A. James Clark
School of Engineering

University of Maryland, College Park

Introduction to Robot Modelling

Size-based Object Marking and Segregation

Members: Prasheel Renkuntla

UID: 116925570

Contents

Acknowledgement

I would like to show my deepest gratitude to Prof. Chad Kessens, Introduction to Robot Modelling, University of Maryland, College Park, whose constant and apt guidance on the project helped me overcome the challenges faced. I would also like to thank all my friends, family, fellow students, and faculty who were involved directly or indirectly towards the completion of this project.

Abstract

Kuka LBR4 iiwa R820 is a 7R serial link manipulator. This robotic arm was designed in comparison to a human arm giving it exceptional capabilities over other 7DoF robot manipulators. This project discusses the use of Kuka R820 arm with an RG2 gripper to maneuver in its configuration space to pick an object, mark it based on the size and then segregate them accordingly.
The kinematics of the robot arm that include Forward and Inverse kinematics, along with the contact model of the gripper are discussed. Further, this project is simulated in CoppeliaSim and the model is evaluated with results from simulation and MATLAB.

List of Figures

List of Tables

1 Introduction

The Kuka LBR 4 iiwa R820 is a 7 degree of freedom (DoF) series link manipulator. LBR stands for "Leichtbauroboter" which in german means lightweight robot. The term "iiwa" means intelligent industrial work assistant, which suits the application of this project. All the joints in this arm are revolute joints giving it a freedom to rotate in a workspace of $\pm 170°$. With its design and high precision, it has a better performance over other 7 DoF serial manipulators.
The RG2 gripper is a flexible gripper model that has the ability to grasp an object and hold them for weights ranging between 2-5kgs. It is used to create an ideal two finger contact model which has two soft finger contacts that is used in simulation. A Robotiq 85 gripper can also be used to grasp the object though it will have a similar contact model.
For the project, we use the Denavit-Hartenberg[5] convention to derive the forward kinematics. Once, the end-effector position is known, we will find the inverse kinematic model for the robot which will be verified in simulation using CoppeliaSim and MATLAB. The Contact model will discuss the grasp analysis of the object between the gripper.

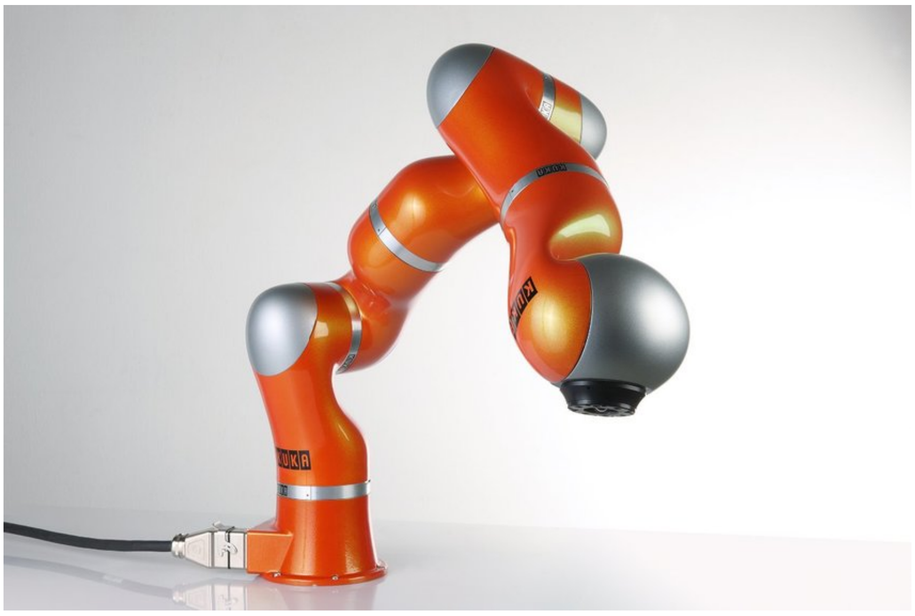
Figure 1: KUKA LBR 4 iiwa R820

image: https://www.coboticsworld.com/cobots-list

1.1 Motivation

Each product undergoes rigorous testing before it is launched into the market. For a food manufacturing plant, each stage from raw material to final product is tested carefully. In such a scenario, when an undesired incident occurs,i.e, a hazardous event that can affect

the personnel working and consequently stops the food production.
Such a problem can be solved by the use of a robot arm in a testing facility. The arm maneuvers through its workspace doing the processes that humans can do without being damaged and also, with an accuracy of more than 90%. Such a safe environment can help both the personnel and the company producing the product.
Hence, to visualise this scenario, a 7 DoF serial link manipulator is used to complete the tasks in a testing facility setting. The arm here will use a gripper to pick an object. Based on the object size, the arm will take it to the corresponding sprayer to get painted, and place it accordingly on a table. Using the mathematical model that we know, we will define the motion of the object to do such a process. With this defined model, we will simulate a robot arm with a gripper to validate the kinematic and contact model.

Figure 2: Autonomous Testing Facility

image : https://www.kuka.com

2 Robot Description

The project will use a 7R series link manipulator from the Kuka LBR4 iiwa series. The arm R820[2] has a capacity of holding about 7-14kgs payload. It was designed according to the human arm perspective. It is a high precision instrument that can accurately reach a position in its workspace. Kuka arms are even used in the medical industries for surgeries due to its precision. It has a low weight of about 23.9kgs for a 7kg payload and 29.9kgs for 14 kg payload. Also, this 1.3m long structure of the arm has an advantage that it has a footprint of only 136mm for a payload capacity of 7kgs. For complex assembly tasks, the repeatability is $\pm 1mm$. The safety functions of the LBR iiwa meet the requirements of performance Level d with structure category 3 (ISO 9283). Java technology is used to program the robot for sequence programming, thus giving it maximum modularity, openness and simplicity.

