Flipkart GRID 2.0

Intelligent Picking - Round 3

Team Name : TEAM SHIFT

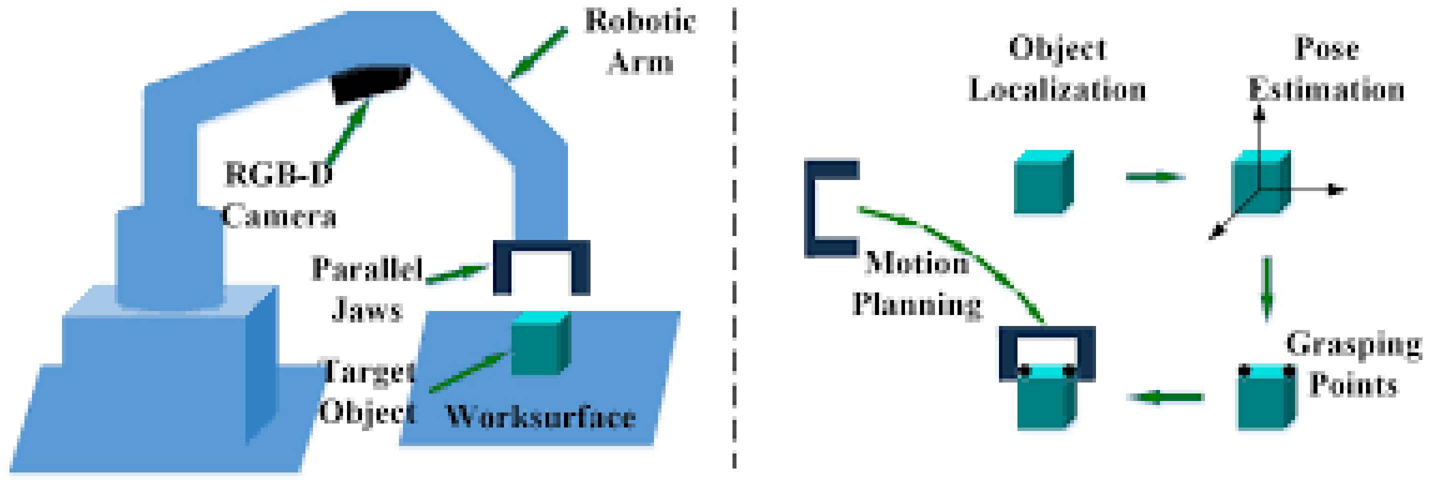
Institute Name : INDIAN INSTITUTE OF INFORMATION TECHNOLOGY RANCHI

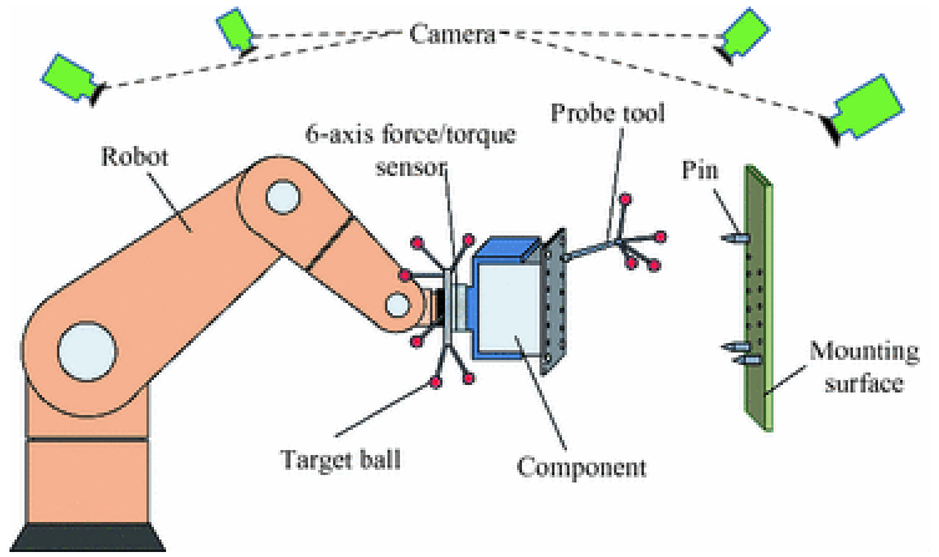
Team Details :-

S.No.	NAME	ACHIEVEMENT
1	Boidi Aniruddh (TEAM LEADER)	• Represented IIIT Ranchi in Technoxian (Quadcopter) , World Robotics Championship at Delhi. • Competed in Robo Soccer Robotic Competition in Technical Fest, Ojass at NIT Jamshedpur.
2	Kavita Yadav	• Represented IIIT Ranchi in Technoxian (Quadcopter) , World Robotics Championship at Delhi. • Competed in Line Follower Robotic Competition in Technical Fest, Ojass at NIT Jamshedpur.
3	Abhishek Kumar	• Summer Internship & Training in Embedded System and Robotics with IOT in association with Ei-Systems in Technex, IIT (BHU) Varanasi. • Competed in Line Follower Robotic Competition in Technical Fest, Ojass at NIT Jamshedpur.
4	Rupal Gupta	• Represented IIIT Ranchi in Technoxian (Quadcopter) , World Robotics Championship at Delhi. • Competed in Line Follower Robotic Competition in Technical Fest, Ojass at NIT Jamshedpur.
5	Vishal Raj	• Optical Society of America (OSA) Member and have organised Student Chapter in our campus at IIIT Ranchi. • Competed in Obstacle Avoider Robotic Competition in Technical Fest, Ojass at NIT Jamshedpur.

<u>Technical Specifications</u> :-

➢ **<u>Detailed structure</u>** :-

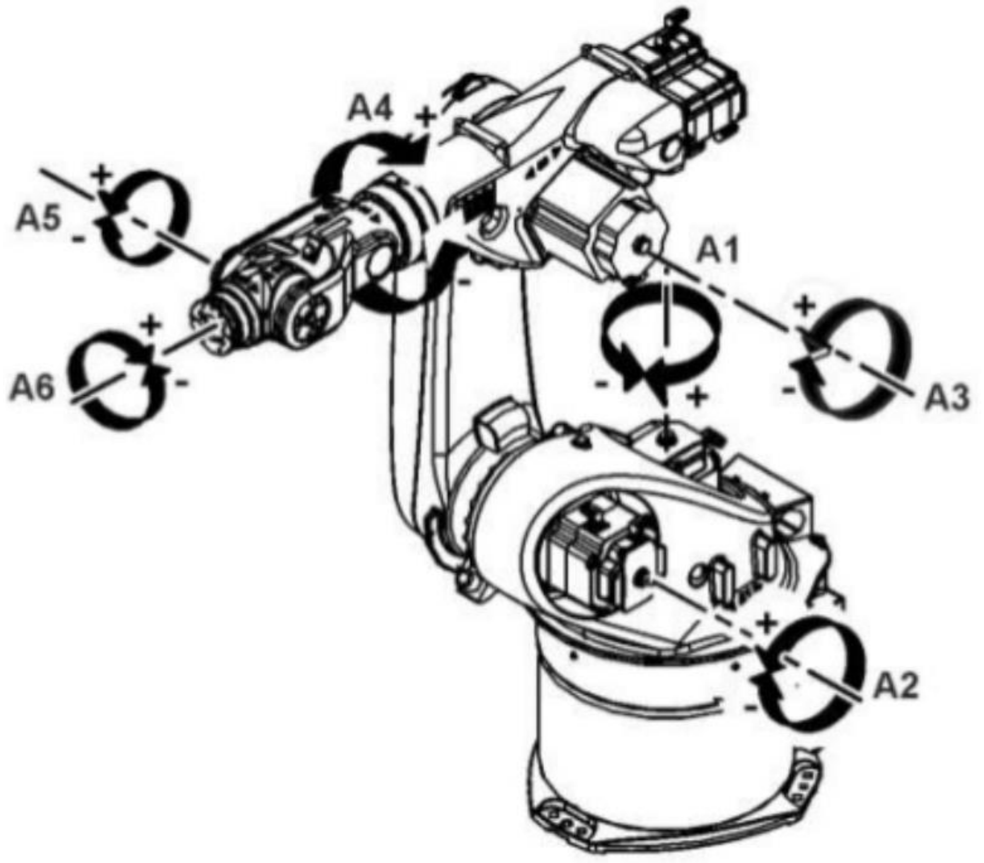




<u>Figure</u>

Technical Specifications (Architecture) :-

- Where A1, A2, A3, A4, A5 and A6 denotes 6 Degree of Freedom.



