program. It is said before the program can be exported to the real robot. The program in this case is done by Karel Language.

The third chapter deals with simulation, the choice of robot, the tool, the task and the environment has been taken, as it has been shown. The industrial application that we chose is palletizing using a big cylindrical robot by FANUC M410ib/140H and a vacuum tool (gripper) to do the pick and place task.

As a future work, we propose to develop the use of the robot like including the intelligent vision system 3D that leads to detection of the parts and its size. Also, working on the software to learn more about it and maybe to create new applications, but it will be better when we work on building robots that could compete with the existed ones in the markets. This is not an impossible thing to do, as long as everything about building and programming the subject is studied.

REFERENCES

[01] Ben-Zion Sandler: ROBOTICS. Designing the machines for automated machinery – 1998

[02] Tesla N: My inventions – Autobigraphy of Nicola Tesla – 1983

[03] Rosheim M: Robot evolution – the development of anthrobots – 1990

[04] Rosheim M: In the footsteps of Leonardo – IEEE Robots and Automaton magazine – 1997

[05] Enciclopedia Britanica

[06] Ming Xie: Fundemntal of robotics - linking perception to Action

[07] John J Craig: Introduction to Robotic (Mechanics and Control) – 2010

[08] C. Blume and W. Jakob: programming languages for industrial robots – 1986

[09] K. Brink: Event-Based control of industrial robot systems. Lic tech Thesis, Lund Inst. Of technology, Lund. Sweden. Department of production and materials engineering – 1996

[10] The Original paper by Denavit and Hartenberg

[11] Peter Coke: Denavit-Hartenberg Notation for common robots – 2014

[12] John W. Prest and G.T. Steven Jr.

[13] John M. Fitzgeralg

[14] Saha. S.K.: Introduction to robotics – 2008

[15] Operating Manual Introducing to RAPID – Robot Ware 5.0. – 2007

[16] Rapid Reference Manual – Overview online

[17] https://en.wikipedia.org/wiki/Karel_(programming_language)
14-05-2017

[18] http://www.onerobotics.com/posts/2013/introduction-to-karel-programming
15-05-2017

[19] http://www.fanuc.eu/uk/en/industrial-applications/industrial-palletising
17-05-2017

ANNEX

The program Simulation (Karel Language)

```
/PROG  SIMULATION
/ATTR
OWNER          = MNEDITOR;
COMMENT        = "";
PROG_SIZE      = 3684;
CREATE         = DATE 17-06-07  TIME 17:22:12;
MODIFIED       = DATE 17-06-07  TIME 17:22:14;
FILE_NAME      = ;
VERSION        = 0;
LINE_COUNT     = 77;
MEMORY_SIZE  = 4004;
PROTECT        = READ_WRITE;
TCD:  STACK_SIZE       = 0,
    TASK_PRIORITY      = 50,
    TIME_SLICE  = 0,
    BUSY_LAMP_OFF      = 0,
    ABORT_REQUEST      = 0,
    PAUSE_REQUEST      = 0;
DEFAULT_GROUP      = 1,*,*,*,*;
CONTROL_CODE       = 00000000 00000000;
/MN
  1:  !FANUC Robotics America ;
  2:  !ROBOGUIDE Generated This TPP ;
  3:  !Run SimPRO.cf to setup frame and ;
  4:  UTOOL_NUM[GP1]=1 ;
  5:  UFRAME_NUM[GP1]=0 ;
  6:J P[1] 100% FINE    ;
```

```
 7:L P[2] 2000mm/sec FINE    ;
 8:  ! Pickup ('Box') From ('Conveyer' ;
 9:  WAIT   2.00(sec) ;
10:L P[3] 2000mm/sec FINE    ;
11:L P[4] 2000mm/sec FINE    ;
12:  ! Drop ('Box') From ('GP: 1 - UT: ;
13:  WAIT   2.00(sec) ;
14:L P[5] 2000mm/sec FINE    ;
15:L P[6] 2000mm/sec FINE    ;
16:  ! Pickup ('Box') From ('Conveyer' ;
17:  WAIT   2.00(sec) ;
18:L P[7] 2000mm/sec FINE    ;
19:L P[8] 2000mm/sec FINE    ;
20:  ! Drop ('Box') From ('GP: 1 - UT: ;
21:  WAIT   2.00(sec) ;
22:L P[9] 2000mm/sec FINE    ;
23:L P[10] 2000mm/sec FINE    ;
24:  ! Pickup ('Box') From ('Conveyer' ;
25:  WAIT   2.00(sec) ;
26:L P[11] 2000mm/sec FINE    ;
27:L P[12] 2000mm/sec FINE    ;
28:  ! Drop ('Box') From ('GP: 1 - UT: ;
29:  WAIT   2.00(sec) ;
30:L P[13] 2000mm/sec FINE    ;
31:L P[14] 2000mm/sec FINE    ;
32:  ! Pickup ('Box') From ('Conveyer' ;
33:  WAIT   2.00(sec) ;
34:L P[15] 2000mm/sec FINE    ;
35:L P[16] 2000mm/sec FINE    ;
```

```
36:  ! Drop ('Box') From ('GP: 1 - UT: ;
37:  WAIT   2.00(sec) ;
38:L P[17] 2000mm/sec FINE    ;
39:L P[18] 2000mm/sec FINE    ;
40:  ! Pickup ('Box') From ('Conveyer' ;
41:  WAIT   2.00(sec) ;
42:L P[19] 2000mm/sec FINE    ;
43:L P[20] 2000mm/sec FINE    ;
44:  ! Drop ('Box') From ('GP: 1 - UT: ;
45:  WAIT   2.00(sec) ;
46:L P[21] 2000mm/sec FINE    ;
47:L P[22] 2000mm/sec FINE    ;
48:  ! Pickup ('Box') From ('Conveyer' ;
49:  WAIT   2.00(sec) ;
50:L P[23] 2000mm/sec FINE    ;
51:L P[24] 2000mm/sec FINE    ;
52:  ! Drop ('Box') From ('GP: 1 - UT: ;
53:  WAIT   2.00(sec) ;
54:L P[25] 2000mm/sec FINE    ;
55:L P[26] 2000mm/sec FINE    ;
56:  ! Pickup ('Box') From ('Conveyer' ;
57:  WAIT   2.00(sec) ;
58:L P[27] 2000mm/sec FINE    ;
59:L P[28] 2000mm/sec FINE    ;
60:  ! Drop ('Box') From ('GP: 1 - UT: ;
61:  WAIT   2.00(sec) ;
62:L P[29] 2000mm/sec FINE    ;
63:L P[30] 2000mm/sec FINE    ;
64:  ! Pickup ('Box') From ('Conveyer' ;
```

```
 65:  WAIT   2.00(sec) ;
 66:L P[31] 2000mm/sec FINE    ;
 67:L P[32] 2000mm/sec FINE    ;
 68:  ! Drop ('Box') From ('GP: 1 - UT: ;
 69:  WAIT   2.00(sec) ;
 70:L P[33] 2000mm/sec FINE    ;
 71:L P[34] 2000mm/sec FINE    ;
 72:  ! Pickup ('Box') From ('Conveyer' ;
 73:  WAIT   2.00(sec) ;
 74:L P[35] 2000mm/sec FINE    ;
 75:L P[36] 2000mm/sec FINE    ;
 76:  ! Drop ('Box') From ('GP: 1 - UT: ;
 77:  WAIT   2.00(sec) ;
/POS
P[1]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =     0.000 deg
};
P[2]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1496.750  mm,       Y =   177.740  mm,       Z =   211.410  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =   -90.000 deg
};
P[3]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
```