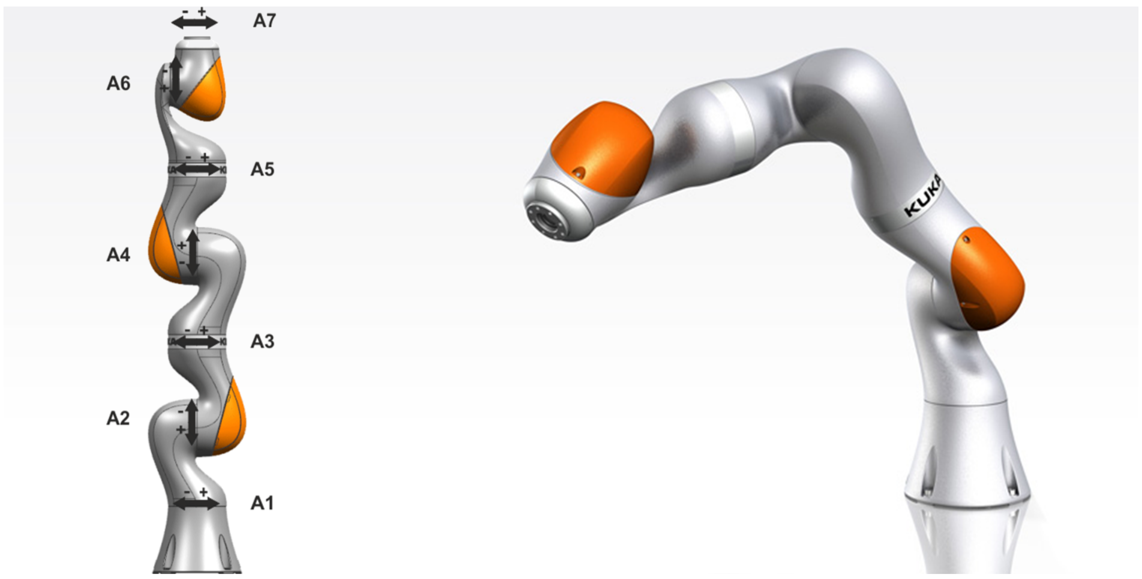

Figure 3: Manipulator axes

image : https://www.kuka.com

2.1 Workspace of the arm

The robot is 1.306 mm high and has 7 revolute joints thus having 7 axes. The axes 2 and 6 are revolve opposite to the axis 4. All these three axes have a rotational limit of about ±120°, while the rest of the joints except the 7th joint can achieve angles upto ±170°. The 7^th revolute joint just has an additional 5 degree advantage over ±170°. The robot workspace is thus given below.

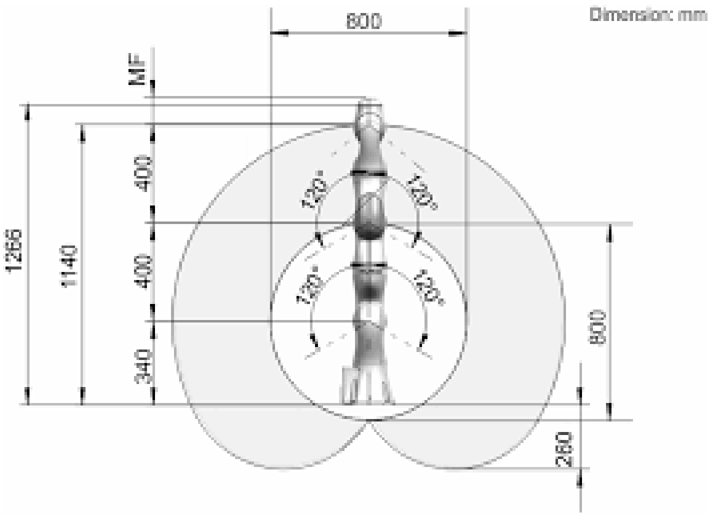

Figure 4: Robot Arm Workspace

2.2 Gripper Description

The RG2[1] is a two finger gripper that can be used for different applications such as assembly, material addition, material handling/packing, material removal, machine tending and quality, etc. In the physical world, it is compatible with only UR Robot arms but it is ideal to create a two finger contact model and to visualise it in simulation.

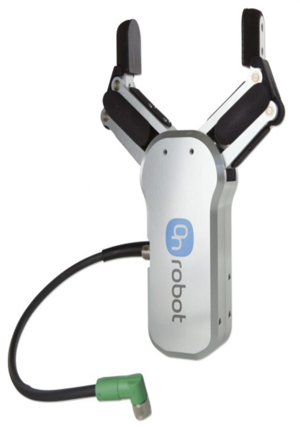

Figure 5: RG2 Gripper

image : https://www.universal-robots.com/plus/handling-grippers/rg2-gripper

3 Model Assumptions

In order to define the kinematic model of the robot manipulator, the following assumptions were made -

- Objects of interest are rigid with high friction coefficient.
  These objects (containers in a chemical lab) are rigid dynamic bodies. Friction coefficient is high as the gripper will be able to hold it steady.
- Objects are placed in Fixed location.
  The arm can position itself near these objects to pick and place at certain positions. Hence, no sensing modelling is defined.
- Painting modifier will run only when the object is in front of it.
- No path planning for the arm.
  Usually, for a real world scenario, there might be obstacles present in the workspace of the arm. When motion is allowed, the arm might collide with them. Since, the object locations are fixed, there is no path planning done on the arm to avoid these obstacles.
- Kinematics of the model are considered.
  The dynamics of the arm are neglected as the arm is allowed to move with a very low velocity and acceleration.

4 Kinematics

The kinematic model[6] studies the motion of a robot mechanism regardless of forces and torque that cause it. It allows to compute the position and orientation of robot manipulator's end-effector relative to the base of the manipulator, as a function of joint variables. It consists of a forward kinematic model and an inverse kinematic model. The forward kinematic model is used to compute the corresponding variables of the end-effector in a given reference frame (e.g. a Cartesian frame). It can be defined as a function f defined between the joint space R^n and the workspace R^m:

$$x = f(q); x \in R^m, q \in R^n \tag{1}$$

The inverse kinematics is used to compute the joint variables when the position of the end-effector is known from the workspace. It relates the variables from workspace to joint space. It is defined as a function $g = f^{-1} from R^m to R^n$:

$$q = g(x) = f^{-1}(x); q \in R^n, x \in R^m \tag{2}$$

Using the forward and inverse kinematic model, one can mathematically define the motion of the robot in a 3D space.

The contact model[4] is used to define the contact forces that will be applied by a gripper on an object. It is a fundamental requirement for analysis, design, plan and control of many robotic tasks. This project will use soft finger contact model with 2 contacts on the object surface.