OBJECT PICKING AUTONOMOUS ROBOT

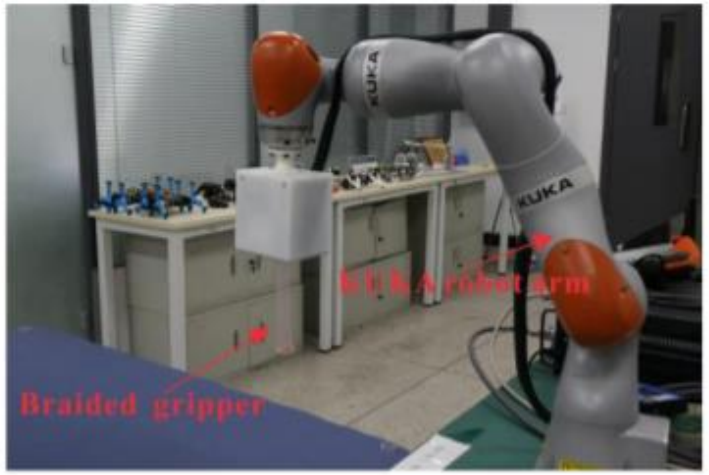
Gripping mechanism :-

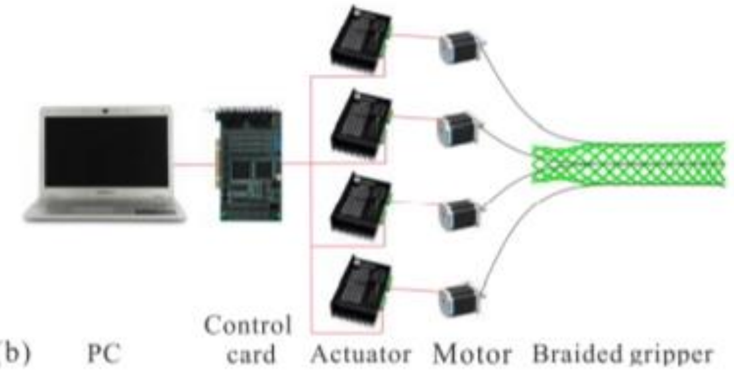
- Robotic grippers, which act as the end effector and contact the objects directly, play a crucial role in the performance of the robots. In this paper, we design and analyze a new robotic gripper based on the braided tube.

- Apart from deployability , a self-forcing mechanism, i.e., the holding force increases withload /object weight , facilitates the braided tube as a robotic gripper to grasp objects with different shapes, weights and rigidities . First, taking a cylindrical object as an example, the self-forcing mechanism is theoretically analyzed, and explicit formulas are derived to estimate the holding force.

- Second, experimental and numerical analyses are also conducted for a more detailed understanding of the mechanism. The results show that a holding force increment by 120% is achieved due to self-forcing, and the effects of design parameters on the holding force are obtained.

- Finally, a braided gripper is fabricated and operated on a our robot arm, which successfully grasps a family of objects with varying shapes, weights and rigidities. To summarize, the new device shows great potentials for a wide range of engineering applications where properties of the objects are varied and unpredictable.

(a)


(b)


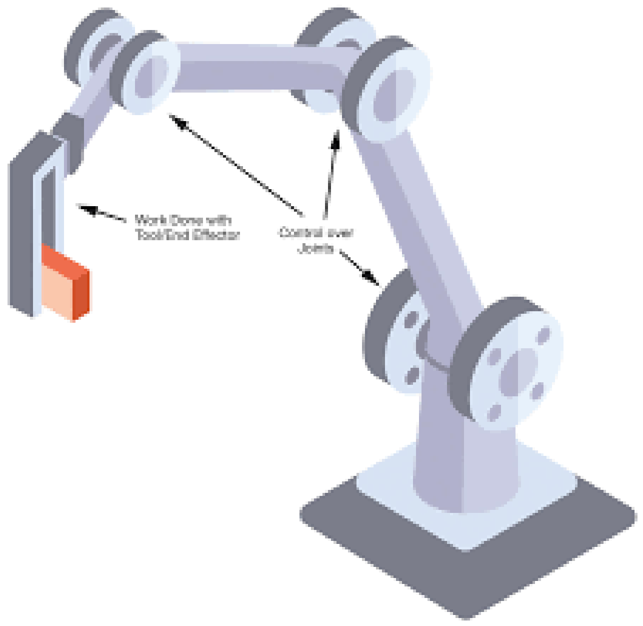
Work Envelope and Payload calculation :-

1. **Reach of the robot across the pickup/drop area**:- Our autonomous

   object picking arm robot will have a maximum height of 3100 mm.

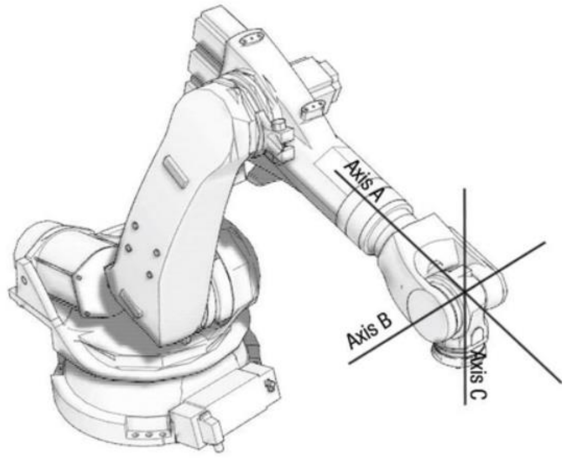
2. **Degrees of Freedom** :- 6 axis of rotation

3. **Payload capacity and calculations** :- 2.2 Kg



- The maximum allowed payload of the robot arm depends on the center of gravity offset. The center of gravity offset is defined as the distance between the center of the tool output flange and the center of gravity.

4. **Free body diagram** :-



ROBOT SPECIFICATION

5. **Torque & force calculations** :-

$T = F*L$ Nm

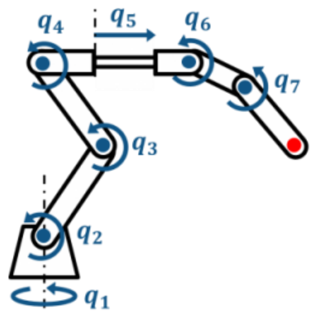
Where , F = Force acting on the motor
L = Length of the shaft

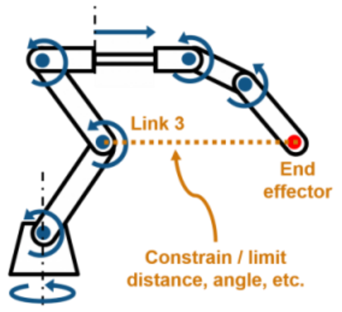
Force, F is given by,

$F = m*g$ N

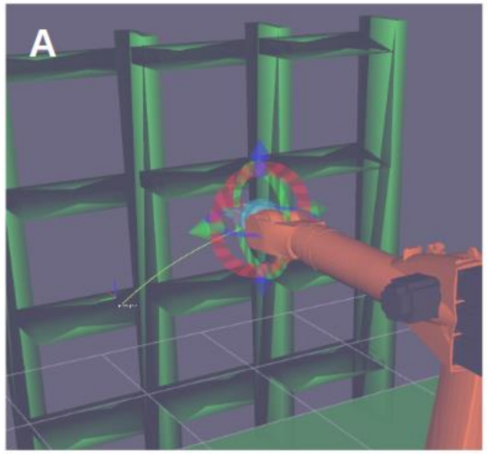
Where , m = mass to be lifted by the motor
g = acceleration due to gravity = 9.8 m/s^2

