Politecnico Di Torino

Masters Of Science In Mechatronic Engineering

Validation by Simulation of a Pick-and-Place Robot Program Guided by a Computer Vision System

Supervisors:

Prof. Paolo Chiabert

Dr. Gianluca D'antonio

Eng.Luciano Cavallero

Author:

Mohamad Sbeity

September,2018

Contents

1 Introduction

1.1 List of Abbreviations

SME	Small and Medium Sized Enterprises
OLP	Offline Programming
TCP/IP	Transmission Control Protocol,Internet Protocol
$hdf5$	Hierarchical Data Format
X, Y, Z	Work piece position in space
α, β, γ	Rotation around X,Y,Z

1.2 Thesis outline

This work is organized as follows:

- chapter one gives an overview of the field to be studied.
- Chapter 2 discusses the background and state of the art of the field to be studied, Computer Vision guided Manipulators,Offline Programming and Motion Planning.
- Chapter 3 discusses the methodology of the presented work.
- Chapter 4 explains the implementation part.
- chapter 5 summarize results obtained.
- chapter 6 presents conclusions and future works to be done.

1.3 Overall Research Purpose

In globalization epoch robots turn out to be of great economic and technological importance within manufacturing, considering industrial ones, industries as a result of the benefits that they are capable of providing. Recently these technological developments led by industry demands drove to a noticed development of larger amounts of these distinct robots. However, the increased order for more production cycles and the demand for better quality requires the need for these robots to be more flexible and adaptable.They can perform composite grasp tasks for both static and dynamic objects fast and precisely with known environment.

But there exist limitations as no feedback information for grasp task is provided.The trajectory and path are preplanned so any change in the environment like missing object or a change in its location, the robot will not know.

Here comes Computer vision to bring feedback information to the robot about the modification of its surrounding to make it aware of it in order to adapt . For a robotic manipulator to achieve useful task it must be programmed. Industrial manipulators require a huge amount of programming to make them functional. Their controllers are very complicated, the commercial robot programming environments are typically closed systems and programming languages varies between developers.

Despite the great evolution of robot's controllers, the robot programming is made in most applications, using one of the following ways:

- Manual on-line programming;
- Off-line programming;

This essay focuses on offline programming and brings to discussion its importance for the programming of industrial manipulators with the assistance computer vision
.

1.4 Motivation

The primary motivation for this Thesis is the demonstration of the great value of simulation and its contribution in factory design as offline programming and simulation help solve Real-World issues in a safe and effective way with the help of Computer Vision that brings consciousness of the environment to the robot.

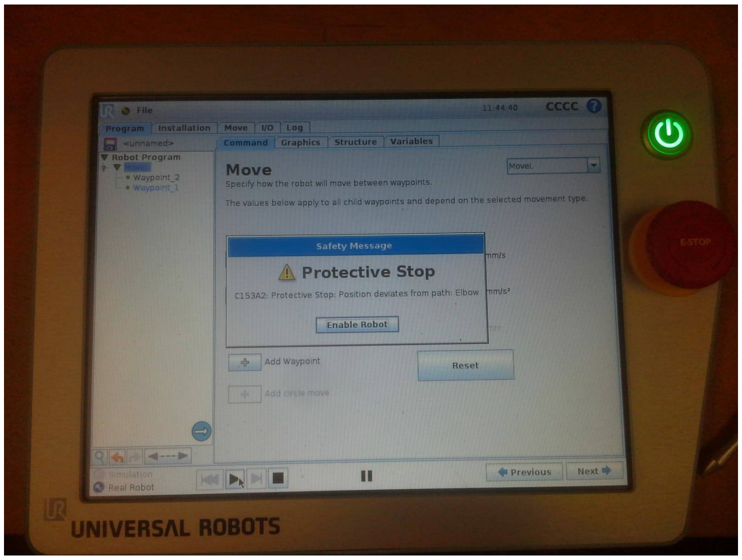
After doing an internship about simulation systems for robotic manipulators the Author had the opportunity to explore the software and understand its utilities by getting involved in the experience of manipulating robots and truly understanding their fundamental concepts.

The work was conducted between Flexcon s.r.l that provided full access to the simulation software used "Visual Components", and Politecnico Di Torino Department of Management Engineering and Production (DIGEP) which provided access to the universal robot UR3 for program validation and under the supervision of both the company and the university.

1.5 Problem Formulation

While programming our UR3 robot by manually moving the arm and teaching the controller some pick and place position respecting its limit workspace installed by (DIGEP) the department laboratory at politecnico di torino, we sometimes experienced a sudden safety stop of the robot as a result of a so called joint limit violations and singularity warning.

This occurs notably when using a Linear motion of the TCP in work space due to the necessity of maintaining a straight motion line which causes an alteration of joint velocities. Which in other words does not mean that the TCP is not able to reach that point, but simply not the way indicated.



Move

Specify how the robot will move between waypoints.

The values below apply to all child waypoints and depend on the selected movement type.

Safety Message

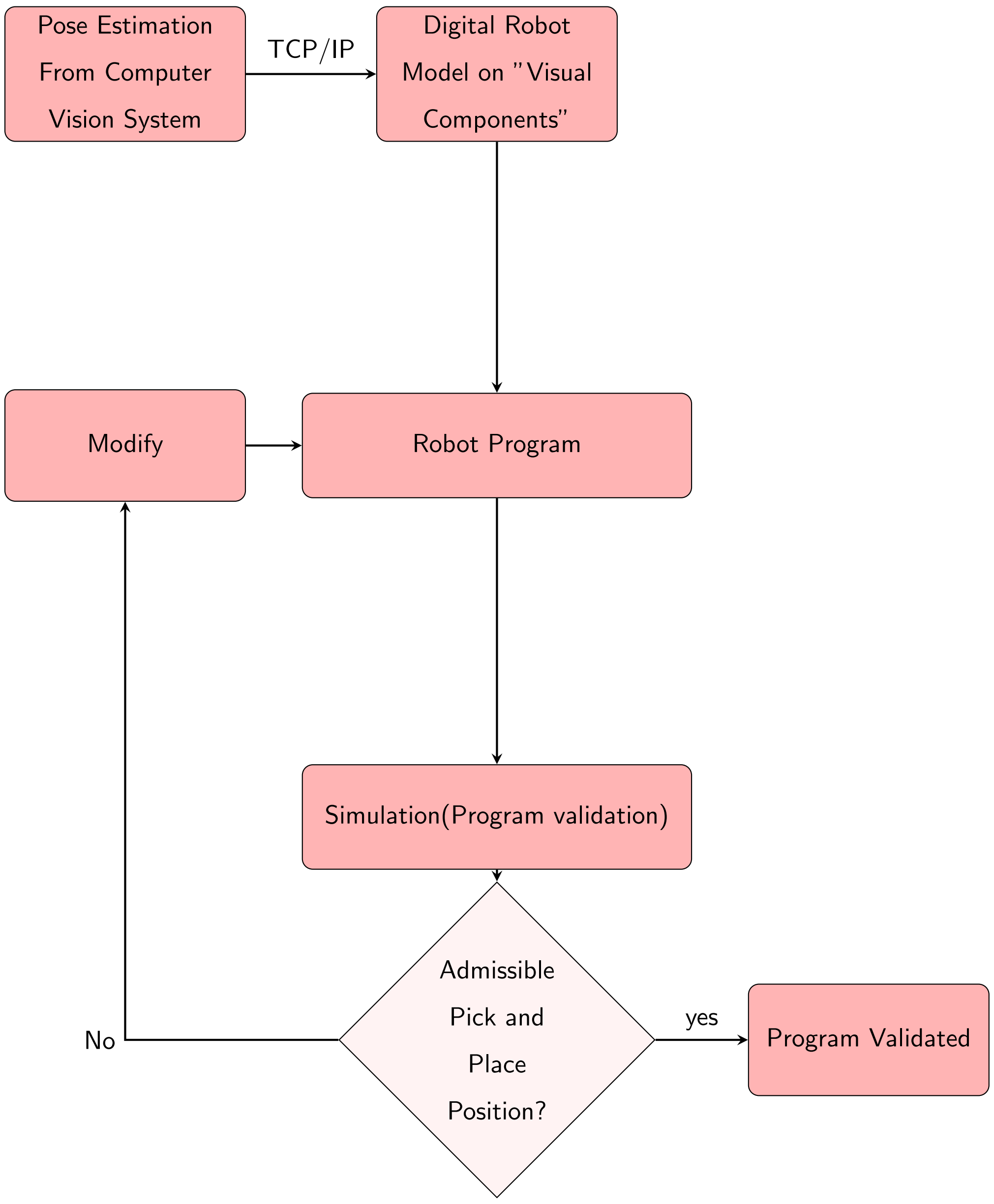
Protective Stop

C153A2: Protective Stop: Position deviates from path: Elbow

Figure 1: Error on Teach Pendant

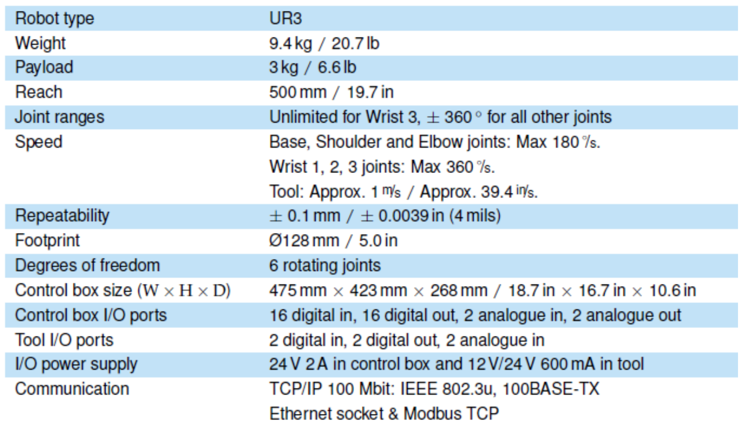
To solve this issue, the robot should be capable to change its posture, or the object should be moved. A Point-To-Point motion could be a solution for such common issue, but while performing this kind of mobility the manipulator is allowed to take curved routes which makes it less flexible in different working areas with work space limitations such as physical obstacles.