4.1 Forward Kinematics

The forward kinematics are a transformation from the joint space q_i to the cartesian space $(x, y, z, \Phi, \Theta, \Psi)$. They are used to obtain the position and orientation of the end-effector of the manipulator when the joint angles using kinematic equations are given. Only the position (x, y, z) are obtained using the Denavit-Hartenberg convention. There are 4 parameters linked to each link i which are :

- θ_i : angle between x_{i-1} and x_i measured about z_{i-1}
- d_i : distance along z_{i-1} to the common normal
- a_i : distance along x_i to the common normal
- α_i : angle between x_{i-1} and x_i measured about z_{i-1}

The D-H parameters to compute the forward kinematics of the manipulator are shown in table below -

Once, the D-H parameters are found, the end effector position can be obtained by com-

Table 1: D-H Parameters for KUKA R820

T	θ	d	a	α
1	$\theta1^*$	$d1$	0	0
2	$\theta2^*$	$d2$	0	90
3	$\theta3^*$	0	0	-90
4	$\theta4^*$	$d3+d4$	0	-90
5	$\theta5^*$	0	0	90
6	$\theta6^*$	$d5+d6$	0	90
7	$\theta7^*$	$d7$	0	-90

puting the Homogeneous transformation matrix:

$$H = H_n^0 = \prod_{i=1}^{n} A_i^{i-1}(\theta_i) \tag{3}$$

where

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Here, the first three columns represent the orientation (rotation matrix) of the end-effector w.r.t the base frame and the last column represents the position of the end-effector.

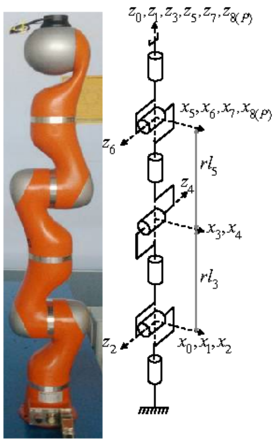
$z_0, z_1, z_3, z_5, z_7, z_{8(P)}$
$x_5, x_6, x_7, x_{8(P)}$
z_6
rl_5
z_4
x_3, x_4
rl_3
z_2
x_0, x_1, x_2

Figure 6: Schematic diagram with frame

image : https://www.semanticscholar.org/

For the Kuka R820, the final homogeneous matrix is given below :

$$H_0^7 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Here, $[p_x, p_y, p_z]^T$, is the end-effector position in the base frame. The computed transformation matrix has 7 joint angles and is calculated using MATLAB. The code is available in Appendix A-1.

4.2 Inverse kinematics

4.2.1 Singular Configurations

There are four kinematic singularities[3] in which motion is not possible in one Cartesian direction. The position of only 1 and 2 axes is important in each of these configurations whereas the other axes can be in any position.

- q_4 Singularity
  The Kinematic singularity is given when $q_4 = 0°$. The motion is blocked in the direction of the robot base or parallel to axis of joints q_3 and q_5.
- When $q_4 = 90°$ and $q_6 = 0°$
  Blocked motion in the joint axis for q_6 and q_2.
- When $q_2 = 0°$ and $q_3 = \pm 90°$
  Blocked motion in the robot axis or parallel to joint axis for q_2 or q_5
- When $q_5 = \pm 90°$ and $q_6 = 0°$
  Blocked motion in the axis parallel to joint axis of q_6

There are other system dependent singularities due to the structure of the robot. These can be avoided with a suitable elbow position.

- Wrist Angle Singularity
  When $q_6 = 0°$, the position of joint axis for q_5 and q_7 cannot be determined.
- q_1 Singularity
  Consider when q_1 is fixed, for any combination of q_2, q_3, q_4, the wrist can be in the same position, when the wrist axis q_5, q_6, q_7 are directly above the axis of q_1.
- q_2 Singularity
  From the above configuration, position of q_1, q_3 can no longer be resolved.
- q_2/q_4 Singlarity
  The position of axis q_1, q_7 can no longer be determined when q_1 and q_7 coincide.

4.2.2 Derviation of Joint Angles

The Kuka R820 has just one degree of redundancy. The last three joints conform a spherical wrist, therefore they are discarded to find the inverse kinematic[7] equations. Due to this, the fourth joint is a degenerated subchain which cannot be the redundant joint, and hence joints 1, 2, 3 can be. We chose the third joint, q_3 as the redundant joint and keep it fixed ($0°$) to analyse the equations for q_1, q_2, q_4. They are computed from the position of end-effector from eq. (5). And, the orientation of the end-effector given in the eq.(5) can be used to compute the joint angles q_5, q_6, q_7 that conform the spherical wrist. For determining the joint angle q_4, we follow the steps below-
First, from eq.(3), for n=7, we can re-write it of the form-

$$(A_1^0)^{-1} H_n^0 = A_2 \cdots A_7. \tag{6}$$

From the fourth column of each resultant matrix on either sides, we have :

$$\begin{aligned} p_x c_1 + p_y s_1 &= c_2(400 + 390c_4) + 390 s_2 c_3 s_4 \\ -p_x s_1 + p_y c_1 &= 390 s_3 s_4 \\ p_z - 310 &= -390 c_2 c_3 s_4 + s_2(390c_4 + 400) \end{aligned} \tag{7}$$

here, $c_i = \cos(q_i)$ and $s_i = \sin(q_i)$. Lengths 310, 390, 400 are the link lengths d2, d3+d4, d5+d6 in mm. From equations in (7) we get-

$$p_x^2 + p_y^2 + (p_z - 310)^2 = 400^2 + 390^2 + 2 * 400 * 390 * c_4$$
$$c_4 = a = \frac{p_x^2 + p_y^2 + (p_z - 310)^2 - 312100}{312000}$$

We will use $a\tan 2(y, x)$ because it covers all possible solutions in each of the four quadrants. Using, the trigonometric equations, we can derive the joint angle q_4 which is given by,

$$q_4 = a\tan 2(\pm\sqrt{1 - a^2}, a) \tag{8}$$

Now, we can compute q_1 from the equation:

$$-pxs_1 + p_y c_1 = 390 s_3 s_4$$
$$\implies q_1 = a\tan 2(390 s_3 s_4, \pm\sqrt{p_x^2 + p_y^2 - 390^2 s_3^2 s_4^2}) - a\tan 2(p_y, p_x) \tag{9}$$