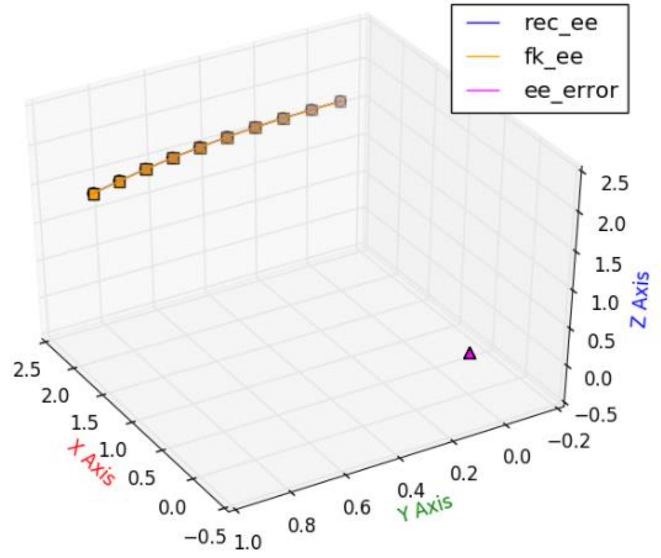
SPECIFICATION	VALUES
Axes / Degree of freedom	6
Payload	2.2 kg
Height Reach	3100 mm
Robot Mass	5 kg
Structure	Articulated
Mounting	Floor