Instead of trying different robot programs on a real robotic manipulator of which some will be invalid and might lead to an anomaly in the robot's motion, the idea is to exploit "Visual Components", a robotic simulator on which we can produce an identical digital twin of the real world robot and experiment several motions like pick and place cycle of a work piece in different positions to validate obtained programs that can later be applied on real world robot.


Pose Estimation From Computer Vision System
TCP/IP
Digital Robot Model on "Visual Components"
Modify
Robot Program
Simulation(Program validation)
Admissible Pick and Place Position?
No
yes
Program Validated

1.5.1 Robot UR3

The robot used is a 6 DoF manipulator UR3 from universal robot



Robot type	UR3
Weight	9.4 kg / 20.7 lb
Payload	3 kg / 6.6 lb
Reach	500 mm / 19.7 in
Joint ranges	Unlimited for Wrist 3, $\pm$ 360 ° for all other joints
Speed	Base, Shoulder and Elbow joints: Max 180 °/s. Wrist 1, 2, 3 joints: Max 360 °/s. Tool: Approx. 1 m/s / Approx. 39.4 in/s.
Repeatability	$\pm$ 0.1 mm / $\pm$ 0.0039 in (4 mils)
Footprint	Ø128 mm / 5.0 in
Degrees of freedom	6 rotating joints
Control box size (W $\times$ H $\times$ D)	475 mm $\times$ 423 mm $\times$ 268 mm / 18.7 in $\times$ 16.7 in $\times$ 10.6 in
Control box I/O ports	16 digital in, 16 digital out, 2 analogue in, 2 analogue out
Tool I/O ports	2 digital in, 2 digital out, 2 analogue in
I/O power supply	24 V 2 A in control box and 12 V/24 V 600 mA in tool
Communication	TCP/IP 100 Mbit: IEEE 802.3u, 100BASE-TX Ethernet socket & Modbus TCP

Figure 2: UR3 Technical Specifications (From Universal Robots)

2 Literature Review

2.1 Computer Vision and Industrial Manipulators

"Computer vision is the construction of explicit, meaningful descriptions of physical objects from images as a prerequisite for recognizing, manipulating, and thinking about objects" [4].
It is an multidisciplinary field that studies how computers can be developed for understanding digital images or videos by:

- acquisition : transformation of the analog world around us into binary inputs interpreted as digital images.
- processing: Algorithm application on digital images to improve their quality by removing defects such as geometric distortion, improper focus, noise[6].
- analyzing : The process of distinguishing objects (regions of interest). from the background and producing quantitative information,
- understanding

obtained digital images in order to extract high dimensional data from the real world to produce numerical data[11].
These data are to be provided as a feedback to the robot so that it can be modified to adapt with the changing environment that its is working in.

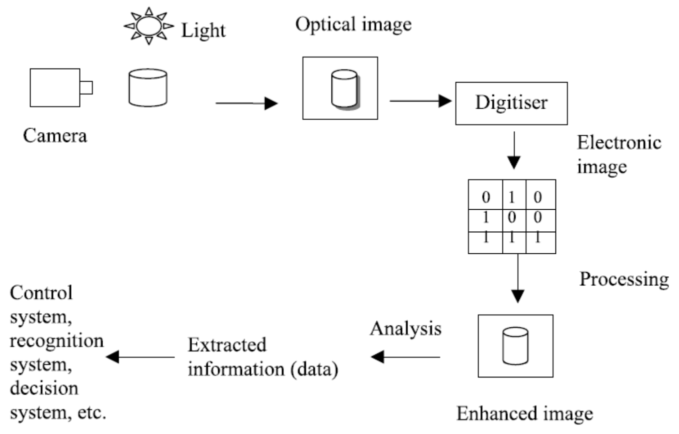


Figure 3: principle of Computer Vision System[11]

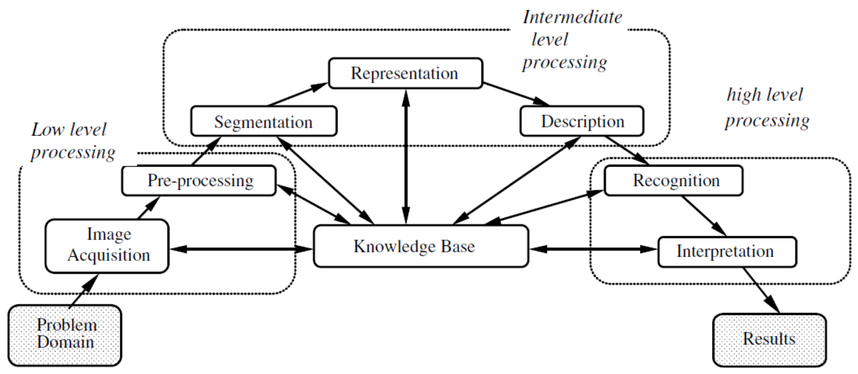


Figure 4: Different levels in image processing [6]

Vision is a very important function for human beings, Robots also require to be aware of their surroundings with which they work and interact as this environment changes continously. Computer vision systems have developed consequently and now have become basic automation component that plays a fundamental role in robot control [17].

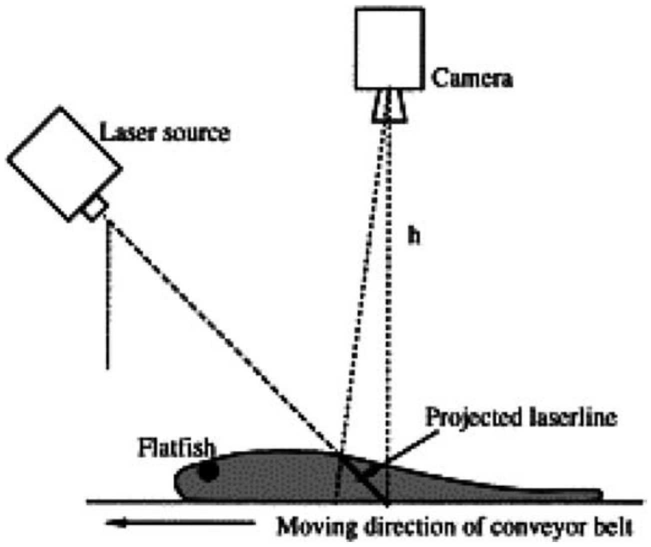


Figure 5: computer vision system for evaluating the volume of fish on a conveyor belt [6]

[13] Presents an application on an object placing planner for a gripped object during pick-and-place motion as they are most common tasks for manipulators to achieve.

This fast growing evolution of computer vision systems continuously is proving to be economically advantageous in mass production due to high skilled employment costs, it reduces manual intervention , improves safety [24], increases quality [6], and raises productivity rates [7] .

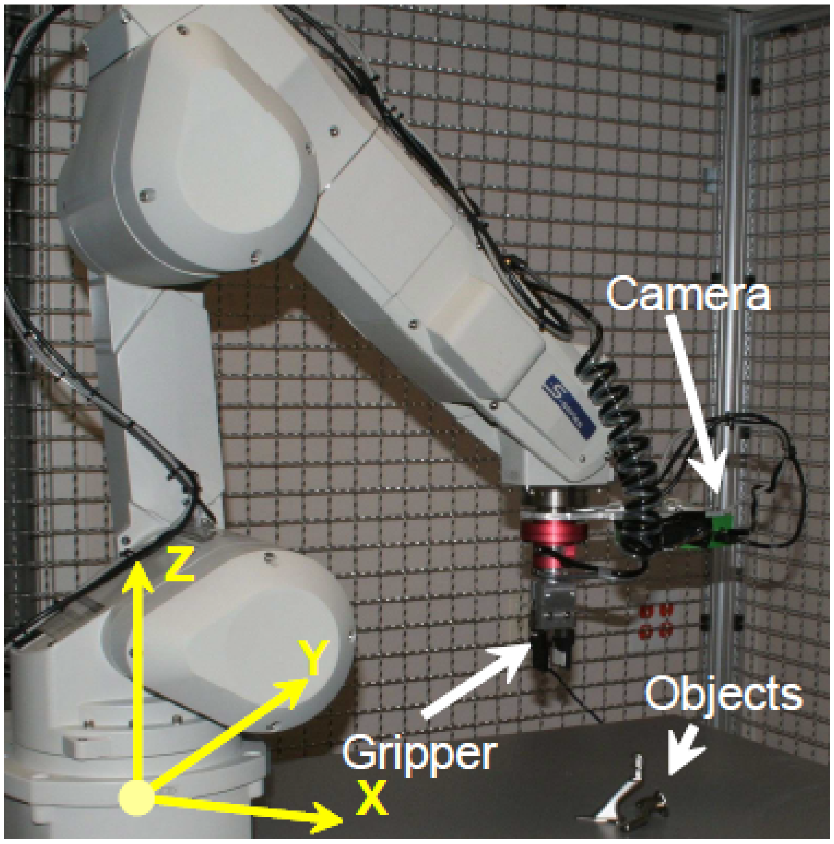


Figure 6: 6 DOF Manipulator Guided By A Vision System [2]

2.2 Robot Programming

When we think about programming, first thing that comes to our minds is endless lines of code in a computer. Despite the fact that it is a popular programming way, it still not the only one.

Figure 7: General Source Code (shutterstock.com)

At the end, every programming instruction ends up in 1s and 0s in an electronic circuit. Nowadays there are several ways to enter binary bits into the robot where some of them don't need a formal knowledge of programming.

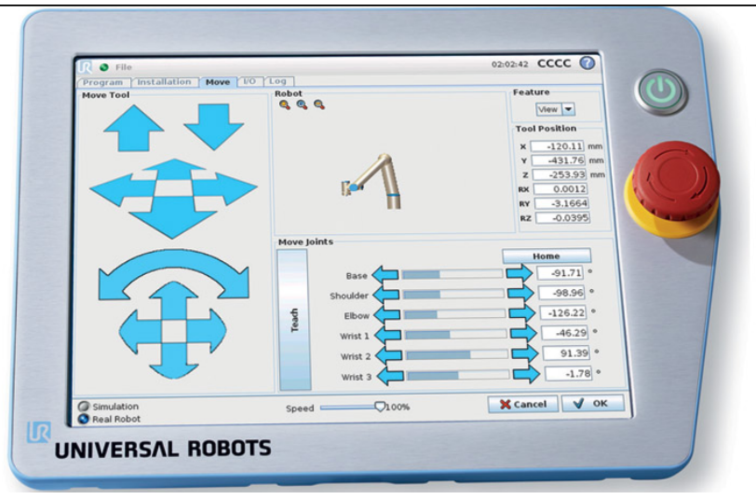
Oftentimes robot programming is a combined between on-line and off-line programming. Generally,on-line is used to teach locations in space while offline is used for task definition where sequence of operations are applied.