Similarly, q_2 can be derived from the following:

$$p_z - 310 = -390 c_2 c_3 s_4 + s_2(390 c_4 + 400)$$
$$\implies q_2 = a\tan 2(p_z - 310, \pm\sqrt{(390c_4 + 400)^2 + (-390c_3 s_4)^2 - (p_z - 310)^2} - a\tan 2((390c_4 + 400), (-390 c_3 s_4)) \tag{10}$$

Now that we know joint angles q_1, q_2, q_4, we can determine the orientation of the end effector from the following equality that -

$$R_5 * R_6 * R_7 = R_4^{-1} * R_3^{-1} * R_2^{-1} * R_1^{-1} * R$$

Comparing the terms from the above resultant matrix on both sides-

$$\begin{bmatrix} c_5c_6c_7 - s_5s_7 & -c_7s_5 - c_5c_6c_7 & -c_5s_6 \\ c_7s_6 & -s_6s_7 & c_6 \\ -c_5s_7 - c_6c_7s_5 & c_6s_5s_7 - c_5c_7 & s_5s_6 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{11}$$

We compare the individual terms on both sides, i.e, $c_6 = a_{23}$ which from the trigonometric equations, we get,

$$q_6 = a\tan 2(\pm\sqrt{a_{13}^2 + a_{33}^2}, a_{23}) \tag{12}$$

From eq.(11), comparing and dividing a_{22}/a_{21}, we get,

$$q_7 = a\tan 2(\frac{-a_{22}}{\sin q_6}, \frac{a_{21}}{\sin q_6}) \tag{13}$$

From eq.(11), comparing and dividing a_{33}/a_{13}, we get,

$$q_5 = a\tan 2(\frac{a_{33}}{\sin q_6}, \frac{-a_{13}}{\sin q_6}) \tag{14}$$

4.3 Model Analysis

The forward kinematics was evaluated by first giving these set of angles $\underline{q}$ = [70, 90, 90, 0, 90, 0, 90]^T with $d1 = 0.1575m, d2 = 0.2025m, d3 + d4 = 0.390m, d5 + d6 = 0.400m, d7 = 0.0450m$. Then, using the D-H table and the H transform matrices, the final point was calculated. The arm was able to reach the first painting site which was at $(-0.0501, 0.6916, 1.0381)m$.
Then, the inverse kinematic model was evaluated by using the inverse kinematic equations. The coordinate supplied to the model was $(0.8452, 0.3754, 0.2748)m$, and the angles were found as, $q = [74.99, 95.00, 88.89, 75.00, 89.98, 0.00, 90.00]^T$ which was tested in simulation by giving these angles directly to the arm. The Joint position graph that depicts all the motion done by the arm is given below-

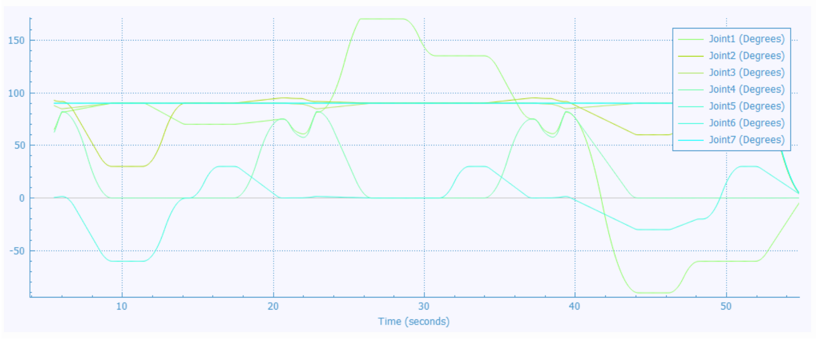

Figure 7: Joint Positions

For each interaction between the gripper and object, the inverse kinematics of the tip (end effector) and target(Object) group changes. The following graph shows the inverse kinematics over the complete time used by the arm to reach its initial pose-

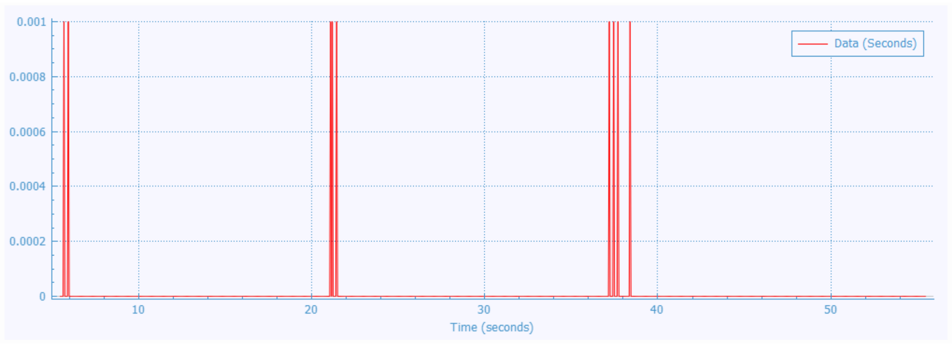

Figure 8: Tip Target IK Graph

On each grasp of the object, the gripper orients itself w.r.t the object. This can be seen in the graph below-

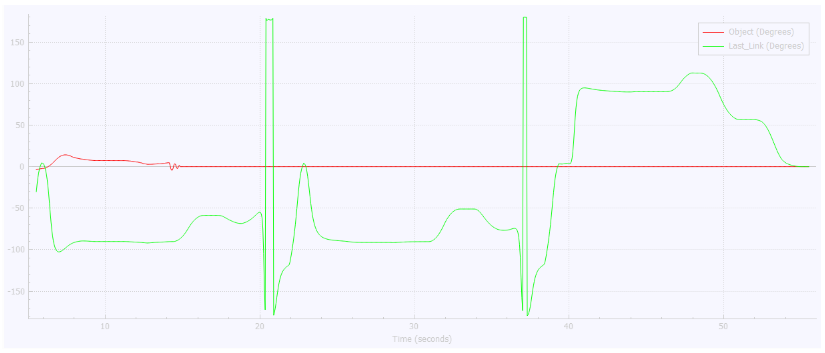

Figure 9: Target motion vs time

5 Contact Model for the gripper

5.1 Model Description

There are different contact models to grasp an object. They are Frictionless point contact model, Point contact model with friction and Soft-finger Contact model. This project includes the use of an RG2 gripper which has two fingers. These two contacts follow the soft-finger contact model to grasp an object.

