Hello Everyone !

Implementation of Pick and Place Application Using Computer Vision Techniques

Ali Jnadi
a.jnadi@innopolis.university

Walid Shaker
w.shaker@innopolis.university

Supervised by:
Prof. Adil Khan
TA: Karam Almaghout

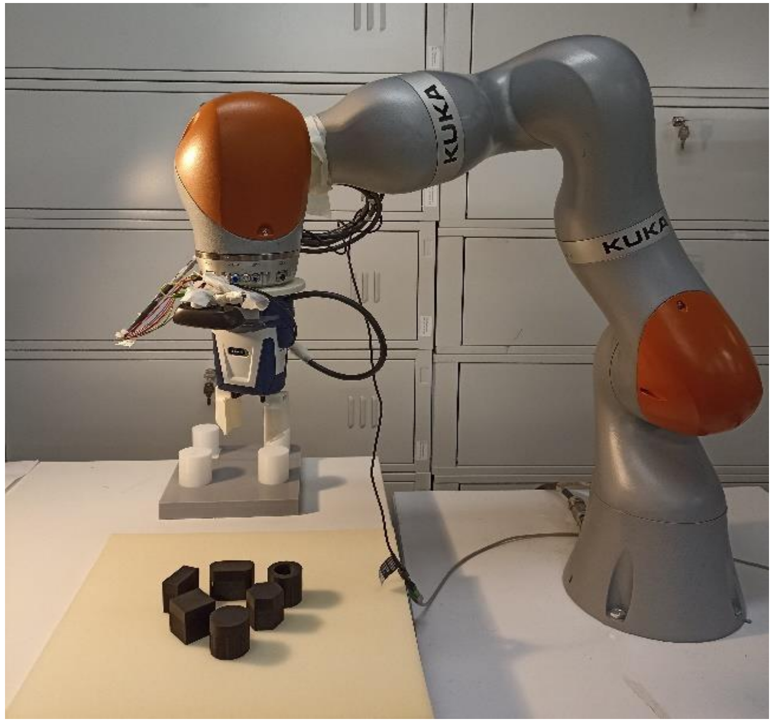
Outline

- Problem Description
- Project Challenges
- Project Significance
- Workplace Setup
- Dataset
- Project Implementation
- Demo

Problem Description

Robots use computer vision in many operations.

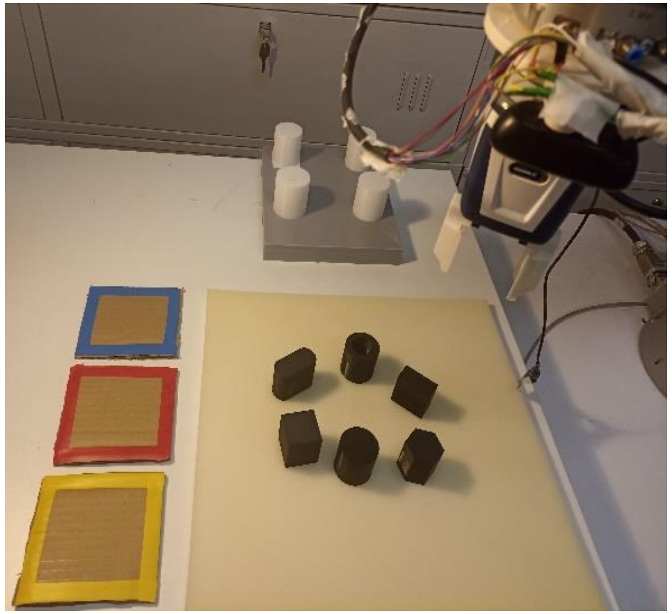
In this project, we are going to address one of these operations, pick and place, which is the most widely used in production lines and assembly processes.