1.Online Programming:

Is a Manual in-situ operation performed by a human operator using the teach pendant and by moving the robot's end effector to a desired position and orientation to be later stored in the memory of the robot.

- Teaching pendant:

  Considered as the most popular method of robot programming where over 90 percent robot programmers use it.It has developed through decades and now it is more like a touch screen tablet as to suit evolving users.It is enough to use the screen buttons to move the robot around and save desired targets separately.[http://www.bara.org.uk/]



Tool Position
X -120.11 mm
Y -431.76 mm
Z -253.93 mm
RX 0.0012
RY -3.1664
RZ -0.0395

Base -91.71 °
Shoulder -98.96 °
Elbow -126.22 °
Wrist 1 -46.29 °
Wrist 2 91.39 °
Wrist 3 -1.78 °

Speed 100%

Figure 8: Teach Pendant[Universal Robots]

- Teaching by Demonstration:

  It adds intuitiveness to the classic teaching pendant method where it includes a direct interaction with the robotic arm moving it manually to desired position or by a joystick attached to the wrist above the end effector.[http://www.bara.org.uk/]

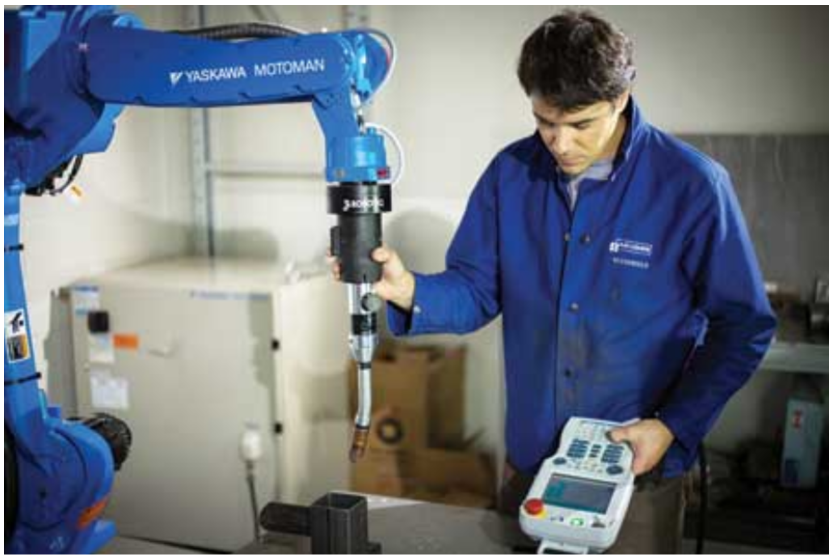


Figure 9: Teaching By Demonstration a Robotic Arm For Welding. [http://www.mfgnewsweb.com]

Although this technique is easy and doesn't recquire programming skills,it has some drawbacks to be mentioned [http://www.robotiq.com/] :

- less practical working with large robots
- Difficult to achieve straight line motion with high accuracy as for circular arcs.
- Difficult to rearrange undesired operator moves
- Difficult to merge connected sensor data
- Task synchronization with other machines or equipment in the work cell is difficult.
- It required large amount of memory
- Obtained Programs exist only in the memory of robot control system,so they are often difficult to transfer for several use.

2. Off-line programming (OLP):

OLP is a programming method where the entire programming process is carried out early during design cycle in a 3D computer modelled environment on an external pc independently from the robot cell so the programming effort is shifted from operators jogging the manipulator to software engineers.Developed program is later to be moved to the robot's PC.

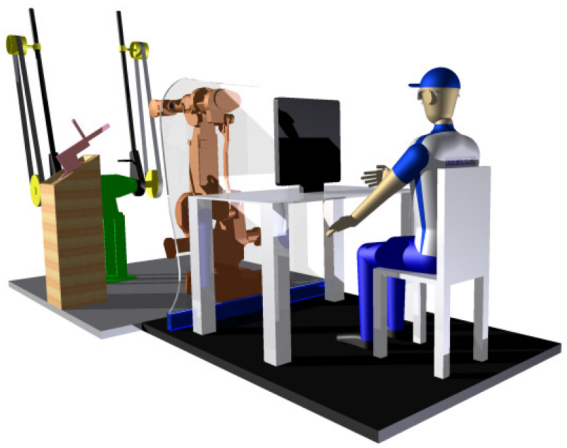


Figure 10: Off-line robot programming concept. Working in an office environment, the user can generate robot programs without interrupting production process [19]

.

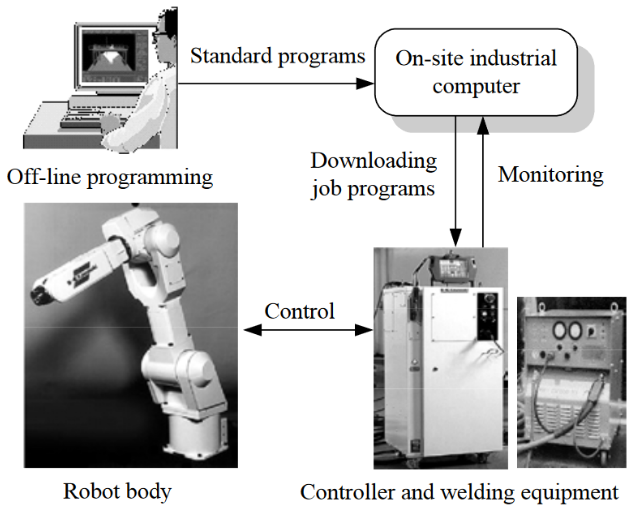


Figure 11: Offline Programming system configuration [15]

It requires a great programming effort and a long delivery time which makes it unfeasible for incorporation in SME where production rate is too small to overcome the time and high costs in using OLP software[18] .

Some various applications have been developed recently such as sheet metal bending cell where robotic cell design and robot programming are embedded in the same interface [9].

In [20] offline programming toolbox for remote laser welding (RLW) that provides a semi automated method for computing close-to-optimal robot programs with the objective of minimizing the cycle time. [21]presents an automated offline programming (AOLP) for robotic welding system. [23] presents a comprehensive review of the recent progresses regarding industrial robots programming methods and the development of OLP.

Drawbacks of offline programming[http://www.robotiq.com/]:

- Simulation models will probably never be able to imitate the real world with 100 percent precision.
- Some time waste solving simulator problems instead of solving production ones.
- Might consume more time for total development where there is extra time spent for simulation development together with testing later on the robot.

Anyhow the mentioned disadvantages are beyond the scope of this work wich discusses how can we benefit from simulation and offline programming.

2.3 Motion Planning

A fundamental robotics task is to plan collision-free motions for complex bodies from a start to a goal position among a collection of static obstacles [8]. This planning involves the study of kinematics which is defined as the transformation from joint to work space and vice-versa.

- Forward Kinematics: Is to find the position and orientation of the end effector(hand) relative to the base given the positions of all of the joints and the values of all of the geometric link parameters.

  Direct position.

  Direct velocity.

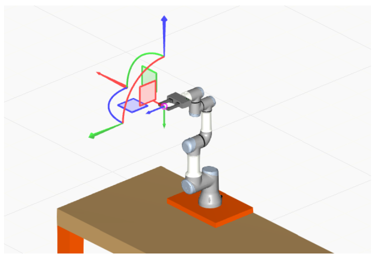


Figure 12: Tool Center Point Frame

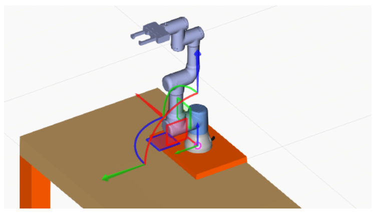


Figure 13: Manipulator Base Frame

$$\begin{bmatrix} B_{v_x} \\ B_{v_y} \\ B_{v_z} \\ B_{w_x} \\ B_{w_y} \\ B_{w_z} \end{bmatrix} = \mathsf{J(q)} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix}$$

where J(q) is a $6 \times N$ matrix called the manipulator jacobian that relates joint velocities to Cartesian velocities.

- Inverse Kinematics: is to find the values of the joint positions given the position and orientation of the end effector relative to the base and the values of all of the geometric link parameters.

Inverse position

Inverse Velocity

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} = \mathsf{J}^{-1}(q) \begin{bmatrix} B_{v_x} \\ B_{v_y} \\ B_{v_z} \\ B_{w_x} \\ B_{w_y} \\ B_{w_z} \end{bmatrix}$$

$J^{-1}(q)$ is the inverse of the jacobian matrix.

While performing different motions for different desired positions rises robots most common kinematic issue that is called singularity which can greatly complicate the programming of tool path trajectories.

It is defined as a configuration in which there is a loss of number of DOFs which means that the mobility of the manipulator is reduced. Mathematically speaking , in such configurations the inverse of the jacobian does not exist and so the joint values will tend to infinity which exceeds their physical limits [1].

Optimal motion planning is very important to the operation of robot manipulators. Its main target is the generation of a trajectory from start to goal that satisfies objectives, such as minimizing path traveling distance [16] or time interval, lowest energy consumption [25] or obstacle avoidance [25] and satisfying the robot's kinematics and dynamics.

An effective trajectory planning algorithm for such manipulators is discussed in [14], given the end effector trajectory in Cartesian space together with relevant constraints and task specifications considering joint acceleration limits and end effector velocity limits.

While it is very crucial to avoid singularities [12] introduces a method to avoid wrist and elbow singularities in a redundant robotic arm. [10] propose a control strategy for dealing with singularities and joint limits.