From the textbook[5], the representation of the grasp consists of a matrix $G \in R^{pXm}$ and a set FC (Friction Cone) $\subset R^m$ which satisfies the following properties -
1. FC is a closed subset of R^m with non-empty interior.
2. $f_1, f_2 \in$ FC $\implies \alpha f_1 + \beta f_2 \in$ FC, for $\alpha, \beta > 0$
The set of wrenches that can be applied to the object by the two contacts is of the form

$$F_o = Gf_c = \begin{bmatrix} R_{c_i} & 0 \\ \hat{p_{c_i}} R_{c_i} & R_{c_i} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} f_{c_i} \tag{15}$$

for $f_c \in FC$ and $f_c \geq 0$ Here, $\hat{p_{c_i}}$ is a matrix that $\in$ SE(3) which is a skew symmetric matrix. Consider the figure below,

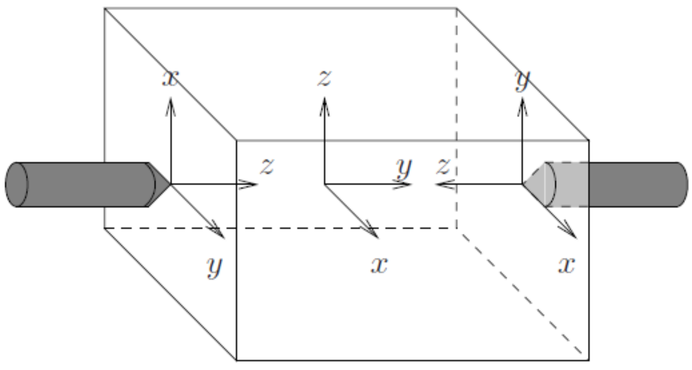

Figure 10: Object grasp with two soft finger contacts

image : https://www.cds.caltech.edu/ murray/books/MLS/pdf/mls94-complete.pdf

The position of the two contacts w.r.t object frame is given by -

$$p_{c_1} = \begin{bmatrix} 0 & 0 & -r \\ 0 & 0 & 0 \\ r & 0 & 0 \end{bmatrix}, p_{c_2} = \begin{bmatrix} 0 & 0 & r \\ 0 & 0 & 0 \\ -r & 0 & 0 \end{bmatrix} \tag{16}$$

The Rotation matrix for eq. (15) R_{c_i} for each of the contacts w.r.t object frame is given by -

$$R_{c_1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, R_{c_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{17}$$

Substituting eq.(16) and eq.(17) in eq.(15), and computing the product $\hat{p_{c_i}} R_{c_i}$ we get the following grasp map,

$$G \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -r & 0 & 0 & 0 & 0 & +r & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & +r & 0 & 0 & -r & 0 & 0 & 0 \end{bmatrix}, \tag{18}$$

Together with the contact forces, $f_c = (f^1_{c1} f^2_{c1} f^3_{c1} f^4_{c1} f^1_{c2} f^2_{c2} f^3_{c2} f^4_{c2}) \in R^8$ the friction cone thus can be defined as,

$$FC = FC_{c_1} x FC_{c_2} \tag{19}$$

$$FC_{c_1} = f_c : \sqrt{(f^1_{c1})^2 + (f^2_{c1})^2} \leq \mu f^3_{c1}, |f^4_{c1}| \leq \gamma f^3_{c1}, f^3_{c1} \geq 0$$

$$FC_{c_2} = f_c : \sqrt{(f^1_{c2})^2 + (f^2_{c2})^2} \leq \mu f^3_{c2}, |f^4_{c2}| \leq \gamma f^3_{c2}, f^3_{c2} \geq 0$$

where, μ is the friction coefficient and $\gamma > 0$ is the torsional friction coefficient.

5.2 Gripper Model Analysis

The contact model of the gripper was visualised using the RG2 gripper. The total wrench force that needs to be applied for the grasp is around 20N which is very high when compared to the mass of the object (0.3kg). This is mainly due to the high friction coefficient that was set in the tool which generally multiplies the forces that act on an object. The left and right finger forces converge for each grasp made. The graph shows the force applied to hold each object-

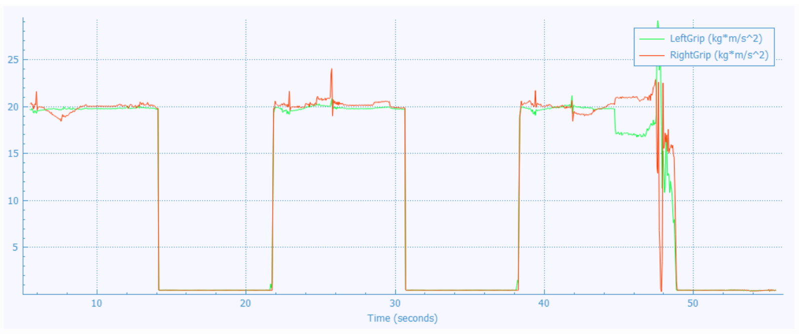

Figure 11: Wrench Force for Grasp

On each grasp, there is a hike in the force whereas there should be a low value for the left and right finger position w.r.t world frame. The following graph verifies the force graph (fig 16)-

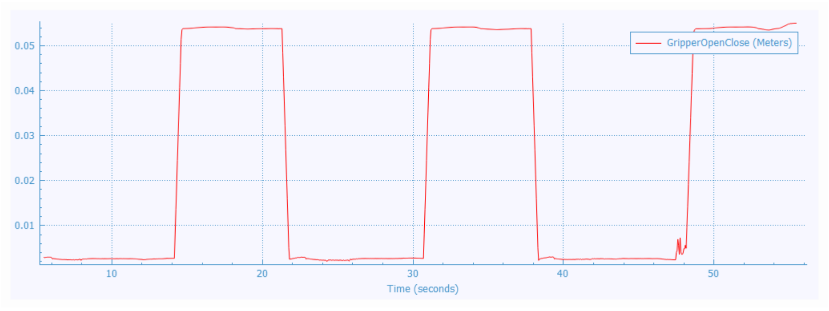

Figure 12: Gripper finger positions for Grasp

6 Simulation

This project was simulated in CoppeliaSim (formerly VRep) and Matlab, for both validating and visualising the kinematic model thus discussed in the report. The custom environment consists of the KUKA R820 arm firmly mounted on the ground. The arm moves in its configuration space entirely to move to three different locations, to demonstrate the efficient use of the arm. The figure below shows the setup -