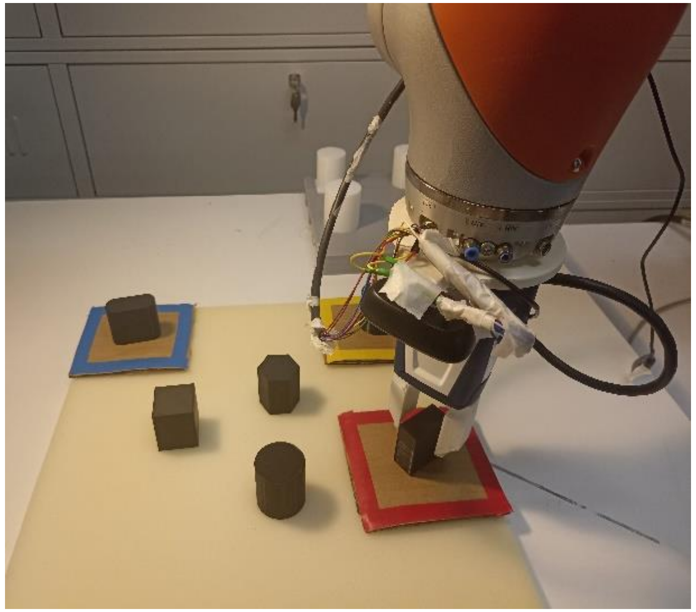


Work environment including manipulator and objects.

Problem Description

The aim of this project is to select 3 objects out of 6 objects in robot work environment, and place them in specific-colored regions, so that each object will be placed in colored region initially specified by the operator.





On the left image, an illustration of the colored regions. On the right, an example of placing operation.

Project Challenges

The challenges of this project go as follow:

- **Object Detection:** *YOLOv4*
- **Box Detection:**
  - *Color Segmentation in HSI*
  - *Morphology operations*
- **Pick and Place:**
  - *Object center and orientation*
  - *Box center and orientation*
- **Localization w.r.t Robot Base:**
  - *Eye-In-Hand Camera Calibration*
  - *Transformations*
- **Hardware Implementation:**
  - *KUKA iiwa*
  - *ROS*

Project Significance

The significance of solving these challenges is as follows:

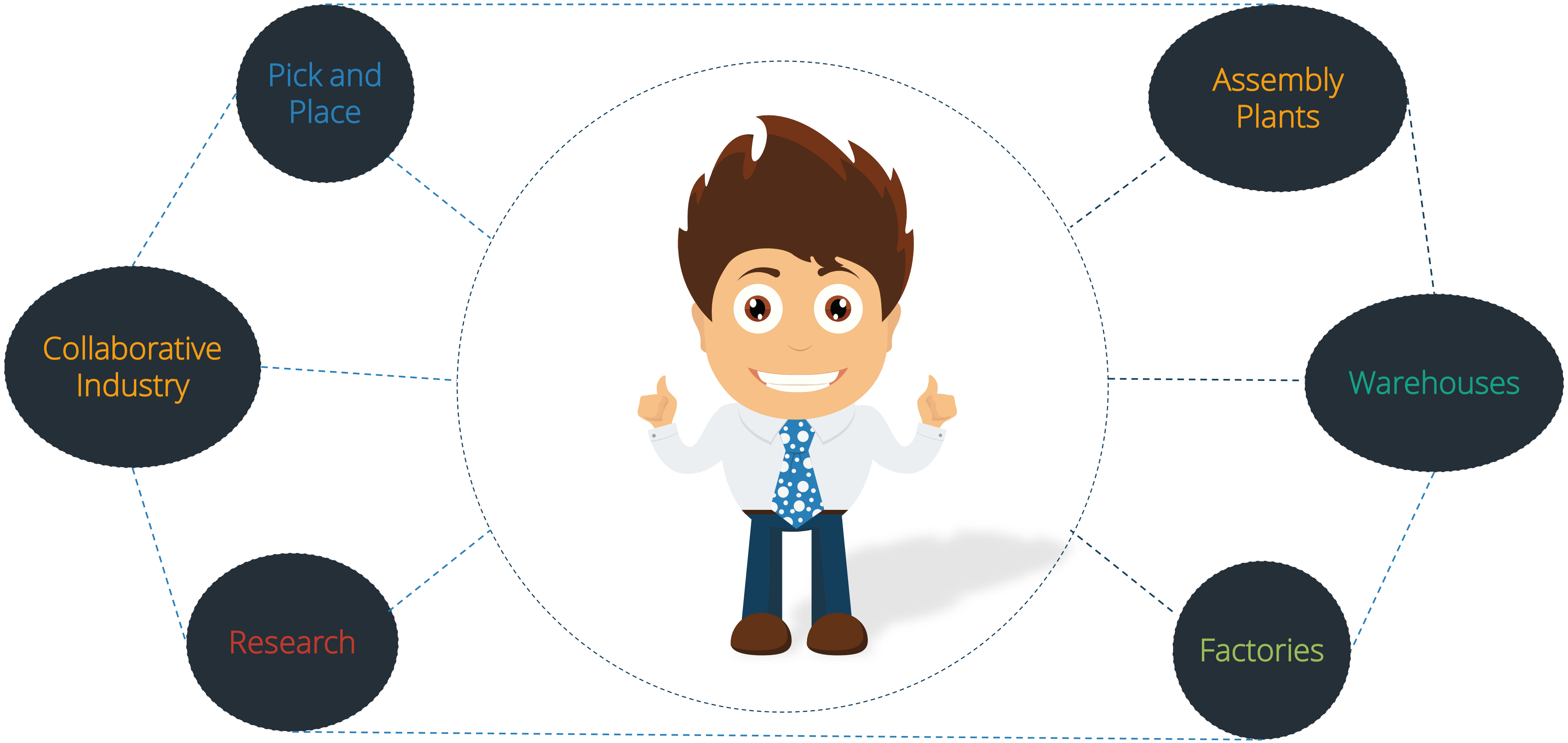
1. Robust and precise pick and place, implementation.