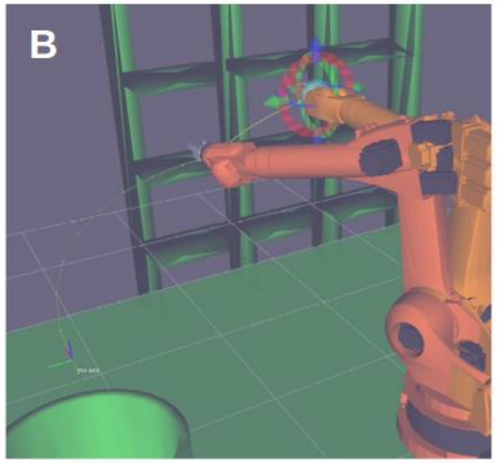


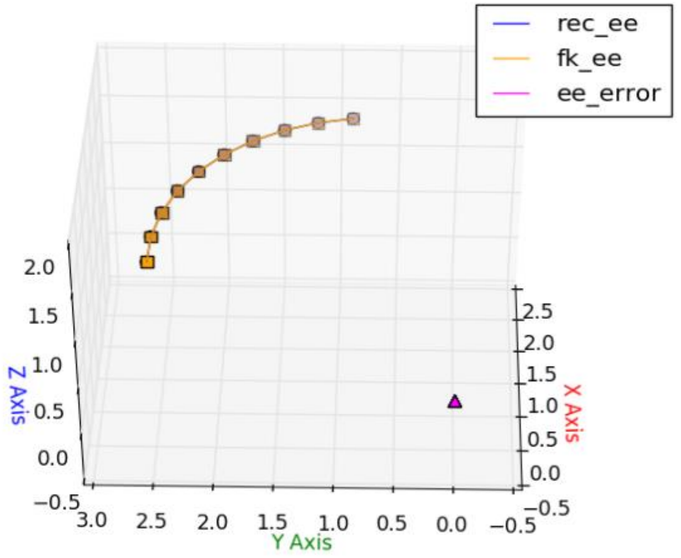


6. **Trajectory computation of moving components** :-

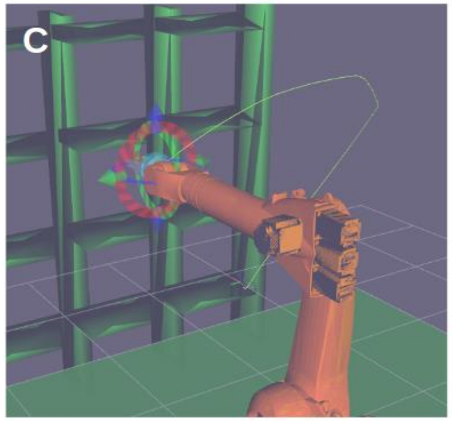


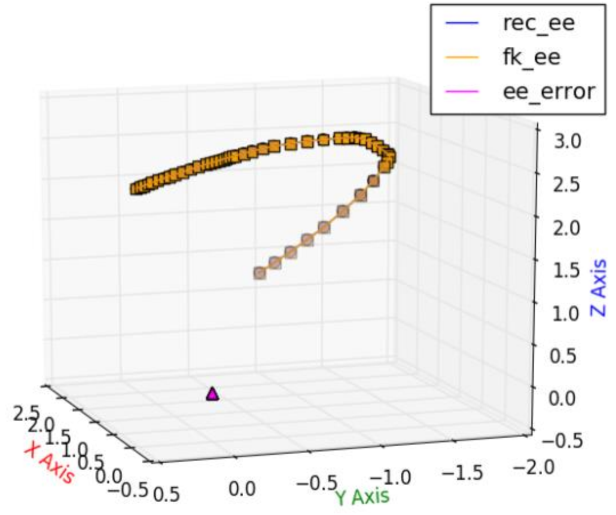


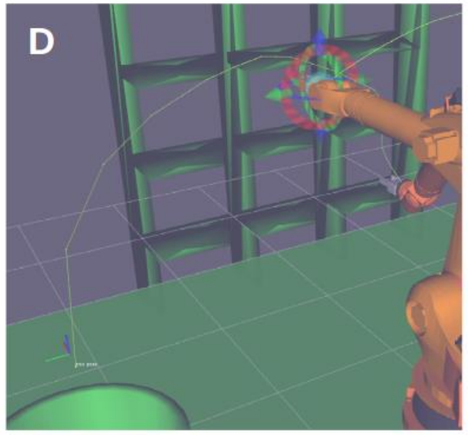


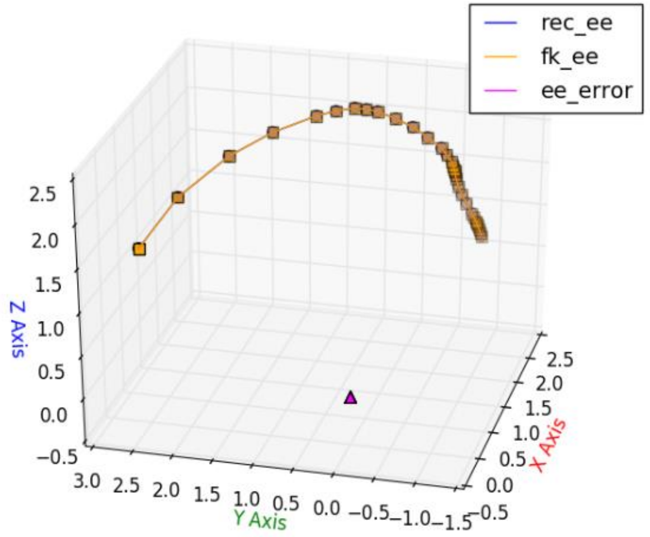


Trajectory computation of moving components (cont.) :-

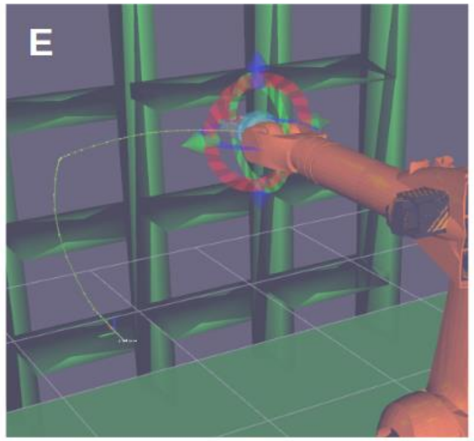


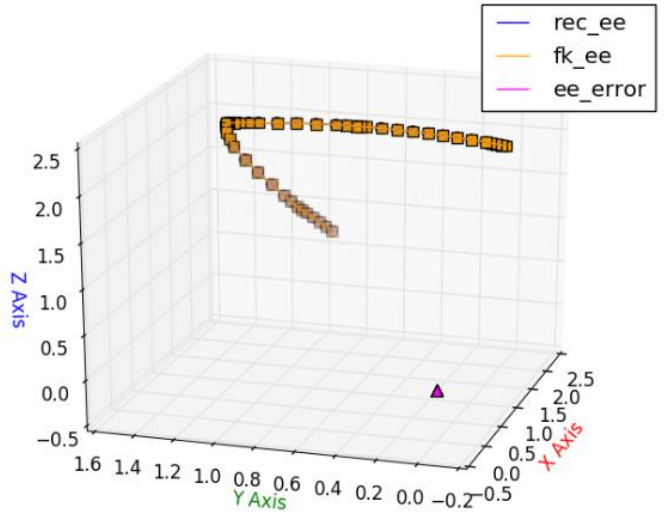


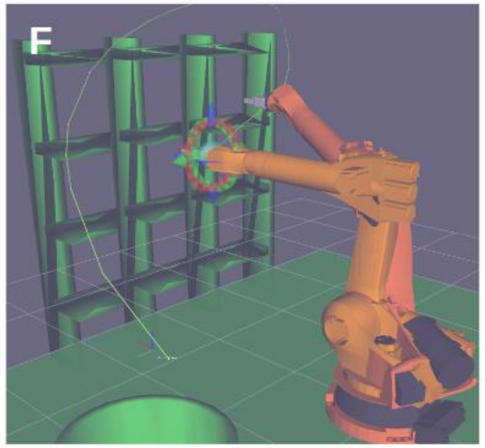


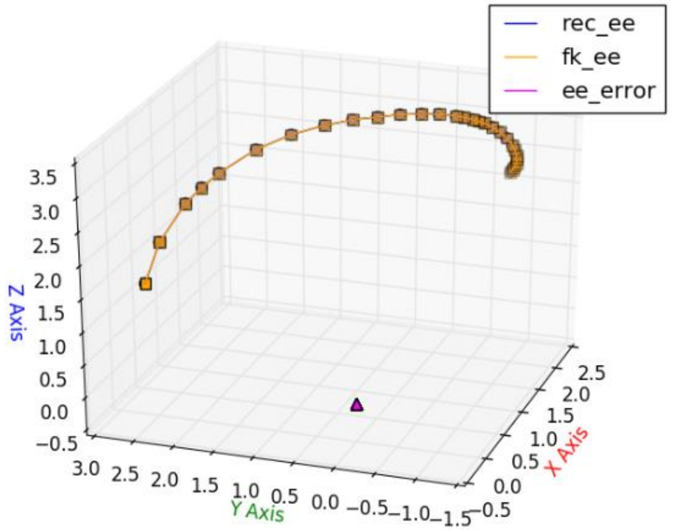


Trajectory computation of moving components (cont.) :-



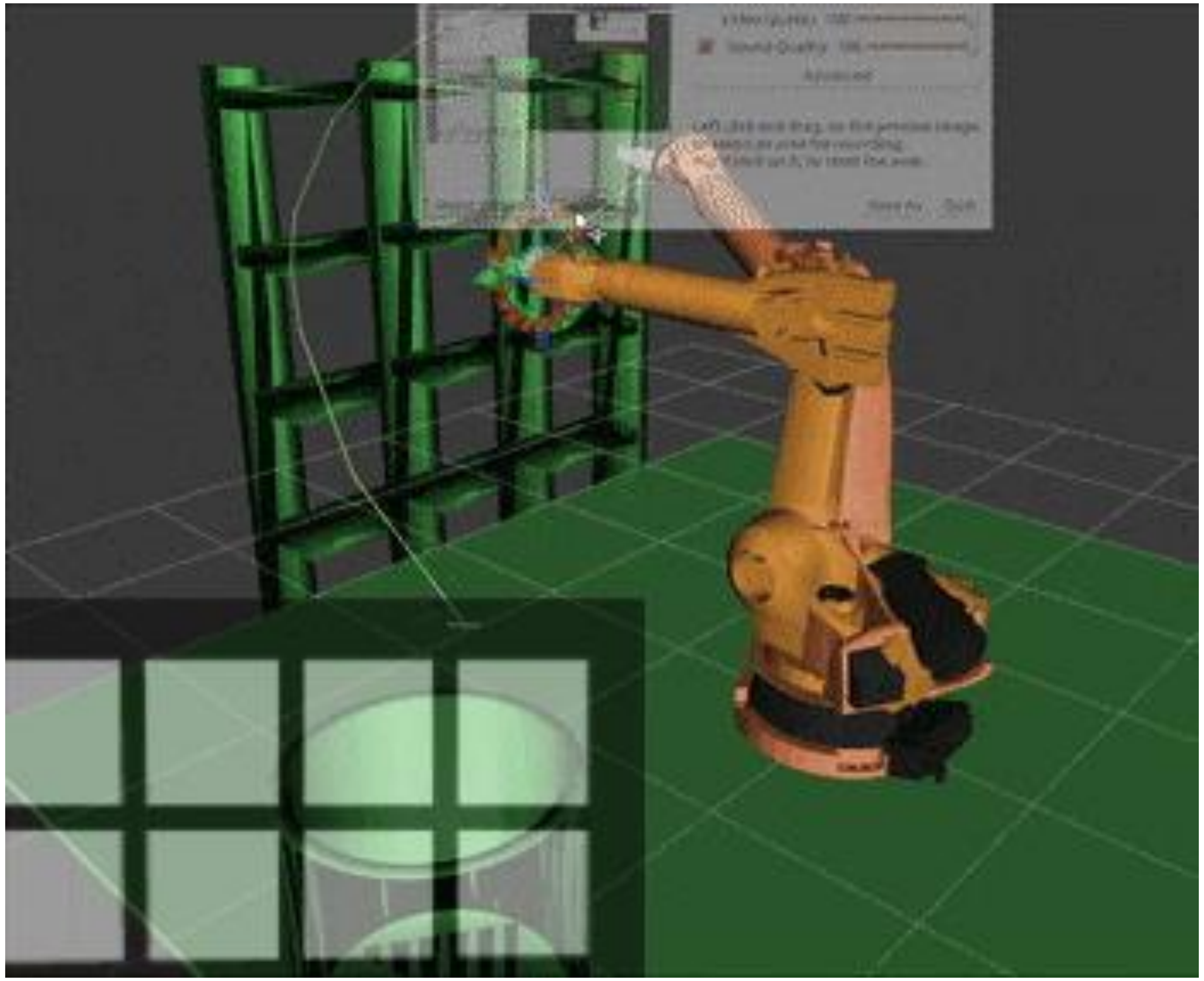






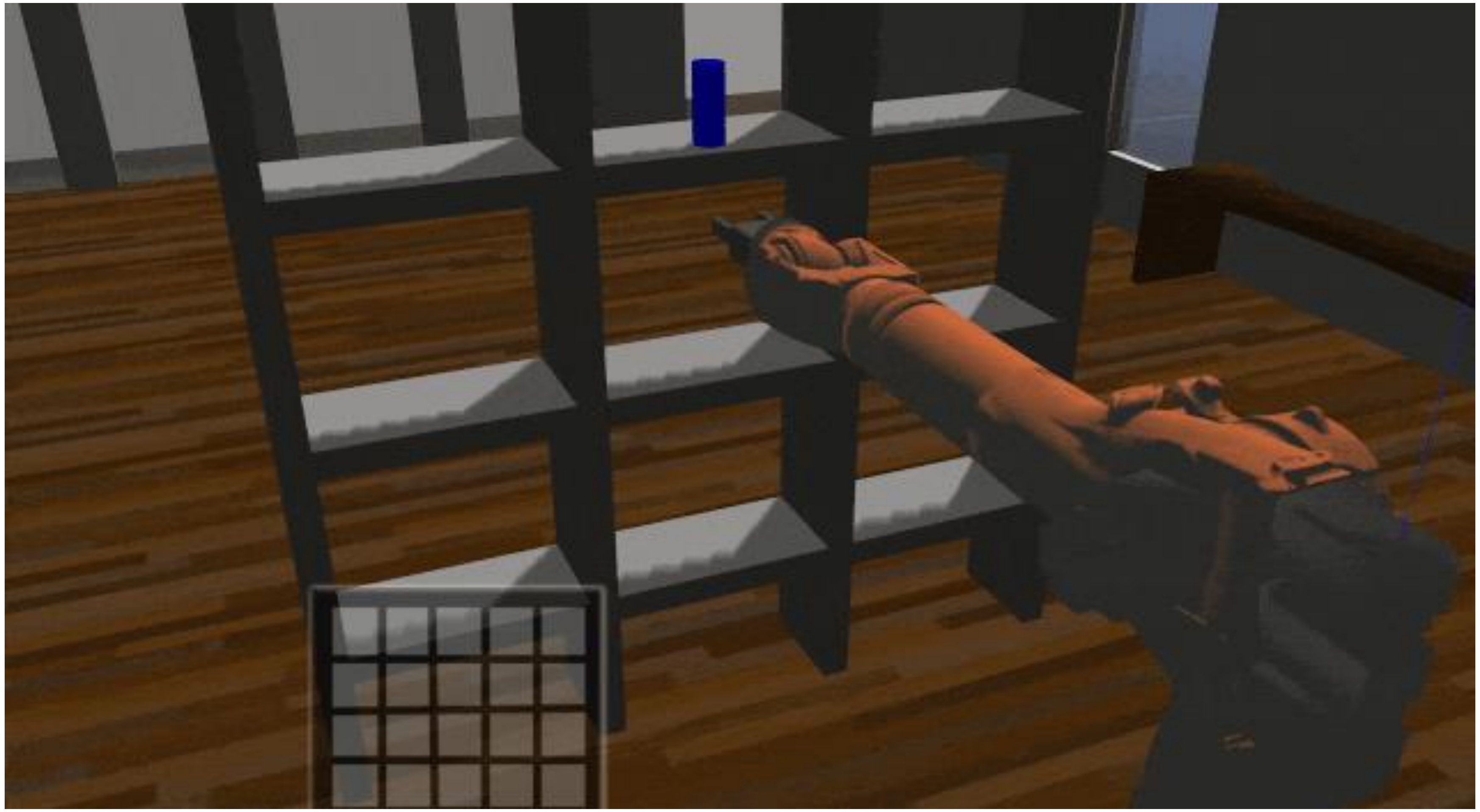
Trajectory motion path Working Models :-