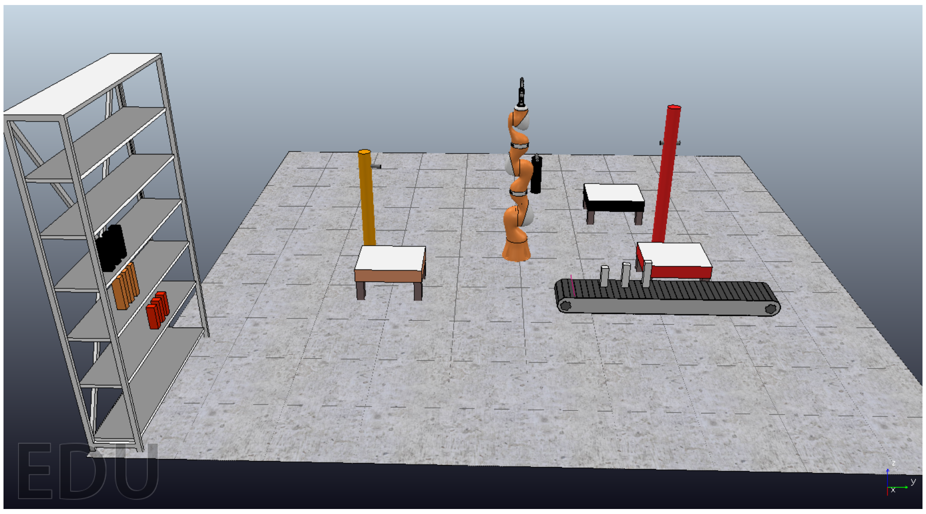

Figure 13: Simulation setup

The three colored cylindrical rods are visualised to show the different heights that the KUKA can achieve in its workspace. Each rod has a different colored sprayer that sprays the object based on the size. The conveyor belt has three boxes of different sizes which are to be grasped by the RG2 gripper connected to the arm. Figure below shows the objects-

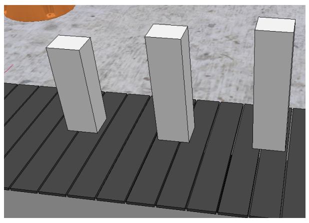

Figure 14: Objects on Conveyor belt

The gripper RG2 is assembled to the connecting point on KUKA R820. The following figure shows the connection-

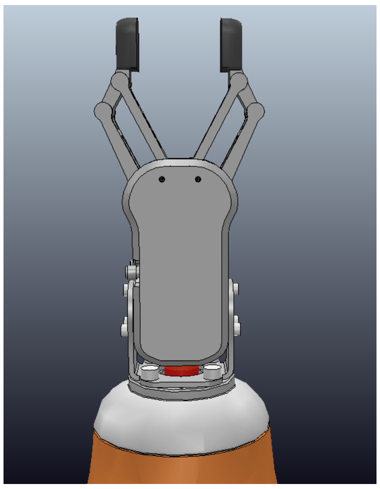

Figure 15: Gripper Connection

Figure below shows the black sprayed object carried by the arm to place it on the table nearby.

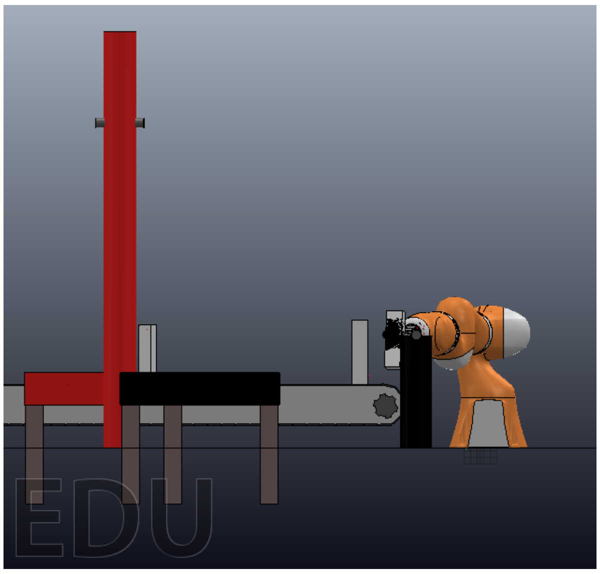

Figure 16: Object Sprayed

The arm is programmed in such a way that given the position of the object, the angles computed from the inverse kinematic equations, make the arm reach the object and complete the task of picking the object, get it sprayed and place it on the table. After successful completion of the task, the arm is programmed to go back to the rest position. The figure below shows the completion of the task.

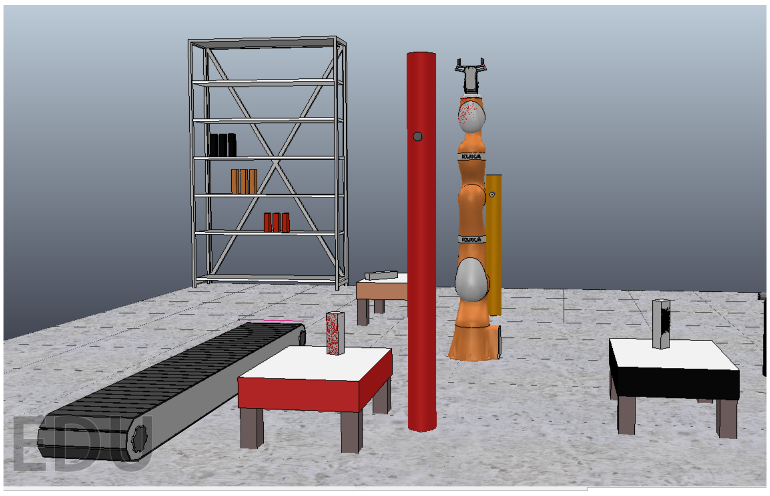

Figure 17: Task Completion

7 Conclusion

The kinematic model of the arm was computed. The Forward kinematic model gave the location of the end effector in the world frame and the inverse kinematic solutions gave the joint angles to reach that particular point. The Soft finger contact model with two contacts was calculated which helped to conceptualise the gripper action. Once, the arm reached the destination point, the gripper model grasps the object with a force that is able to hold the object. With simulation in CoppeliaSim and MATLAB, this model was analysed.