2. Ensuring the system reliability will help in decreasing the error percentage at the detecting stage.

3. Saving time and increasing the productivity of the assembly operation.

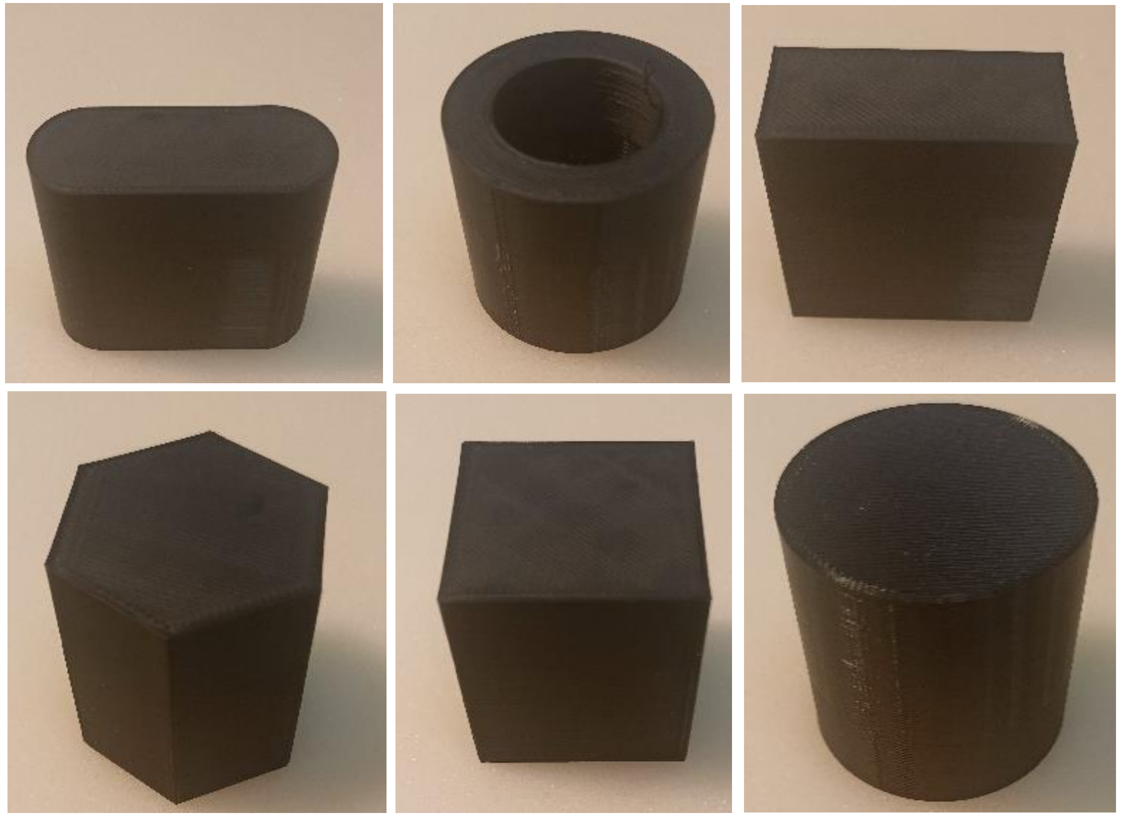
4. Also, ensuring flexibility is quite significant, since the parts and their related regions might change for another task.