- Video link: (https://github.com/Vrajsingh/flipkart_grid_robotics_arm_2.0-trajectory)



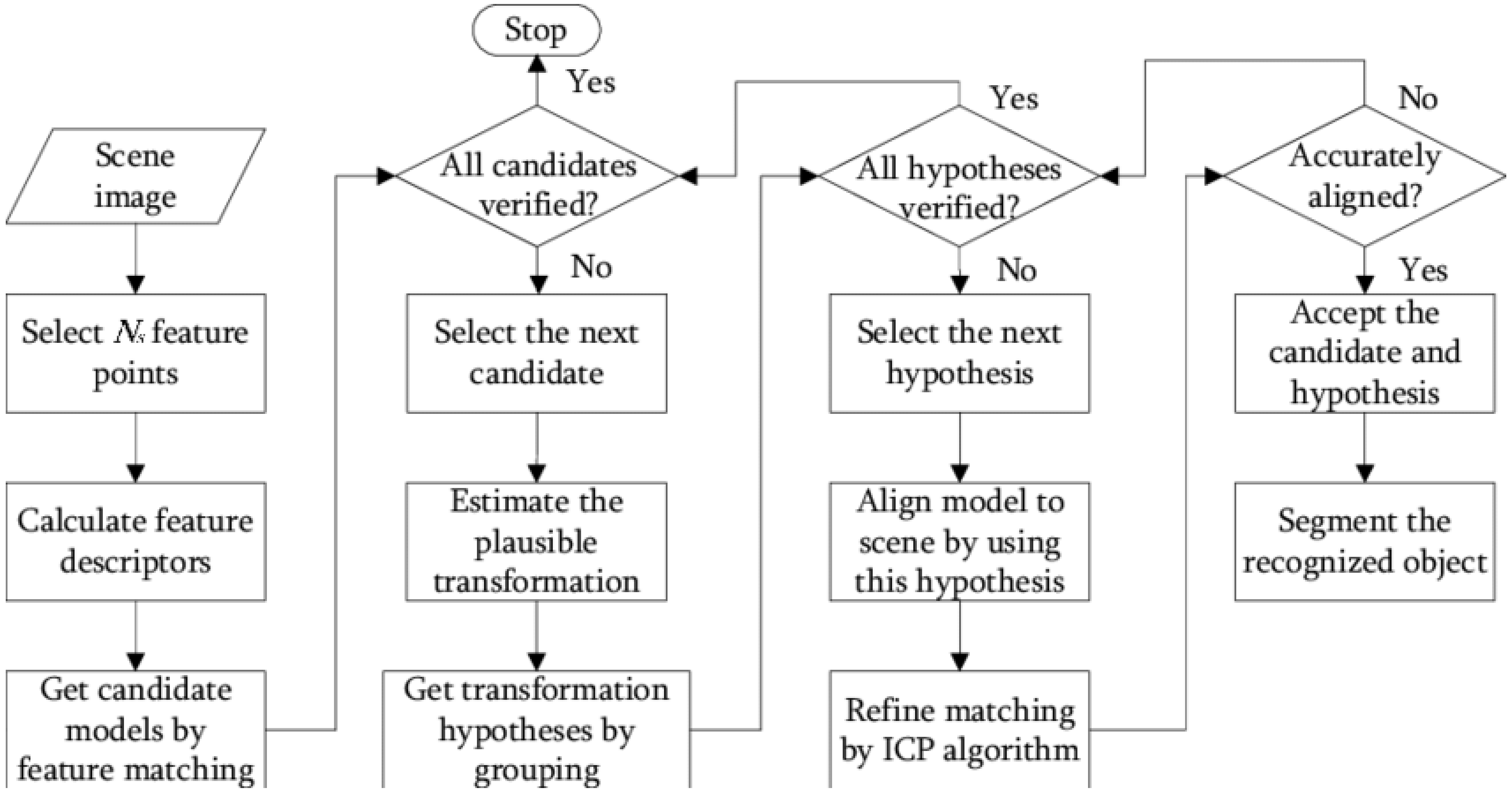
Working Model :-

- Video link : https://github.com/Vrajsingh/flipkard_grid_robotics_2.0



Object Recognition Details :-

- **Flow Diagrams :-**



Scene image
Select N_s feature points
Calculate feature descriptors
Get candidate models by feature matching
All candidates verified?
Yes
Stop
No
Select the next candidate
Estimate the plausible transformation
Get transformation hypotheses by grouping
All hypotheses verified?
Yes
No
Select the next hypothesis
Align model to scene by using this hypothesis
Refine matching by ICP algorithm
Accurately aligned?
No
Yes
Accept the candidate and hypothesis
Segment the recognized object

Vision-based Robotic Grasping (Grasp Planning) :-

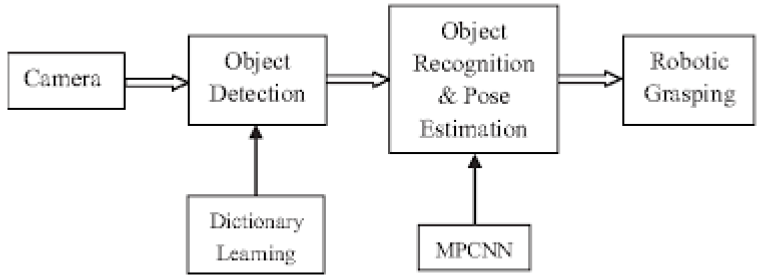
- The essential information to grasp the target object is the 6D gripper pose in the camera coordinate, which contains the 3D gripper position and the 3D gripper orientation to execute the grasp. Within the methods of vision-based robotic grasping, the estimation of 6D gripper poses varies aiming at different grasp manners, which can be categorized into **2D planar grasp** and **6DoF grasp**.

- **2D planar grasp** means that the target object lies on a plane workspace and the grasp is constrained from one direction. The essential information is simplified from 6D into 3D, which are the 2D in-plane positions and 1D rotation angle. There exist methods of **evaluating grasp contact points** and methods of **evaluating grasp oriented rectangles**.

- **6DoF grasp** means that the gripper can grasp the object from various angles in the 3D domain, and the essential 6D gripper pose could not be simplified. Based on whether the grasp is conducted on the complete shape or on the single-view point cloud, methods are categorized into **methods based on the partial point cloud** and **methods based on the complete shape**. Methods based on the partial point cloud contains **methods of estimating candidate grasps** and **methods of transferring grasps** from existing grasps database. Methods based on the complete shape contains **methods of estimating 6D object pose** and **methods of shape completion**. Most of current 6DoF grasp methods aim at known objects where the grasps could be precomputed manually or by simulation, and the problem is thus transformed into a **6D object pose estimation** problem.