7.1 Applications

This model shows that the arm can pick, color and place an object efficiently in its workspace. Following are some applications where this arm can be used-

- Chemical testing
- Product Packaging
- Food manufacturing plant

General applications include -

- Collaborative Robots
- Machine Trending
- Material Handling

7.2 Drawbacks and Challenges

The object size should be in a certain limit for the gripper to hold in its grasp. The model described here only discusses about the arm being firmly mounted on the ground. A high friction coefficient ensured that the gripper was able to hold the object in place though the object had a mass of only 0.3kg. Also, the robot arm cannot be operated in a crowded workspace.

7.3 Future Work

The kinematic model of the arm and the contact model for the gripper were discussed in the project. Further, path planning for the arm to reach a point in its workspace can be implemented to avoid collisions with the objects. Once, it reaches the end point, sensing of the object can be used to determine the size and shape of the object. In the future, a fully autonomous testing facility can be developed that would not harm any personnel working in the test site.

References

[1] O. R. ApS. Rg2 gripper. `https://www.universal-robots.com/media/1226143/rg2-datasheet-v14.pdf` Accessed: 2019-12-05.

[2] K. GmbH. Product catalog. `https://www.kuka.com/-/media/kuka-downloads/imported/9cb8e311bfd744b4b0eab25ca883f6d3/kuka_lbr_iiwa_brochure_en.pdf?rev=5a25f7eac825492e92af6343dbf5bc6b` Accessed: 2019-12-05.

[3] K. R. GmbH. Kuka sunrise os 1.11. `http://www.oir.caltech.edu/twiki_oir/pub/Palomar/ZTF/KUKARoboticArmMaterial/KUKA_SunriseOS_111_SI_en.pdf` Accessed: 2019-12-06.

[4] J. W. B. Imin Kao, Kevin Lynch. Contact modelling and manipulation. `https://link.springer.com/referenceworkentry/10.1007%2F978-3-540-30301-5_28` Accessed: 2019-12-06.

[5] S. S. S. Richard M. Murray, Zexiang Li. A mathematical introduction to robotic manipulation. `https://www.cds.caltech.edu/~murray/books/MLS/pdf/mls94-complete.pdf` Accessed: 2019-12-05.

[6] E. D. W Khalil. Kinematic model. `https://www.sciencedirect.com/topics/engineering/kinematic-model` Accessed: 2019-12-05.

[7] I. Zaplana and L. Basanez. A novel closed-form solution for the inverse kinematics of redundant manipulators through workspace analysis. *Mechanism and Machine Theory*, 121:829–843, 03 2018.

A Appendix

A.1 Matlab Code to calculate H-Matrix

```
syms d1;
syms d2;
syms d3;
syms d4;
syms d5;
syms d6;
syms d7;
syms dee;
syms theta1;
syms theta2;
syms theta3;
syms theta4;
syms theta5;
syms theta6;
syms theta7;
syms alpha1;
syms alpha2;
syms alpha3;
syms alpha4;
syms alpha5;
syms alpha6;
syms alpha7;

A1 = [cosd(theta1) -sind(theta1)*cosd(alpha1) sind(theta1)*sind(
    alpha1) 0;
      sind(theta1) cosd(theta1)*cosd(alpha1) -cosd(theta1)*sind(
          alpha1) 0;
      0 sind(alpha1) cosd(alpha1) d1;
      0 0 0 1];
A2 = [cosd(theta2) -sind(theta2)*cosd(alpha2) sind(theta2)*sind(
    alpha2) 0;
      sind(theta2) cosd(theta2)*cosd(alpha2) -cosd(theta2)*sind(
          alpha2) 0;
      0 sind(alpha2) cosd(alpha2) d2;
      0 0 0 1];
A3 = [cosd(theta3) -sind(theta3)*cosd(alpha3) sind(theta3)*sind(
    alpha3) 0;
      sind(theta3) cosd(theta3)*cosd(alpha3) -cosd(theta3)*sind(
          alpha3) 0;
      0 sind(alpha3) cosd(alpha3) 0;
    0 0 0 1];
A4 = [cosd(theta4) -sind(theta4)*cosd(alpha4) sind(theta4)*sind(
   alpha4) 0;
    sind(theta4) cosd(theta4)*cosd(alpha4) -cosd(theta4)*sind(
       alpha4) 0;
    0 sind(alpha4) cosd(alpha4) d3+d4;
    0 0 0 1];
A5 = [cosd(theta5) -sind(theta5)*cosd(alpha5) sind(theta5)*sind(
   alpha5) 0;
    sind(theta5) cosd(theta5)*cosd(alpha5) -cosd(theta5)*sind(
       alpha5) 0;
    0 sind(alpha5) cosd(alpha5) 0;
    0 0 0 1];
A6 = [cosd(theta6) -sind(theta6)*cosd(alpha6) sind(theta6)*sind
   (90) 0;
    sind(theta6) cosd(theta6)*cosd(alpha6) -cosd(theta6)*sind(
       alpha6) 0;
    0 sind(alpha6) cosd(alpha6) d5+d6;
    0 0 0 1];
A7 = [cosd(theta7) -sind(theta7)*cosd(alpha7) sind(theta7)*sind(
   alpha7) 0;
    sind(theta7) cosd(theta7)*cosd(alpha7) -cosd(theta7)*sind(
       alpha7) 0;
    0 sind(alpha7) cosd(alpha7) d7;
    0 0 0 1];

H1 = A1;
H2 = mtimes(A1,A2);
H3 = mtimes(A1,mtimes(A2,A3));
H4 = mtimes(A1,mtimes(A2,mtimes(A3,A4)));
H5 = mtimes(A1,mtimes(A2,mtimes(A3,mtimes(A4,A5))));
H6 = mtimes(A1,mtimes(A2,mtimes(A3,mtimes(A4,mtimes(A5,A6)))));
H7 = mtimes(A1,mtimes(A2,mtimes(A3,mtimes(A4,mtimes(A5,mtimes(A6,
   A7))))));
disp(H7);
```