3 Validation By Simulation Of a Pick and Place Robot Program

3.1 "Visual Components" Simulator

The simulation software used is Visual Components that provides offline programming facilities together with work area layout composed of ready made components with the possibility of creation of new desired components even robots either modelled in several CAD modelling software or in Visual Components, on which we can generate and validate desired programs.
Generated programs that are written via python script can be used for other manipulators with just a slight change in the code which makes this property of great benefits being flexible and time saving.

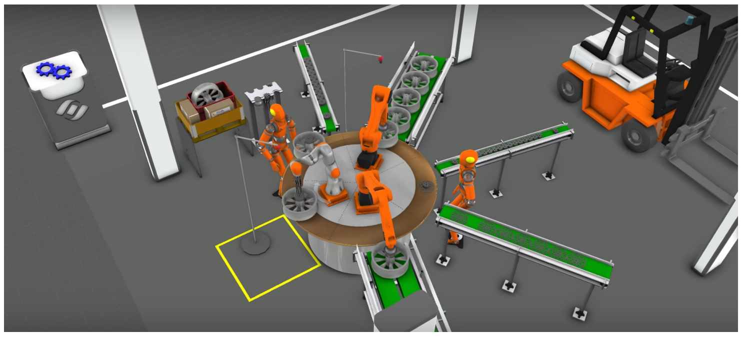


Figure 14: Layout Example

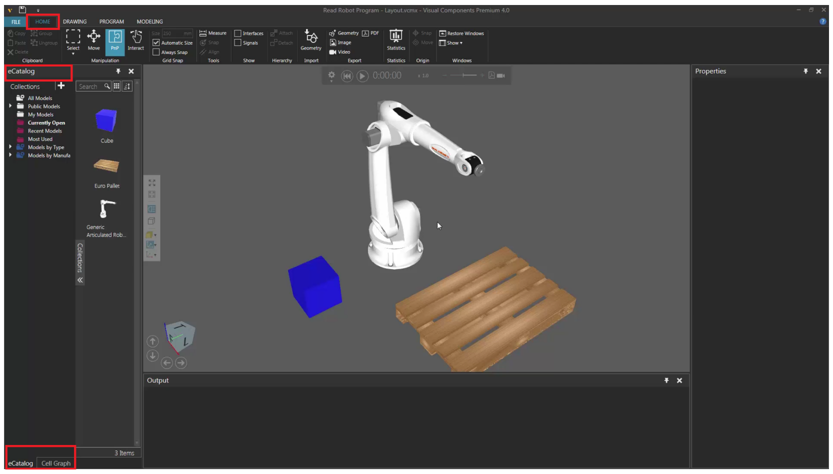


Figure 15: Home Tab

In figure 17 we can see visual components user interface where we have the 3d work space area,under this area we have the output space where various errors or desired components,matrix and other messages will be printed.
On the left bottom we have ecatalog where we can choose different components for our model and simply drag them into wrok space,they are devided into categories like manufacturer and type of component.
Near ecatalog we have cell graph in which we can see different components that our model is composed of.

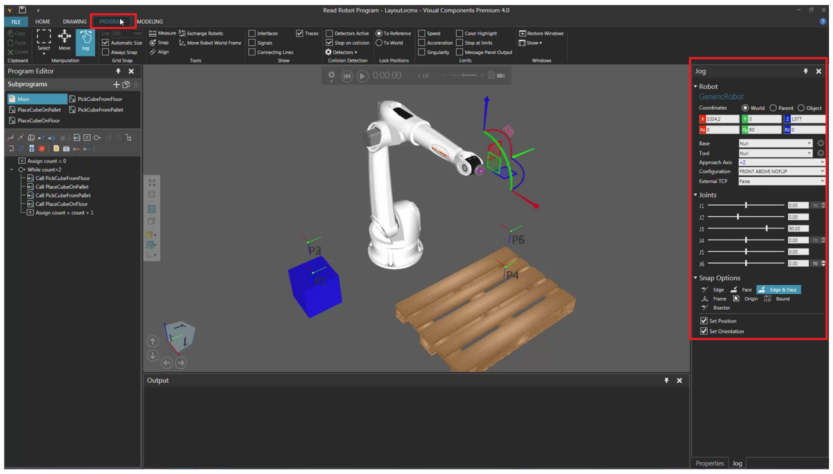


Figure 16: Program Tab

On program tab we can jog our robot manually and teach it some positions,on the right we can monitor joint values and different informations about the robot like its position in the work space,configuration,etc.

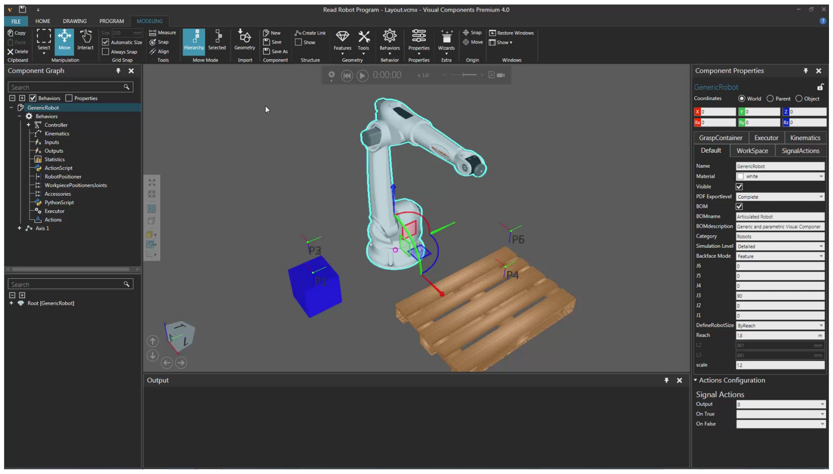


Figure 17: Modelling Tab

Modelling tab is where we can add behaviours and properties to our components that we can see on the left under component graph.

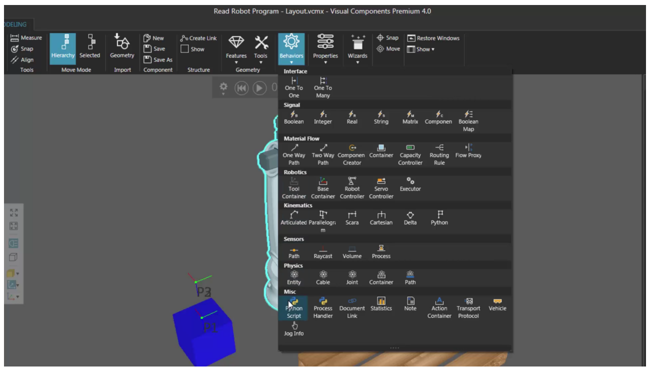


Figure 18: Adding a Python Script From Behaviors Ribbon

where we added a python script to develop our program for this study.

3.2 Program Vlidation

A virtual model composed of the bench, a universal robot UR3 connected via TCP/IP to a Computer vision system for pose estimation and the work piece to be picked and placed.

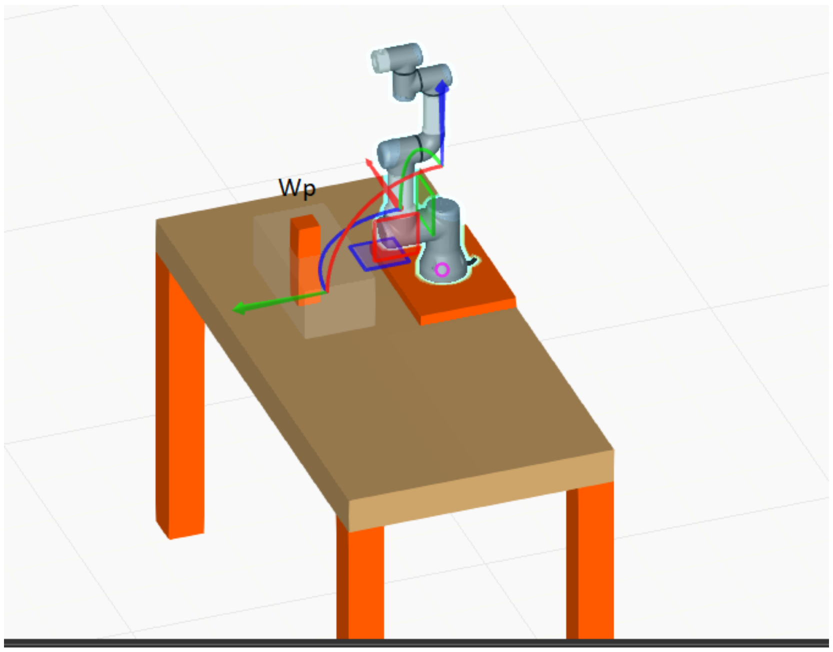


Figure 19: Virtual Model

A Program is developed as a python script for the purpose of a pick and place cycle of a workpiece composed of the following motions:

Approach pick position

Pick Position

Retreat pick position

Approach place position

Place position

Retreat place position

The program receives the position of the work piece from a computer vision system via TCP/IP connection.Then it checks whether each motion is admissible by any of eight possible configurations:

FRONT ABOVE NOFLIP

FRONT ABOVE FLIPPED

FRONT BELOW NOFLIP

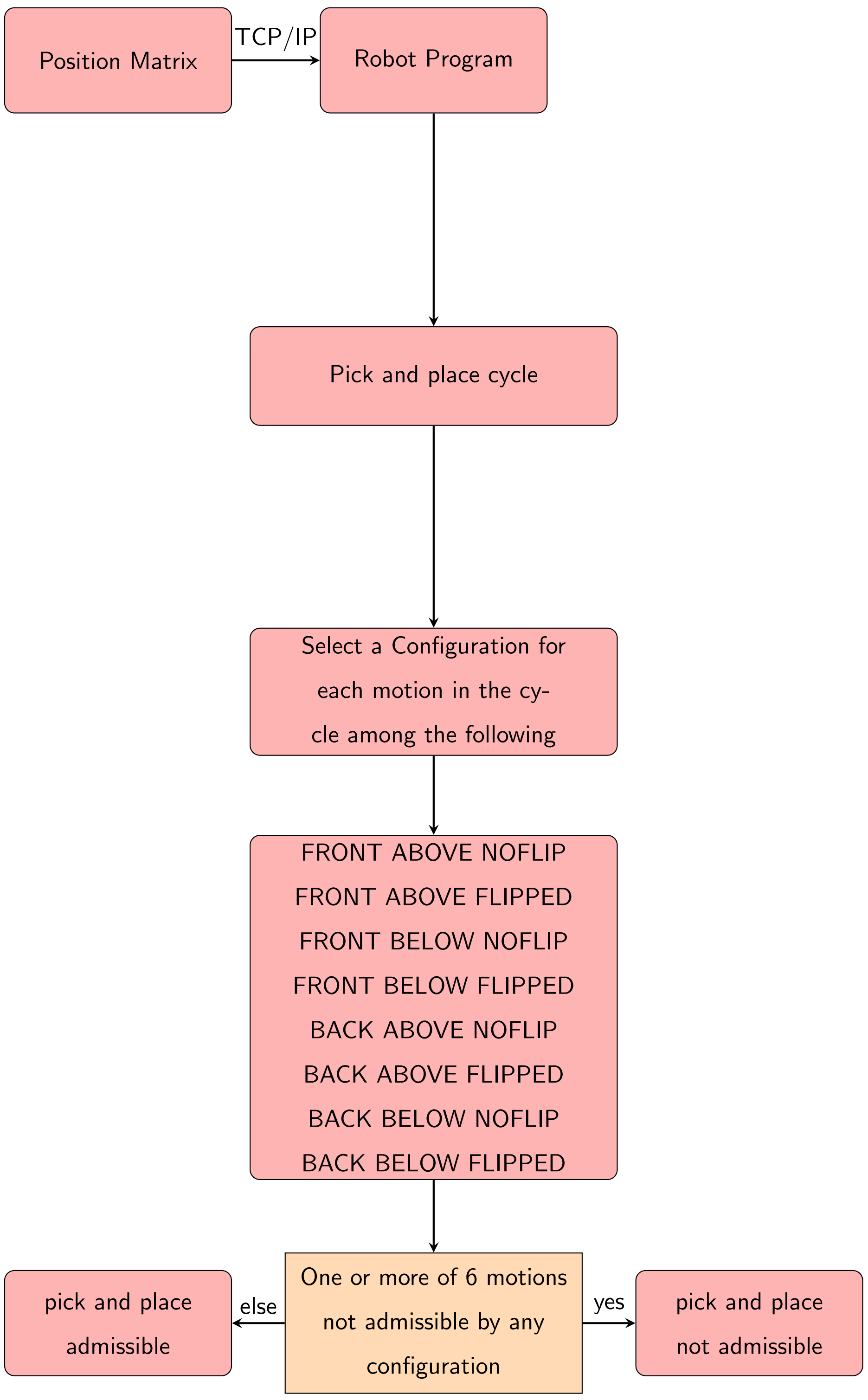
FRONT BELOW FLIPPED

BACK ABOVE NOFLIP

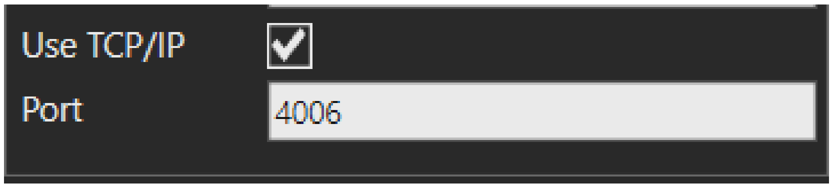
BACK ABOVE FLIPPED

BACK BELOW NOFLIP

BACK BELOW FLIPPED


Position Matrix
TCP/IP
Robot Program
Pick and place cycle
Select a Configuration for each motion in the cycle among the following
FRONT ABOVE NOFLIP
FRONT ABOVE FLIPPED
FRONT BELOW NOFLIP
FRONT BELOW FLIPPED
BACK ABOVE NOFLIP
BACK ABOVE FLIPPED
BACK BELOW NOFLIP
BACK BELOW FLIPPED
One or more of 6 motions not admissible by any configuration
else
pick and place admissible
yes
pick and place not admissible

On Home and in robot properties we enable TCP/IP connection specifying the port for the connection with the computer vision system.

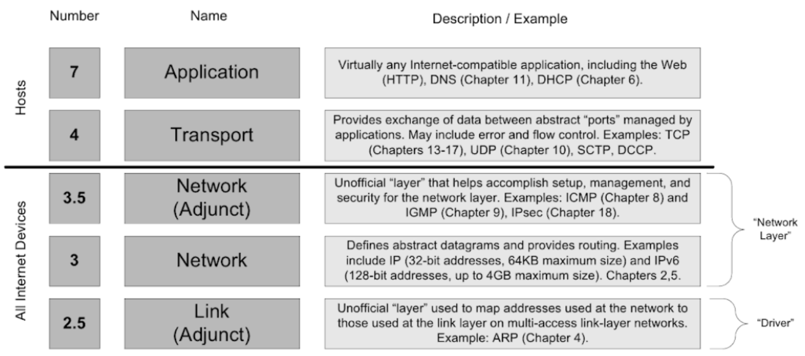


Use TCP/IP ✓
Port 4006

Figure 20: TCP/IP selection

Where TCP/IP is a protocol with which different computers and embedded systems can comunicate as it is fundamental for modern comunication. The most universal API for these networks is called SOCKETS[100].

Ports themselves are abstract positive numbers composed of 16 bits that ranges between 0 and 65535. Figure 12 discribes layering reference model which is inspired by ARPANET and adopted by TCP/IP suite.

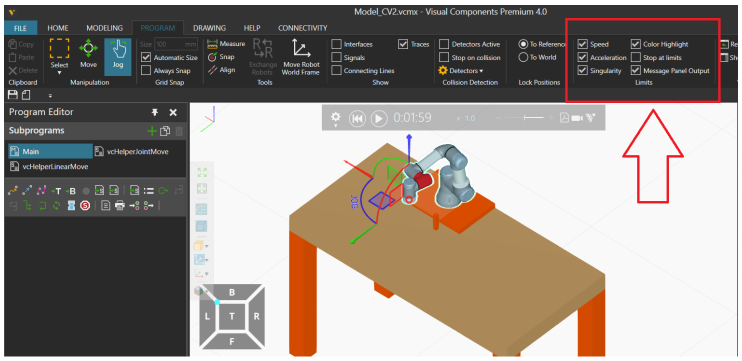


	Number	Name	Description / Example	
Hosts	7	Application	Virtually any Internet-compatible application, including the Web (HTTP), DNS (Chapter 11), DHCP (Chapter 6).	
Hosts	4	Transport	Provides exchange of data between abstract "ports" managed by applications. May include error and flow control. Examples: TCP (Chapters 13-17), UDP (Chapter 10), SCTP, DCCP.	
All Internet Devices	3.5	Network (Adjunct)	Unofficial "layer" that helps accomplish setup, management, and security for the network layer. Examples: ICMP (Chapter 8) and IGMP (Chapter 9), IPsec (Chapter 18).	"Network Layer"
All Internet Devices	3	Network	Defines abstract datagrams and provides routing. Examples include IP (32-bit addresses, 64KB maximum size) and IPv6 (128-bit addresses, up to 4GB maximum size). Chapters 2,5.	"Network Layer"
All Internet Devices	2.5	Link (Adjunct)	Unofficial "layer" used to map addresses used at the network to those used at the link layer on multi-access link-layer networks. Example: ARP (Chapter 4).	"Driver"

Figure 21: Layering Reference Model[100]

For some different picking positions and while the robot is performing a Linear motion we notice some joint limit violations such as speed and acceleration for several joints depending on the position.

Under Program commandd it is possible to enable Speed,Acceleration and Singularity limits. Even colors so that we can monitor which joint violates its limits as we can see in figure 13.



Limits: [✓ Speed] [✓ Acceleration] [✓ Singularity] [✓ Color Highlight] Stop at limits [✓ Message Panel Output]

Figure 22: Robot Stops At Joint Limits

Here is a graph that tracks joint values and saves them so we can check for any encountered violation and further monitoring purposes.

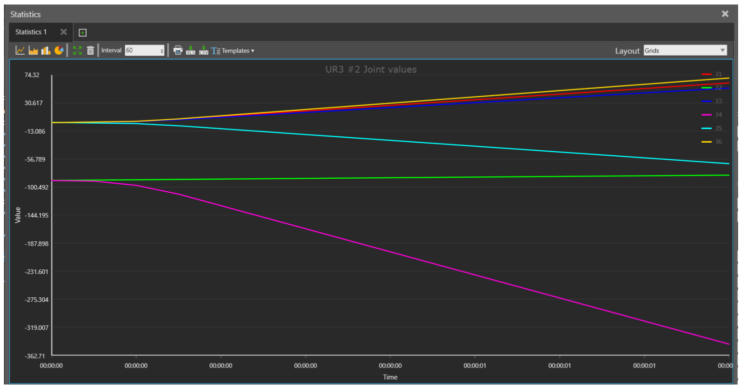
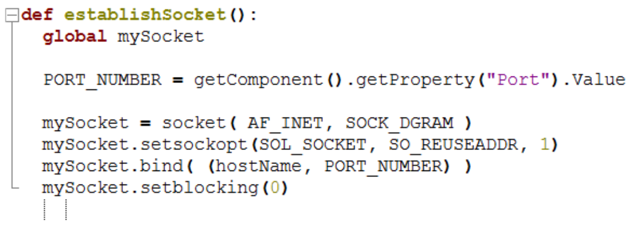


Figure 23: Joints Values Statistics

First of all, for the connection between the computer vision and the robot a Socket is created:

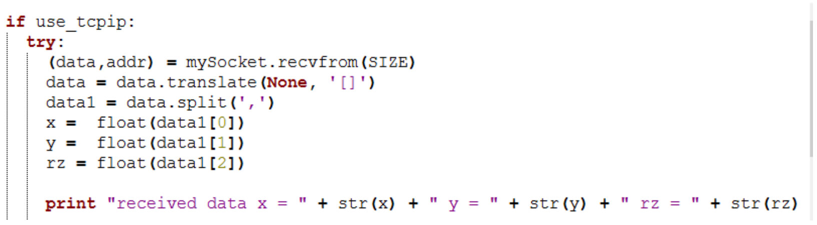

```
def establishSocket():
  global mySocket

  PORT_NUMBER = getComponent().getProperty("Port").Value

  mySocket = socket( AF_INET, SOCK_DGRAM )
  mySocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
  mySocket.bind( (hostName, PORT_NUMBER) )
  mySocket.setblocking(0)
```