Vision-based Robotic Grasping (Grasp Planning) cont. :-

- Besides, most of the robotic grasping approaches require **the target object's location** in the input data first. This involves three different stages: **object localization without classification**, **object detection** and **object instance segmentation**. Object localization without classification only outputs the potential regions of the target objects without knowing their categories. Object detection provides bounding boxes of the target objects with their categories. Object instance segmentation further provides pixel or point-level regions of the target objects with their categories.



COMPONENT DETAILS :-

❑ **SOFTWARE**:-

- **Our software will developed in Ubuntu 16.04. ROS Kinetic :-**

1. **ROS Kinetic** (http://wiki.ros.org/kinetic/Installation/)
2. **Driver for our robot arms from universal robots** (https://github.com/ros- industrial/ur_modern_driver)
3. **Universal Robot package for ROS Kinetic** (http://wiki.ros.org/action/show/universal_robots?action=show&redirect=universal_robot)
4. **MoveIt!** (https://moveit.ros.org/)
5. **Robotiq ROS package** (http://wiki.ros.org/robotiq/)
6. **Mask R-CNN** (https://github.com/matterport/Mask_RCNN)
7. **AprilTag ROS package** (https://github.com/AprilRobotics/apriltag_ros)

COMPONENT DETAILS (cont.) :-

- ❑ **HARDWARE** :- **Robot arm kit (Total cost = Rs. 24,074.94)**

1. **Arduino Uno R3 based Bluetooth + USB 18 Servo Controller (1pc) Rs.2596.0**

   (https://robokits.co.in/control-boards/rc-servo-controller/arduino-uno-r3-based-bluetooth-usb-18-servo-controller)
2. **Metal Gear Standard Servo Economy (6pcs) Rs 2,902.8**

   (https://robokits.co.in/motors/rc-servo-motors/metal-gear-standard-analog-servo-economy)
3. **Multipurpose Aluminium Standard Servo Bracket (4pcs) Rs. 2,265.6**

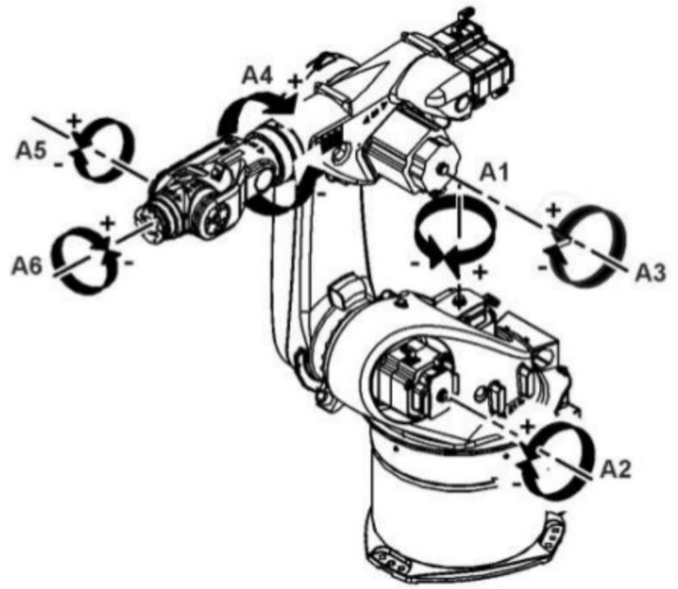
   (https://robokits.co.in/robot-parts/servo-accessories/multipurpose-aluminium-standard-servo-bracket)
4. **Long U Aluminium Servo Bracket (1pc) Rs.354.0**

   (https://robokits.co.in/robot-parts/servo-accessories/long-u-aluminium-servo-bracket)
5. **Short U Shape Aluminium Servo Bracket (2pcs) Rs. 566.4**

   (https://robokits.co.in/robot-parts/servo-accessories/short-u-shape-aluminium-servo-bracket)
6. **L Shaped Interconnect Servo Bracket (2pcs) Rs. 377.6**

   (https://robokits.co.in/robot-parts/servo-accessories/l-shaped-interconnect-servo-bracket)
7. **Parallel Jaw Robotic Gripper (1pc) Rs. 1,003.0**

   (https://robokits.co.in/robot-parts/accessories/parallel-jaw-robotic-gripper)



COMPONENT DETAILS (cont.) :-

❑ **HARDWARE :-**

8. **Modified Computer SMPS for 12V 25A, 5V 20A and 3.3V 15A output (1pc) Rs. 1,672.06**

   (https://robokits.co.in/power-supply/ac-dc-smps/modified-computer-smps-for-12v-25a-5v-20a-and-3.3v-15a-output)

9. **Step Down Module 5A (1pc) Rs. 243.08**

   (https://robokits.co.in/power-supply/dc-dc-power-supply/step-down-dc-dc-adjustable-voltage-regulator-5axl4015e1-based)

10. **Metal Horn for Servo 25T (6pcs) Rs. 2,265.6**

    (https://robokits.co.in/motors/rc-servo-motor/metal-horn-for-servo-25t)

11. **Bearings (3pcs) Rs. 849.6**

    (https://robokits.co.in/mechanical-parts/cnc-router-xyz-axis-part/astro-lmuu-lmluu-bearings/astro-lm6uu-linear-motion-bearings-for-3d-printer-xyz-axis)

12. **Rotating base assembly + DF15RSMG 360 Degree Motor (1pc) Rs.2,847.34**

    (https://www.dfrobot.com/product-1213.html)

13. **Screw nut set (1pc) Rs. 212.40**

    (https://robokits.co.in/3d-printer/accessories/m3-screw-nut-for-adjustable-heatbed-10-pcs-set)

14. **Realsense Camera SR300 Rs. 5,919.46**

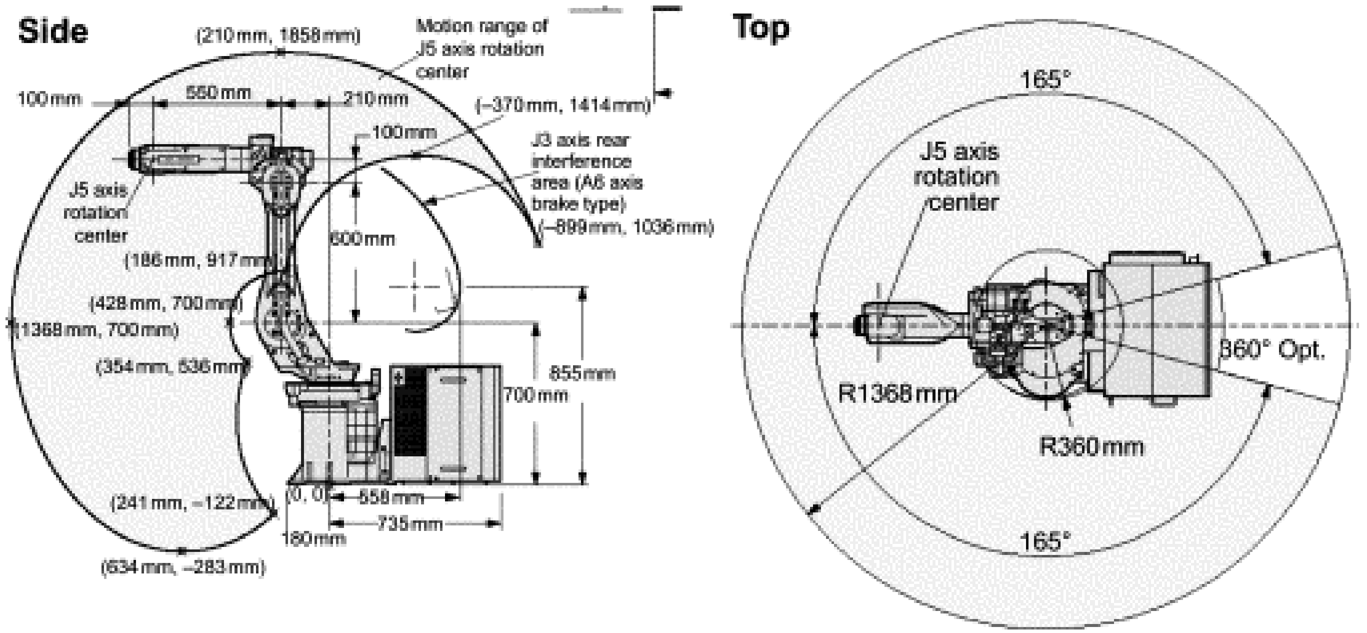
    (https://www.intelrealsense.com/coded-light/)