X = 1941.640 mm, Y = -246.040 mm, Z = 1087.870 mm,

W = 180.000 deg, P = 0.000 deg, R = 0.000 deg

};

P[4]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = -294.230 mm, Y = -2366.260 mm, Z = -149.690 mm,

W = 180.000 deg, P = 0.000 deg, R = 180.000 deg

};

P[5]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = 1941.640 mm, Y = -246.040 mm, Z = 1087.870 mm,

W = 180.000 deg, P = 0.000 deg, R = 0.000 deg

};

P[6]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = 1941.640 mm, Y = -246.040 mm, Z = 1087.870 mm,

W = 180.000 deg, P = 0.000 deg, R = 0.000 deg

};

P[7]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = 1941.640 mm, Y = -246.040 mm, Z = 1087.870 mm,

W = 180.000 deg, P = 0.000 deg, R = 0.000 deg

};

P[8]{

GP1:

```
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  -294.230  mm,       Y = -1866.260  mm,      Z =  -149.690  mm,
        W =   180.000 deg,       P =     0.000 deg,      R =   180.000 deg
};
P[9]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,      Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,      R =     0.000 deg
};
P[10]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1496.750  mm,       Y =   177.740  mm,      Z =   211.410  mm,
        W =   180.000 deg,       P =     0.000 deg,      R =   -90.000 deg
};
P[11]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,      Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,      R =     0.000 deg
};
P[12]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  -294.230  mm,       Y = -1366.260  mm,      Z =  -149.690  mm,
        W =   180.000 deg,       P =     0.000 deg,      R =   180.000 deg
};
P[13]{
```

```
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =     0.000 deg
};
P[14]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1496.750  mm,       Y =   177.740  mm,       Z =   211.410  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =   -90.000 deg
};
P[15]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =     0.000 deg
};
P[16]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =   205.770  mm,       Y = -2366.260  mm,       Z =  -149.690  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =   180.000 deg
};
P[17]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =     0.000 deg
};
```

P[18]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = 1496.750 mm, Y = 177.740 mm, Z = 211.410 mm,

W = 180.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[19]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = 1941.640 mm, Y = -246.040 mm, Z = 1087.870 mm,

W = 180.000 deg, P = 0.000 deg, R = 0.000 deg

};

P[20]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = 205.770 mm, Y = -1866.260 mm, Z = -149.690 mm,

W = 180.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[21]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = 1941.640 mm, Y = -246.040 mm, Z = 1087.870 mm,

W = 180.000 deg, P = 0.000 deg, R = 0.000 deg

};

P[22]{

GP1:

UF : 0, UT : 1, CONFIG : 'N U T, 0, 0',

X = 1496.750 mm, Y = 177.740 mm, Z = 211.410 mm,

W = 180.000 deg, P = 0.000 deg, R = -90.000 deg

```
};
P[23]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =     0.000 deg
};
P[24]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =   205.770  mm,       Y = -1366.260  mm,       Z =  -149.690  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =   180.000 deg
};
P[25]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =     0.000 deg
};
P[26]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1496.750  mm,       Y =   177.740  mm,       Z =   211.410  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =   -90.000 deg
};
P[27]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
```

```
        W =   180.000 deg,      P =     0.000 deg,      R =     0.000 deg
};
P[28]{
  GP1:
        UF : 0, UT : 1,         CONFIG : 'N U T, 0, 0',
        X =   705.770  mm,      Y = -2366.260  mm,      Z =  -149.690  mm,
        W =   180.000 deg,      P =     0.000 deg,      R =   180.000 deg
};
P[29]{
  GP1:
        UF : 0, UT : 1,         CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,      Y =  -246.040  mm,      Z =  1087.870  mm,
        W =   180.000 deg,      P =     0.000 deg,      R =     0.000 deg
};
P[30]{
  GP1:
        UF : 0, UT : 1,         CONFIG : 'N U T, 0, 0',
        X =  1496.750  mm,      Y =   177.740  mm,      Z =   211.410  mm,
        W =   180.000 deg,      P =     0.000 deg,      R =   -90.000 deg
};
P[31]{
  GP1:
        UF : 0, UT : 1,         CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,      Y =  -246.040  mm,      Z =  1087.870  mm,
        W =   180.000 deg,      P =     0.000 deg,      R =     0.000 deg
};
P[32]{
  GP1:
        UF : 0, UT : 1,         CONFIG : 'N U T, 0, 0',
```

```
        X =   705.770  mm,       Y = -1866.260  mm,       Z =  -149.690  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =   180.000 deg
};
P[33]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =     0.000 deg
};
P[34]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1496.750  mm,       Y =   177.740  mm,       Z =   211.410  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =   -90.000 deg
};
P[35]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =  1941.640  mm,       Y =  -246.040  mm,       Z =  1087.870  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =     0.000 deg
};
P[36]{
  GP1:
        UF : 0, UT : 1,          CONFIG : 'N U T, 0, 0',
        X =   705.770  mm,       Y = -1366.260  mm,       Z =  -149.690  mm,
        W =   180.000 deg,       P =     0.000 deg,       R =   180.000 deg
};
/END
```

Abstract

This memoir focuses on programming and simulating a palletizing robot, using the famous FANUC software ROBOGUIDE, this software uses KAREL language in programming. The robot that is presented is an industrial cylindrical robot with 5 joints, it consists on palletizing boxes by picking them up from a conveyor and placing them in a pallet which will be supposed to be stocked. Doing it requires an end of arm tooling, and for this application it is better to choose a vacuum gripper as I did. Then it comes the creation of the environment, using the available objects in the software (Conveyer and Pallet) however it is possible to bring any object designed in another CAD software. Finally, programming the robot and simulating the application take an important part as they are the main objectives of the memoir.

Résumé

Ce mémoire se concentre sur la programmation et la simulation d'un robot de palettisation, en utilisant le logiciel ROBOGUIDE de FANUC, ce logiciel utilise la langue de programmation KAREL. Le robot présenté en ce mémoire est un robot industriel cylindrique avec 5 articulations dont il consiste à palettiser des boites en les prenant à partir d'un convoyeur et en les plaçant dans une palette pour les transporter vers le stock. Le fait nécessite un outil que pour cette application, il est préférable de choisit une « vacuum tool » comme c'était déjà fait. Ensuite, il s'agit de la création de l'environnement du robot en utilisant les objets disponibles dans ce logiciel (convoyer et palette) mais il est aussi possible d'apporter des autres objets conçus dans des autres logiciels de CAO. Enfin, la programmation du robot et la simulation de la tache prennent la part la plus intéressante puisqu'ils sont les principaux objectifs du mémoire.

ملخص

هذه المذكرة ترتكز على برمجة ومحاكاة ذراع الية، تختص في حمولة البضائع، وذالك باستخدام برنامج الشركة FANUCالمسمى ROBOGUIDE حيث انه يستخدم لغة البرمجة KAREL، الروبوت الذي يتم تقديمه هو صناعي اسطواني ذو 5 مفاصل حيث يهتم بتحميل البضائع التي هي على شكل علب بأخذها من الناقل المتحرك ووضعها في الحمالة، لكن هذا العمل بستلزم وجود أداة لتقوم به حيث من الأفضل ان نختارVacuum tool ثم يأتي بعد ذلك انشاء محيط خلية العمل بواسطة المكونات المتوفرة في البرنامج "الناقل المتحرك والحمالة" بالرغم من إمكانية استعمال مكونات سبق انشاؤها في برامج CAD. أخيراً، برمجة الروبوت ومحاكات التطبيق ياخذان جزءاً هاما لانهما الهدفين الرئيسين لهذه المذكرة.