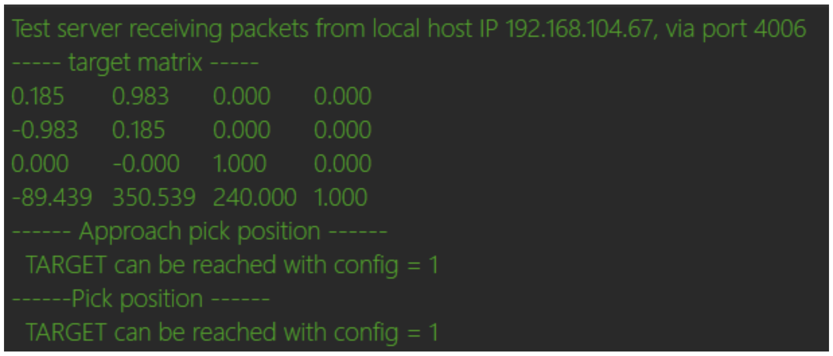
Figure 24: Socket

Then the position of the workpiece is received asa a matrix via TCP/IP.


```
if use_tcpip:
  try:
    (data,addr) = mySocket.recvfrom(SIZE)
    data = data.translate(None, '[]')
    data1 = data.split(',')
    x =  float(data1[0])
    y =  float(data1[1])
    rz = float(data1[2])

    print "received data x = " + str(x) + " y = " + str(y) + " rz = " + str(rz)
```

Figure 25: Receive a matrix via TCP/IP



```
Test server receiving packets from local host IP 192.168.104.67, via port 4006
----- target matrix -----
0.185      0.983      0.000      0.000
-0.983     0.185      0.000      0.000
0.000      -0.000     1.000      0.000
-89.439   350.539   240.000   1.000
------ Approach pick position ------
  TARGET can be reached with config = 1
------Pick position ------
  TARGET can be reached with config = 1
```

Figure 26: Receive a matrix via TCP/IP

Using universal inverse robots kinematic solver (Controller CB3) the fixed place position and the variable position of work piece is calculated.

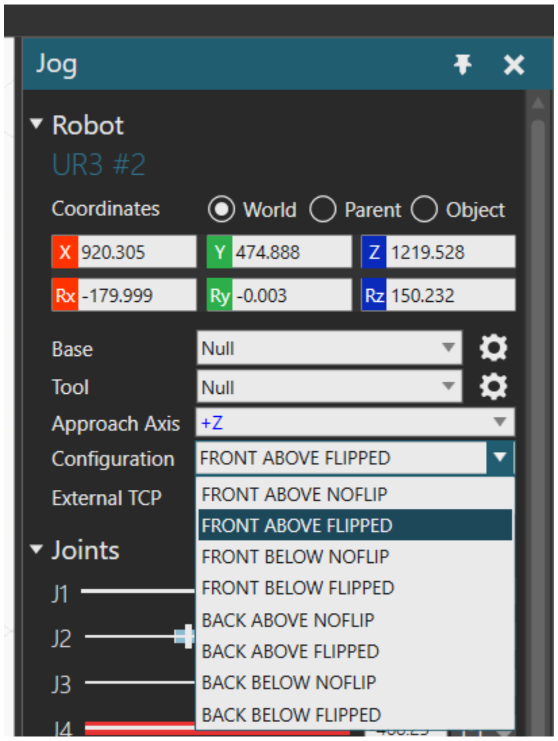
$$T_{wp}^{cvrf} = T_{r}^{cvrf} \times T_{wp}^{r}$$

- T_{wp}^{cvrf} : Transformation matrix of the work piece with respect to the Computer Vision Reference Frame.

- T_{r}^{cvrf} : Inverse Transformation matrix of the Computer Vision Reference with respect to the robot Reference Frame.

- T_{wp}^{r} : Transformation matrix of the work piece with respect to the robot Reference Frame.

In order for the robot to pick the work piece it can do that by eight different configurations.



Coordinates ● World ○ Parent ○ Object

X 920.305 Y 474.888 Z 1219.528

Rx -179.999 Ry -0.003 Rz 150.232

Base Null

Tool Null

Approach Axis +Z

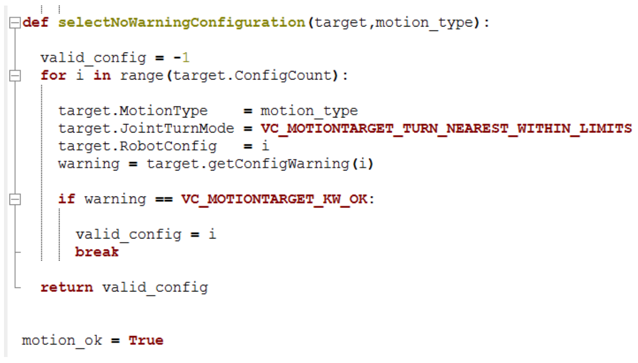
Configuration FRONT ABOVE FLIPPED

External TCP

FRONT ABOVE NOFLIP
FRONT ABOVE FLIPPED
FRONT BELOW NOFLIP
FRONT BELOW FLIPPED
BACK ABOVE NOFLIP
BACK ABOVE FLIPPED
BACK BELOW NOFLIP
BACK BELOW FLIPPED

Figure 27: Robot Configurations

We select among the configurations that we have:



```
def selectNoWarningConfiguration(target,motion_type):

  valid_config = -1
  for i in range(target.ConfigCount):

    target.MotionType     = motion_type
    target.JointTurnMode  = VC_MOTIONTARGET_TURN_NEAREST_WITHIN_LIMITS
    target.RobotConfig    = i
    warning = target.getConfigWarning(i)

    if warning == VC_MOTIONTARGET_KW_OK:

      valid_config = i
      break

  return valid_config


motion_ok = True
```

Figure 28: Select no warning configuration

For the pick and place cycle to be achieved the robot follows the following motions:

Approach pick position (Joint Motion)

Pick Position (Linear Motion)

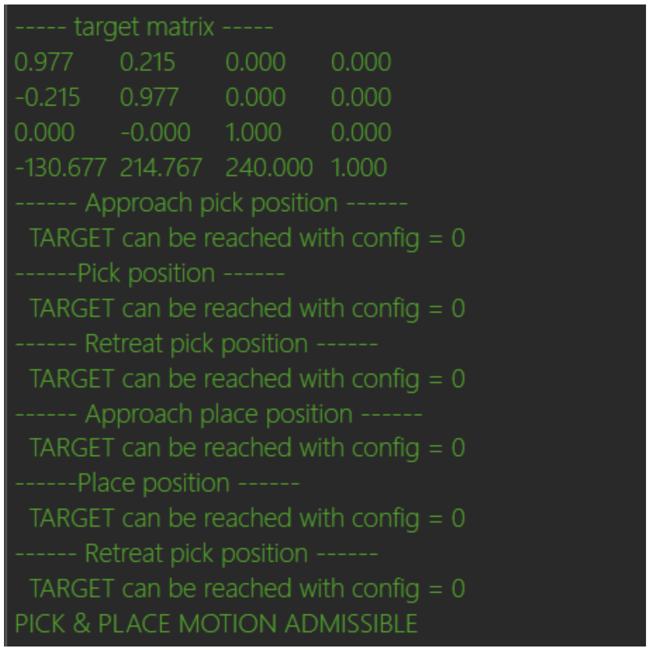
Retreat pick position (Linear Motion)

Approach place position (Joint Motion)

Place position (Linear Motion)

Retreat place position (Linear Motion)

An admissible position of the work piece by the shown configurations.



```
----- target matrix -----
0.977     0.215     0.000     0.000
-0.215    0.977     0.000     0.000
0.000     -0.000    1.000     0.000
-130.677  214.767   240.000   1.000
------ Approach pick position ------
  TARGET can be reached with config = 0
------Pick position ------
  TARGET can be reached with config = 0
------ Retreat pick position ------
  TARGET can be reached with config = 0
------ Approach place position ------
  TARGET can be reached with config = 0
------Place position ------
  TARGET can be reached with config = 0
------ Retreat pick position ------
  TARGET can be reached with config = 0
PICK & PLACE MOTION ADMISSIBLE
```

Figure 29: Admissible work piece position

Where :

(config = 0) refers to FRONT ABOVE NOFLIP

(config = 1) refers to FRONT ABOVE FLIPPED

(config = 2) refers to FRONT BELOW NOFLIP

(config = 3) refers to FRONT BELOW FLIPPED

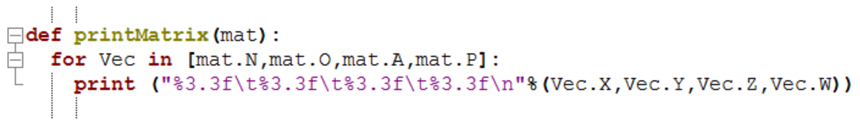
(config = 4) refers to BACK ABOVE NOFLIP

(config = 5) refers to BACK ABOVE FLIPPED

(config = 6) refers to BACK BELOW NOFLIP

(config = 7) refers to BACK BELOW FLIPPED

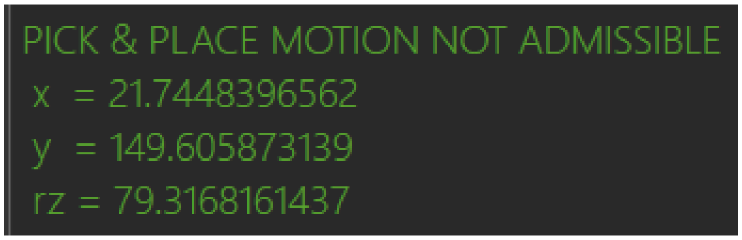
Print matrix function to be called in the program whenever we want to print a matrix on the output.



```
def printMatrix(mat):
    for Vec in [mat.N,mat.O,mat.A,mat.P]:
        print ("%3.3f\t%3.3f\t%3.3f\t%3.3f\n"%(Vec.X,Vec.Y,Vec.Z,Vec.W))
```

Figure 30: Print Matrix Function

When we encounter a non admissible work piece position we get the co-ordinates displayed on the output:



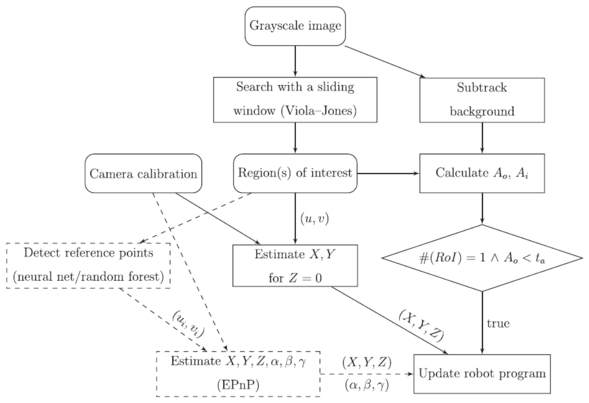
```
PICK & PLACE MOTION NOT ADMISSIBLE
 x  = 21.7448396562
 y  = 149.605873139
 rz = 79.3168161437
```

Figure 31: Non Admissible work piece position

4 Implementation

4.1 Computer Vision System

The computer vision system for pose estimation is taken from a case of study [3] that can recognize and localize a reflective work piece, and allows for automatic adjustments of the robot program.