A.2 Lua Script

```
enableIk=function(enable)
    if enable then
        sim.setObjectMatrix(ikTarget,-1,sim.getObjectMatrix(ikTip
           ,-1))
        for i=1,#jointHandles,1 do
            sim.setJointMode(jointHandles[i],sim.jointmode_ik,1)
        end
        sim.setExplicitHandling(ikGroupHandle,0)
    else
        sim.setExplicitHandling(ikGroupHandle,1)
        for i=1,#jointHandles,1 do
            sim.setJointMode(jointHandles[i],sim.jointmode_force,0)
        end
    end
end

setGripperData=function(open,velocity,force)
    if not velocity then
        velocity=0.11
    end
    if not force then
        force=20
    end
    if not open then
        velocity=-velocity
    end
    local data=sim.packFloatTable({velocity,force})
    sim.setStringSignal(modelName..'_rg2GripperData',data)
end


function sysCall_threadmain()
    -- Initialize some values:
    jointHandles={-1,-1,-1,-1,-1,-1,-1}
    for i=1,7,1 do
        jointHandles[i]=sim.getObjectHandle('LBR_iiwa_14_R820_joint'..i)
    end
    ikGroupHandle=sim.getIkGroupHandle('IK_Group')
    ikTip=sim.getObjectHandle('tip')
    ikTarget=sim.getObjectHandle('target')
    modelBase=sim.getObjectAssociatedWithScript(sim.handle_self)
    modelName=sim.getObjectName(modelBase)

    -- Set-up some of the RML vectors:
    vel=150
    accel=30
    jerk=40

    currentVel={0,0,0,0,0,0,0}
    currentAccel={0,0,0,0,0,0,0}
    maxVel={vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180}
    maxAccel={accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180}
    maxJerk={jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180}
    targetVel={0,0,0,0,0,0,0}

    ikMaxVel={0.4,0.4,0.4,1.8}
    ikMaxAccel={0.8,0.8,0.8,0.9}
    ikMaxJerk={0.6,0.6,0.6,0.8}

    initialConfig={0,0,0,0,0,0,0}
    pickConfig={75*math.pi/180,95*math.pi/180,90*math.pi/180,75*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180}
    --infront of sprayer
    dropConfig1={90*math.pi/180,30*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,-60*math.pi/180,90*math.pi/180}
    --drop site #2
    dropConfig2={180*math.pi/180,90*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180}
    --no effect
    dropConfig3={-90*math.pi/180,60*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,-30*math.pi/180,90*math.pi/180}
    --no effect
    --dropConfig4={-189.38*math.pi/180,24.94*math.pi/180,64.36*math.pi/180,0.75*math.pi/180,-90.02*math.pi/180,-9.41*math.pi/180,0*math.pi/180}
    actualDropConfig1={70*math.pi/180,90*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180}
    fallBackConfig1={70*math.pi/180,90*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,30*math.pi/180,90*math.pi/180}
    actualDropConfig2={135*math.pi/180,90*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180}
```

```lua
    fallBackConfig2={135*math.pi/180,90*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,30*math.pi/180,90*math.pi/180}
    actualDropConfig3={-60*math.pi/180,90*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,-20*math.pi/180,90*math.pi/180}
    fallBackConfig3={-60*math.pi/180,90*math.pi/180,90*math.pi/180,0*math.pi/180,90*math.pi/180,30*math.pi/180,90*math.pi/180}

    dropConfigs={dropConfig1,dropConfig2,dropConfig3}
    actualConfigs={actualDropConfig1,actualDropConfig2,actualDropConfig3}
    fallBackConfigs={fallBackConfig1,fallBackConfig2,fallBackConfig3}

    dropConfigIndex=1
    droppedPartsCnt=0

    enableIk(false)
    setGripperData(true)
    sim.setInt32Parameter(sim.intparam_current_page,0)
    --sim.wait(10)
    while droppedPartsCnt<3 do

        --reach near conveyor belt
        if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
            sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,pickConfig,targetVel)
        end

        --move forward a bit to grab the container
        if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
            enableIk(true)
            sim.setInt32Parameter(sim.intparam_current_page,1)

            pos=sim.getObjectPosition(ikTip,-1)
            quat=sim.getObjectQuaternion(ikTip,-1)
            sim.rmlMoveToPosition(ikTarget,-1,-1,nil,nil,ikMaxVel,ikMaxAccel,ikMaxJerk,{pos[1]+0.105,pos[2],pos
            [3]},quat,nil)
        end
        --grasp the container
        if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
            setGripperData(false)
            sim.wait(0.5)
        end
        --move a bit backwards for next location
        if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
            sim.rmlMoveToPosition(ikTarget,-1,-1,nil,nil,ikMaxVel,ikMaxAccel,ikMaxJerk,{pos[1]-0.1,pos[2],pos[3]+0.1},quat,nil)
        end
        --Get in front of the sprayer.
        if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
            enableIk(false)
            if(dropConfigIndex == 1) then
            sim.setInt32Parameter(sim.intparam_current_page,2)
            elseif (dropConfigIndex == 2) then
            sim.setInt32Parameter(sim.intparam_current_page,3)
            else
            sim.setInt32Parameter(sim.intparam_current_page,4)
            end
            sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,dropConfigs[dropConfigIndex],targetVel)
        end
        sim.wait(2)

        if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
            sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,actualConfigs[dropConfigIndex],targetVel)
        end

        if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
            setGripperData(true)
            sim.wait(0.2)
            end

            if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
                    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,fallBackConfigs[dropConfigIndex],targetVel)
            end
            sim.wait(0.2)

            --get up to go normal position
            if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
                sim.rmlMoveToPosition(ikTarget,-1,-1,nil,nil,ikMaxVel,ikMaxAccel,ikMaxJerk,pos,quat,nil)
            end

            --go back to the conveyor
            if sim.getSimulationState()~=sim.simulation_advancing_abouttostop then
                enableIk(false)

                sim.setInt32Parameter(sim.intparam_current_page,0)
                dropConfigIndex=dropConfigIndex+1
                if dropConfigIndex>3 then
                    dropConfigIndex=1
                end

                droppedPartsCnt=droppedPartsCnt+1
            end

        end

    -- go to start once done.
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,initialConfig,targetVel)
    sim.stopSimulation()
end
```