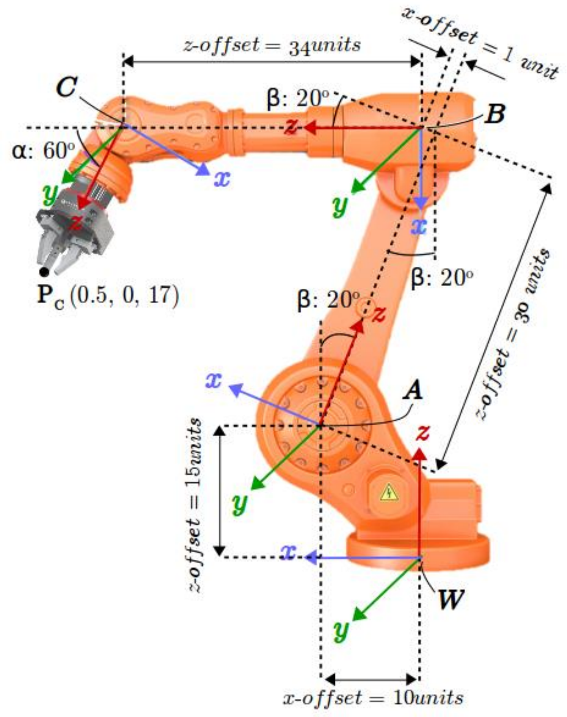
Execution plan with timelines :-

- ❑ Design & Simulation - 1 day
- ❑ Procurement – 5 days
- ❑ Manufacturing, Fabrication and Assembly – 1 day
- ❑ Software development – 1 day
- ❑ Testing – 2 days

- It will take a timeline of 10 days to have a complete working system of Autonomous Object Picking Robotic Arm . In coming slides all the detailed explanation of our execution plan has been explained.

• **<u>Design</u> :-**


Side
(210 mm, 1858 mm)
Motion range of J5 axis rotation center
100 mm
550 mm
210 mm
(−370 mm, 1414 mm)
100 mm
J3 axis rear interference area (A6 axis brake type)
(−899 mm, 1036 mm)
J5 axis rotation center
600 mm
(186 mm, 917 mm)
(428 mm, 700 mm)
(1368 mm, 700 mm)
(354 mm, 536 mm)
855 mm
700 mm
(241 mm, −122 mm)
(0, 0)
558 mm
735 mm
180 mm
(634 mm, −283 mm)
Top
165°
J5 axis rotation center
360° Opt.
R1368 mm
R360 mm
165°


$z\text{-}offset = 34units$
$x\text{-}offset = 1\ unit$
C
B
β: 20°
α: 60°
x
y
z
β: 20°
P_c (0.5, 0, 17)
β: 20°
$z\text{-}offset = 30\ units$
A
$z\text{-}offset = 15units$
W
$x\text{-}offset = 10units$

- **Procurement** :-

➢ We will purchase all the required components through an online platform Robokits (https://robokits.co.in/).

- **Manufacturing, Fabrication and Assembly** :-

➢ Today most robots are used in manufacturing operations.

➢ The applications can be divided into three categories: (1) material handling, (2) processing operations, and (3) assembly and inspection.

1. **Material Handling** : Material handling will include loading and unloading of object which is essential process as per requirement.
2. **Processing Operation** : Processing operation includes the motion of our pick-place-robotic-arm . It should follow the requirement as per the guidelines.
3. **Assembly and Inspection** : After having an exact figure, the last work comes to assemble all the components to have a solid body which will be ready for work. When the structure will get ready then the final step will be to do the inspection part to check whether the system is working as per the requirement or not.

Software Development :-

- The primary code for the project will be in the **ROS node**, IK_server in the kuka_arm **ROS package**, and the arm will be operated in a ROS based simulator environment consisting of **Gazebo**, **RViz** and **MoveIt!.** The IK_server node receives end-effector (gripper) poses from the robot simulator and performs Inverse Kinematics, providing a response to the simulator with calculated joint variable values (joint angles in this case). The IK analysis will be implemented in the IK_server node using **Sympy** and **Numpy** libraries.

❑ The following are selected notes of interest regarding this implementation :-

- All python code conforms to **PEP 8** wherever possible with occasional deviations to accommodate **Sympy** and **Numpy** matrix notations, and where it improves readability. Similarly, function docstrings are conformed to **PEP 257** convention.

- The DH (**Denavit–Hartenberg**) table for the arm robot is constructed from the kr210.urdf.xacro file in the kuka_arm ROS package.

- The gripper or EE orientations are converted to the desired **Euler angles** from the **quaternions** received in the IK **service** request using ROS **geometry transformations module**.

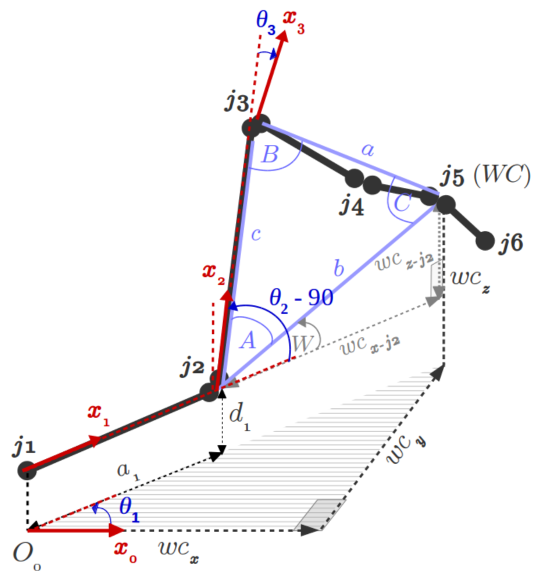
- URDF vs DH frame misalignment in gripper/EE pose is addressed by aligning the URDF and DH EE frames through a sequence of intrinsic rotations: 180 deg yaw and -90 deg pitch.

Software Development (cont.)

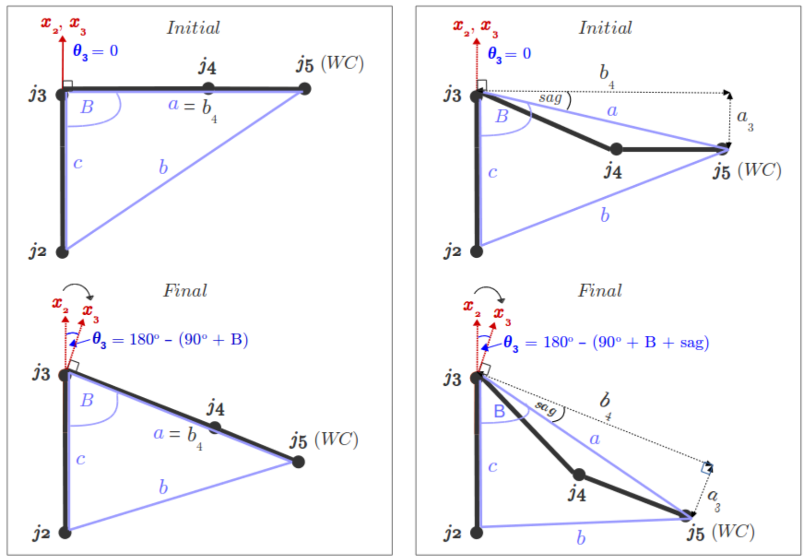
- In the **Sympy** implementation, the DH Table will be substituted in individual transforms *before* composing the overall homogeneous transform between the base frame and the end-effector. This is because it takes considerable calculation time (in the order of several seconds) to symbolically simplify the product of six 4x4 transform matrices each consisting of trigonometric expressions. By substituting some of the values for constants and angles in these trigonometric expressions for each individual matrix, the overall composite homogeneous transform matrix will simplified much faster (down to the order of milliseconds).

- As described in figures (a) and (b), to account for sag in side_a of the SSS triangle (the line segment connecting joints 3 and 5 (WC (wrist centre))) caused by joint 4, first, length of side_a is recalculated, and second, the sag angle formed between y3-axis and side_a will be calculated and accounted for in the calculation of theta_3.

(a)



(b)



(a) no sag ($a_3 = 0$)

(b) sag ($a_3 > 0$)

Software Development (cont.)