Grayscale image
Search with a sliding window (Viola–Jones)
Subtrack background
Camera calibration
Region(s) of interest
Calculate A_o, A_i
(u, v)
Detect reference points (neural net/random forest)
Estimate X, Y for $Z = 0$
$\#(RoI) = 1 \wedge A_o < t_a$
(X, Y, Z)
true
(u_i, v_i)
Estimate $X, Y, Z, \alpha, \beta, \gamma$ (EPnP)
(X, Y, Z)
(α, β, γ)
Update robot program

Figure 32: Work flow for Work piece detection [3]

Ai: the area of the foreground objects within every detected bounding box i.

Ao: the area of the foreground objects outside of the detected bounding boxes.

4.1.1 Work Piece-auto-label

This project allows to indicate the work piece and its reference points on a first frame, and allows to track it throughout the entire video sequence, and automatically build a training or validation set for work piece recognition.

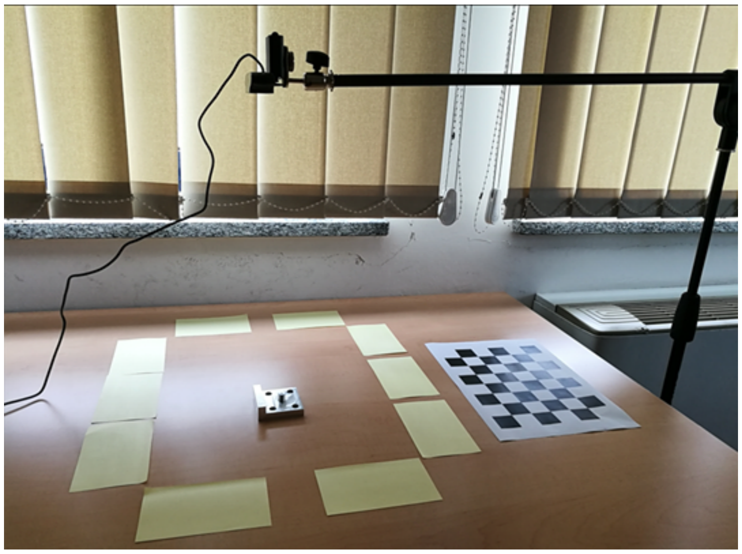


Figure 33: USB Camera Logitech

For this purpose a logitech usb camera is used for video registration. We used our video for the acquisition of 2400 different frames of the work piece where there are 5 points of interest in each frame:
Top left-corner,Top right-corner,Bottom left-corner,Bottom right-corner,Center.
Of these obtained frames about 80 percent will be used for Training and 20 percent for testing.

4.1.2 Annotate program

Annotate program provides a user interface to annotate a video sequence.It uses template matching (cross-correlation to "guess" annotations for the next frame).On every frame the region of interest and 5 points of interest have to be indicated.

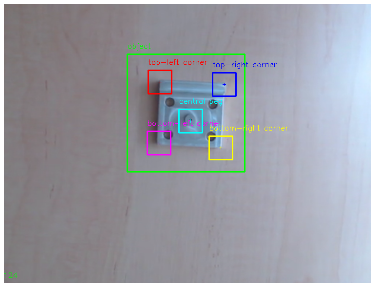


Figure 34: Points of interest in frame 124

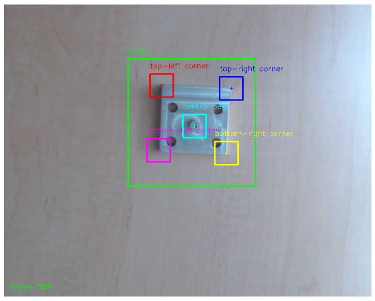


Figure 35: Points of interest in frame 568

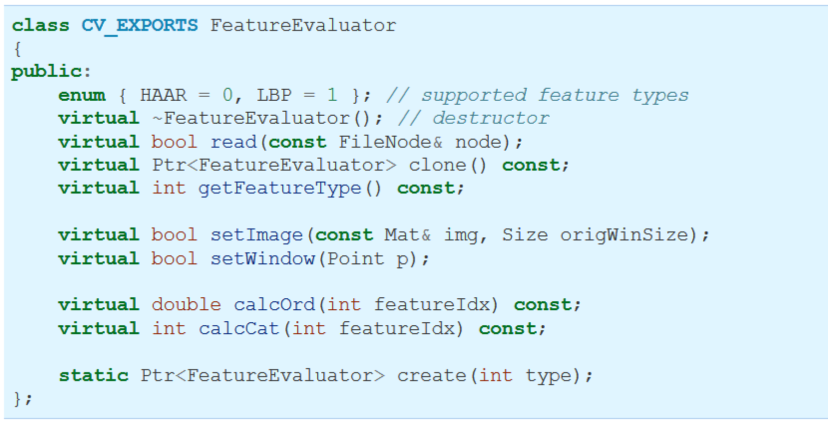
4.1.3 Train Program

Train program Calibrate camera, convert images to hdf5 data sets, train regression methods to predict locations of the reference points. The input data can be prepared with work piece-auto-label.

In order for this to be achieved, two methods were used :

- Cascade Classification :

  First a classifier is trained with a few hundred sample views of a particular object which is the work piece in our case that are scaled to the same size. After a classifier is trained, it can be applied to a region of interest of the same size as used during the training in an input image[5].



```
class CV_EXPORTS FeatureEvaluator
{
public:
    enum { HAAR = 0, LBP = 1 }; // supported feature types
    virtual ~FeatureEvaluator(); // destructor
    virtual bool read(const FileNode& node);
    virtual Ptr<FeatureEvaluator> clone() const;
    virtual int getFeatureType() const;

    virtual bool setImage(const Mat& img, Size origWinSize);
    virtual bool setWindow(Point p);

    virtual double calcOrd(int featureIdx) const;
    virtual int calcCat(int featureIdx) const;

    static Ptr<FeatureEvaluator> create(int type);
};
```

Figure 36: Base class for computing feature values in cascade classifiers[5]

- Decision Trees :

  Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features[22].

Trees.png Trees.png

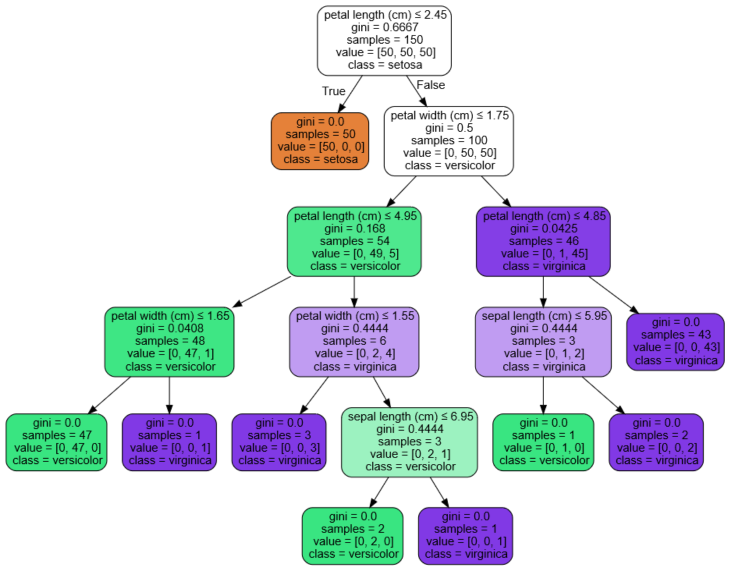
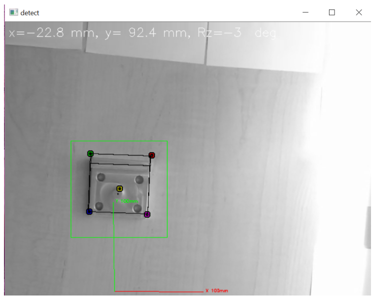


Figure 37: Base class for computing feature values in cascade classifiers[22]

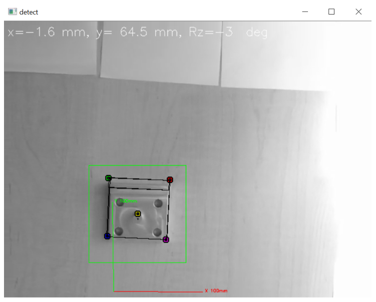
4.1.4 Detect Program

Detect program apply results of the train program to a live or recorded video stream. Then it calculates the position of the work piece for each of the obtained frames.



```
x=−22.8 mm, y= 92.4 mm, Rz=−3  deg
```

Figure 38: Work piece Position



```
x=−1.6 mm, y= 64.5 mm, Rz=−3  deg
```

Figure 39: Work piece Position

On the top of the detect program display we get the X,Y position and the Rz rotation of the work piece which will be later transferred to the robot system via TCP/IP to predict the robot's ability to pick or place the workpiece by the program developed on our simulation interface .

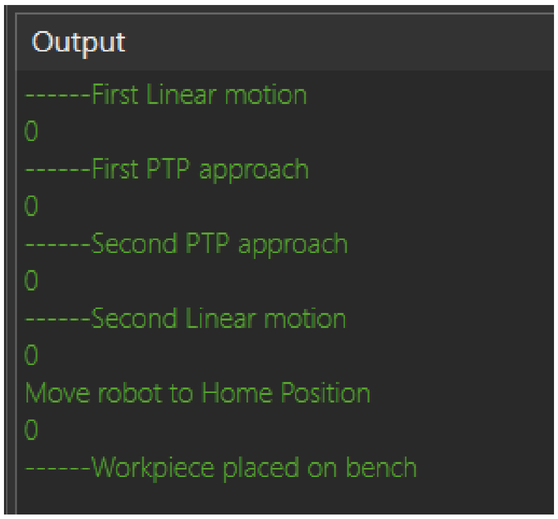
5 Results

During simulation and after trying several pick and place cycles,we registered a variation of admissible and non admissible positions that we can see in the table below:

	Worpiece Coordinates(mm)		
	X	Y	Z
Possible(confirmed both LINEAR and PTP)	716.073	339.029	945
Violate limits	736.073	339.029	945
Possible(Configwarning)	594.202	471.513	945
Violate limits	795.777	376.573	945

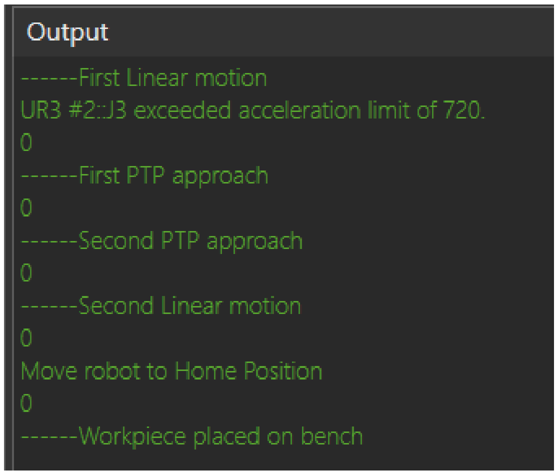
Table 1: Some Admissible and non-Admissible positions of the workpiece

Following figures shows in the output space the result of each cycle:



```
Output
------First Linear motion
0
------First PTP approach
0
------Second PTP approach
0
------Second Linear motion
0
Move robot to Home Position
0
------Workpiece placed on bench
```

Figure 40: 1st Position



```
Output
------First Linear motion
UR3 #2::J3 exceeded acceleration limit of 720.
0
------First PTP approach
0
------Second PTP approach
0
------Second Linear motion
0
Move robot to Home Position
0
------Workpiece placed on bench
```

Figure 41: 2nd Position

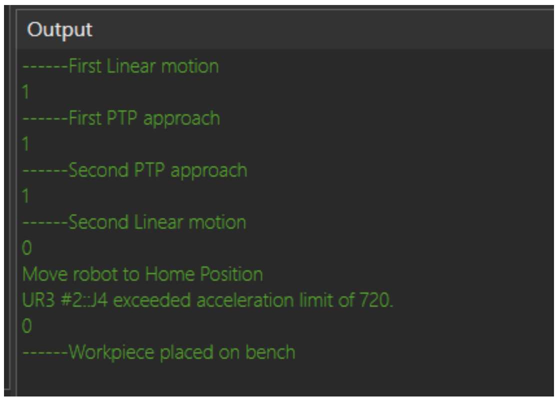

```
Output
------First Linear motion
1
------First PTP approach
1
------Second PTP approach
1
------Second Linear motion
0
Move robot to Home Position
UR3 #2::J4 exceeded acceleration limit of 720.
0
------Workpiece placed on bench
```

Figure 42: 3rd Position

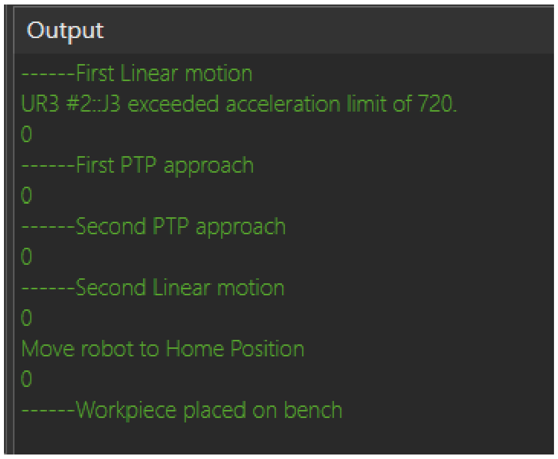

```
Output
------First Linear motion
UR3 #2::J3 exceeded acceleration limit of 720.
0
------First PTP approach
0
------Second PTP approach
0
------Second Linear motion
0
Move robot to Home Position
0
------Workpiece placed on bench
```

Figure 43: 4th Position

In this thesis a we presented an application of computer vision-based control for a universal robot on a simulation software which forms a closed loop control for the purpose of the prediction of the ability of the robot to pick and place the workpiece in the obtained position.

We created a pick and place robot program that recieves the position of the workpiece from a computer vision system, a work piece-auto-label project that allows to indicate workpiece and its reference points through a video sequence to later build a validation set for work piece recognition.

Afterwards the program decides whether this position is admissible by the robot or not before the robot starts to move towards the object in order to avoid unlikely behaviour like getting stuck in singularity or exceeding joints limits.

For admissible positions the robot approaches workpiece and picks it,then it places it on a predifined position.For non admissible position the program gives us a warning about this position and prints the matrix on the output.

6 Conclusions And Future Works

The findings of this study are restricted To visual components kinematic solver for which its is recommended to connect the robot's controller to visual components for future works in order to produce more precise and better results.

Furthermore, the computer vision software could be trained in a better way to optimize its usage and to the possibility of its usage as a live recording which allows the adjustment of the program at the same time as the workpiece changes its position.

We demonstrated how simulation can be useful in solving robot's programming issues like pick and place specifically as they are very common among robot's tasks, first by identifying these issues and then solving them.

References

[1] Za’er S Abo-Hammour et al. “Continuous genetic algorithms for collision-free cartesian path planning of robot manipulators”. In: *International Journal of Advanced Robotic Systems* 8.6 (2011), p. 74.

[2] Amit Agrawal et al. “Vision-guided robot system for picking objects by casting shadows”. In: *The International Journal of Robotics Research* 29.2-3 (2010), pp. 155–173.

[3] Sergey Astanin et al. “Reflective workpiece detection and localization for flexible robotic cells”. In: *Robotics and Computer-Integrated Manufacturing* 44 (2017), pp. 190–198.

[4] Dana H Ballard and Christopher M Brown. “Computer vision, article, 4 pages prentice-hall”. In: *Englewood Cliffs, New Jersey, believed to be published more than one year prior to the filing date of the present application* (1982).

[5] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).

[6] Tadhg Brosnan and Da-Wen Sun. “Improving quality inspection of food products by computer vision—-a review”. In: *Journal of food engineering* 61.1 (2004), pp. 3–16.

[7] Tadhg Brosnan and Da-Wen Sun. “Inspection and grading of agricultural and food products by computer vision systems—a review”. In: *Computers and electronics in agriculture* 36.2-3 (2002), pp. 193–213.

[8] Peter Donelan. “Kinematic singularities of robot manipulators”. In: *Advances in Robot Manipulators*. InTech, 2010.

[9] Gábor Erdős et al. “Process planning and offline programming for robotic remote laser welding systems”. In: *International Journal of Computer Integrated Manufacturing* 29.12 (2016), pp. 1287–1306.

[10] Kevin R Fall and W Richard Stevens. *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley, 2011.

[11] Rafael C Gonzalez, Steven L Eddins, and Richard E Woods. *Digital Image Publishing Using MATLAB*. Prentice Hall, 2004.

[12] Hyejin Han and Jaeheung Park. "Robot control near singularity and joint limit using a continuous task transition algorithm". In: *International Journal of Advanced Robotic Systems* 10.10 (2013), p. 346.

[13] Kensuke Harada et al. "Validating an object placement planner for robotic pick-and-place tasks". In: *Robotics and Autonomous Systems* 62.10 (2014), pp. 1463–1477.

[14] Avinash Kumar. "Inverse Kinematics in a robotic arm and methods to avoid singularities". In: *Solid Mechanics & Design, IIT KanpurRoll N0. 10105017* (2011).

[15] Ji-Hyoung Lee, Chang-Sei Kim, and Keum-Shik Hong. "Off-Line Programming in the Shipbuilding Industry". In: *International Journal of Control, Automation, and Systems* 3.1 (2005), pp. 32–42.

[16] Yanjie Liu et al. "A Method of Energy-Optimal Trajectory Planning for Palletizing Robot". In: *Mathematical Problems in Engineering* 2017 (2017).

[17] Petar Marić. "Computer vision systems for the enhancement of industrial robots flexibility". In: *Facta universitatis-series: Automatic Control and Robotics* 10.1 (2011), pp. 1–18.

[18] Pedro Neto. "Off-line programming and simulation from CAD drawings: Robot-assisted sheet metal bending". In: *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*. IEEE. 2013, pp. 4235–4240.

[19] Pedro Neto, J Norberto Pires, and A Paulo Moreira. "CAD-based off-line robot programming". In: *Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on*. IEEE. 2010, pp. 516–521.

[20] Zengxi Pan et al. "Automated offline programming for robotic welding system with high degree of freedoms". In: *Advances in Computer, Communication, Control and Automation*. Springer, 2011, pp. 685–692.

[21] Zengxi Pan et al. “Recent progress on programming methods for industrial robots”. In: *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*. VDE. 2010, pp. 1–8.

[22] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[23] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[24] Mohan Manubhai Trivedi, Tarak Gandhi, and Joel McCall. “Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety”. In: *IEEE Transactions on Intelligent Transportation Systems* 8.1 (2007), pp. 108–120.

[25] Zhixuan Wei et al. “Manipulator motion planning using flexible obstacle avoidance based on model learning”. In: *International Journal of Advanced Robotic Systems* 14.3 (2017), p. 1729881417703930.