- Once the composite Rotation matrix, R3_6, is symbolically evaluated using Sympy for calculating thetas 1, 2, 3, the IK_server node is reimplemented in Numpy to optimize speed:

  - mpmath matrix and trig imports are replaced with numpy
  - dh params (dict keys) are redefined as strings instead of sympy symbols
  - q symbols are replaced with thetas to make joint variables consistent
  - dh params are moved from global scope since joint variables are not immutable like sympy symbols
  - sympy matrices are replaced with numpy matrices
  - dh variable is added to function args and accessed with string keys
  - R0_3 is converted to a float array before passing in numpy.lingalg.inv to compute inverse
  - Remove single and composite transforms (required for fk only)
  - joint angles are updated in dh params dictionary after they are calculated
  - To improve accuracy of theta 2 and 3, both the length of side_a (containing sag) and the sag angle are rounded-off to a high and consistent number of digits

• **TESTING**

- After implementing the IK_server ROS node, the pick-and-place operation can be tested by launching the project in **test** mode by setting the *demo* flag to *"false"* in inverse_kinematics.launch file under /pick-and place/kuka_arm/launch/.

- In addition, the spawn location of the target object can be modified if desired. To do this, modify the **spawn_location** argument in target_description.launchunder /pick-and-place/kuka_arm/launch/ where 0-9 are valid values for spawn_location with 0 being random mode.

- The project will launched by calling the safe_spawner shell script in a fresh terminal :

- `$ cd ~/catkin_ws/src/pick-place-robot/kuka_arm/scripts`
- `$ ./safe_spawner.sh`

TESTING (cont.)

Note: If Gazebo and RViz do not launch within a couple of seconds, close all processes started by this shell script by entering Ctrl+C in each of the sprung up terminals. Then rerun the safe_spawner script.

Once Gazebo and RViz are up and running, and the following can be seen in the gazebo world:

- Robot
- Shelf
- Blue cylindrical target in one of the shelves
- Dropbox right next to the robot

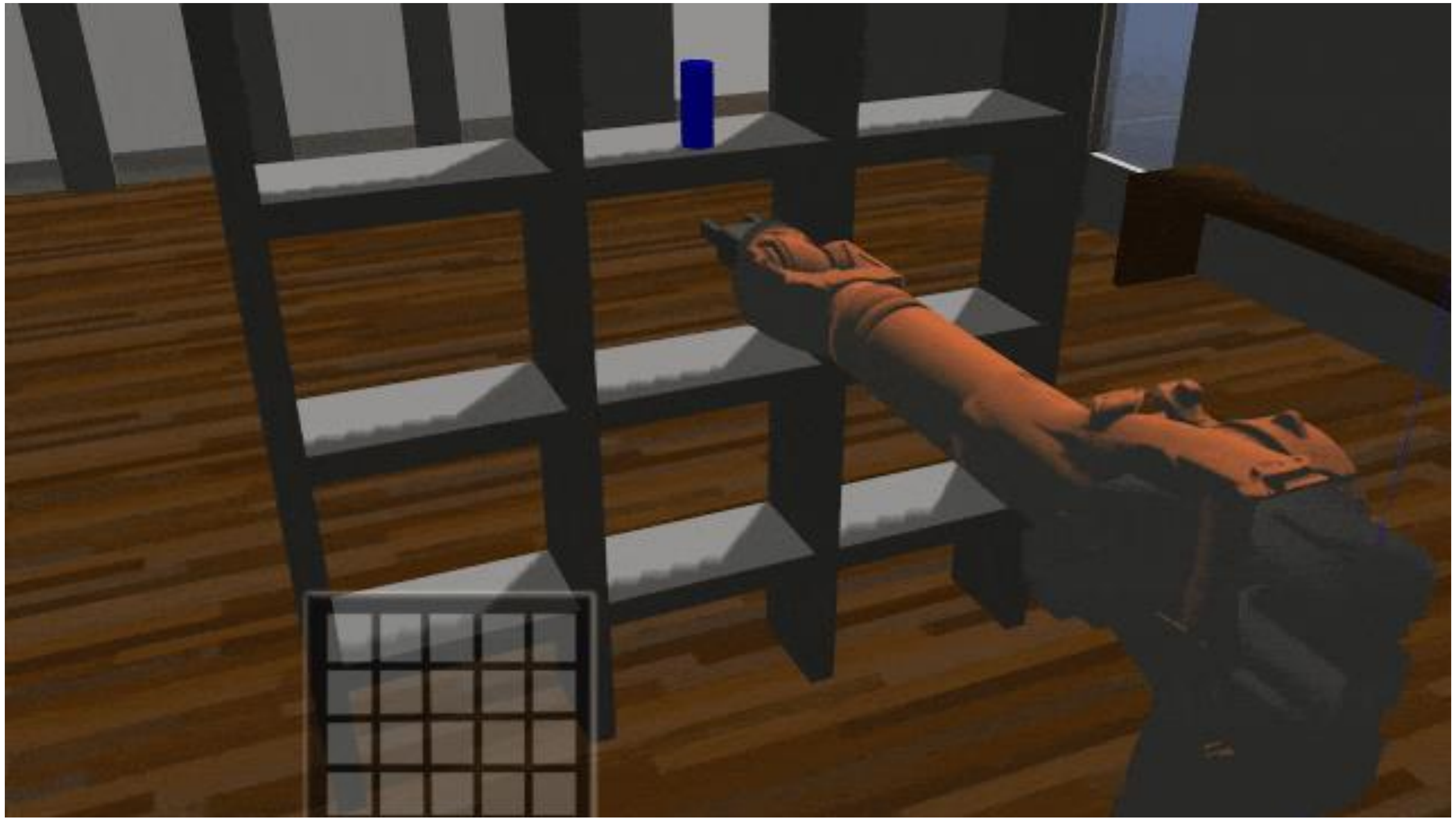
The IK_server ROS node is run from a new terminal window as follows

```
$ cd ~/catkin_ws/src/pick-place-robot/kuka_arm/scripts
$ rosrun kuka_arm IK_server.py
```

With Gazebo and RViz windows arranged side-by-side, the **Next** button on left side of RViz window can be clicked to proceed from one state to another, while the **Continue** button can be clicked to continuously run a complete pick-and-place cycle.
Warning: The terminal window from which the safe_spawner shell script is called needs to be monitored for a Failed to call service calculate_ik error, in which case, all running processes need to be killed and the safe_spawner script called again.

TESTING (cont.)

→ As shown in figure below, the status message in RViz changes as the different stages of pick-and-place simulation are traversed. Actuation is observed in the Gazebo window.
(**PowerPoint slideshow presentation mode is preferred to have a better view of working of below image**)



Current Progress
:-

- **We have already setup Ubuntu 16.04. ROS Kinetic for work environment.**
- **Driver for our robot arms from universal robots is ready.**
- **Universal Robot package for ROS Kinetic is also with us.**
- **MoveIt! Software has been installed in our software.**
- **Robotiq ROS package is ready.**
- **Mask R-CNN is ready.**
- **AprilTag ROS package is also ready.**

➢ **Summary :-** Till now we are done with our software part so the only thing that lefts is to purchase the hardware component. After the purchase work done, we will work on assembling of our robotic arm and will connect it with the software as setup earlier.

Reference :-

- Narong Aphiratsakun. (2015). MT411 Robotic Engineering, Asian Institute of Technology (AIT). http://slideplayer.com/slide/10377412/
- Siciliano et al. (2010). Robotics: Modelling, Planning and Control, (*Springer*)
- Elashry, Khaled & Glynn, Ruairi. (2014). An Approach to Automated Construction Using Adaptive Programing. 51-66. 10.1007/978-3-319-04663-1_4, (*Springer*)
- Yi Cao, Ke Lu, Xiujuan Li and Yi Zang (2011). Accurate Numerical Methods for Computing 2D and 3D Robot Workspace [Journal] // International Journal of Advanced Robotic Systems : INTECH, August 2011. – 6 : Vol. VIII – pp. 1-13.
- Understanding Euler Angles. CHRobotics. http://www.chrobotics.com/library/understanding-euler-angles
- Craig, JJ. (2005). Introduction to Robotics: Mechanics and Control, 3rd Ed (*Pearson Education*)
- Salman Hashmi https://github.com/Salman-H

THANK YOU