Applications



Workplace Setup

3D model is designed and then 3D printing technology is used to print the objects.



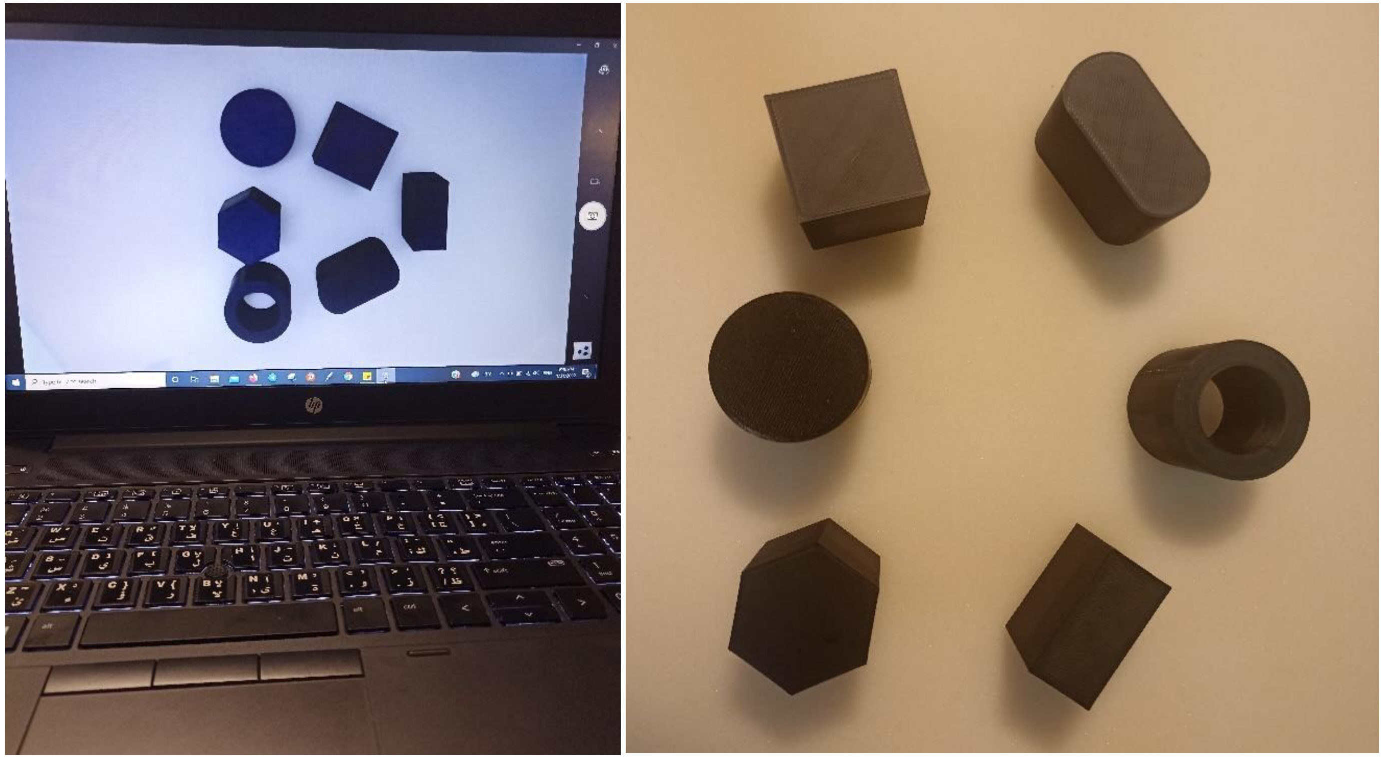
Objects slot, hollow cylinder, rectangle, hexagon, cube, cylinder.

Dataset

Dataset Collecting

100 images have been taken for different positions, orientation, and grouping of the objects.

Initially multiple images are taken for each object individually.

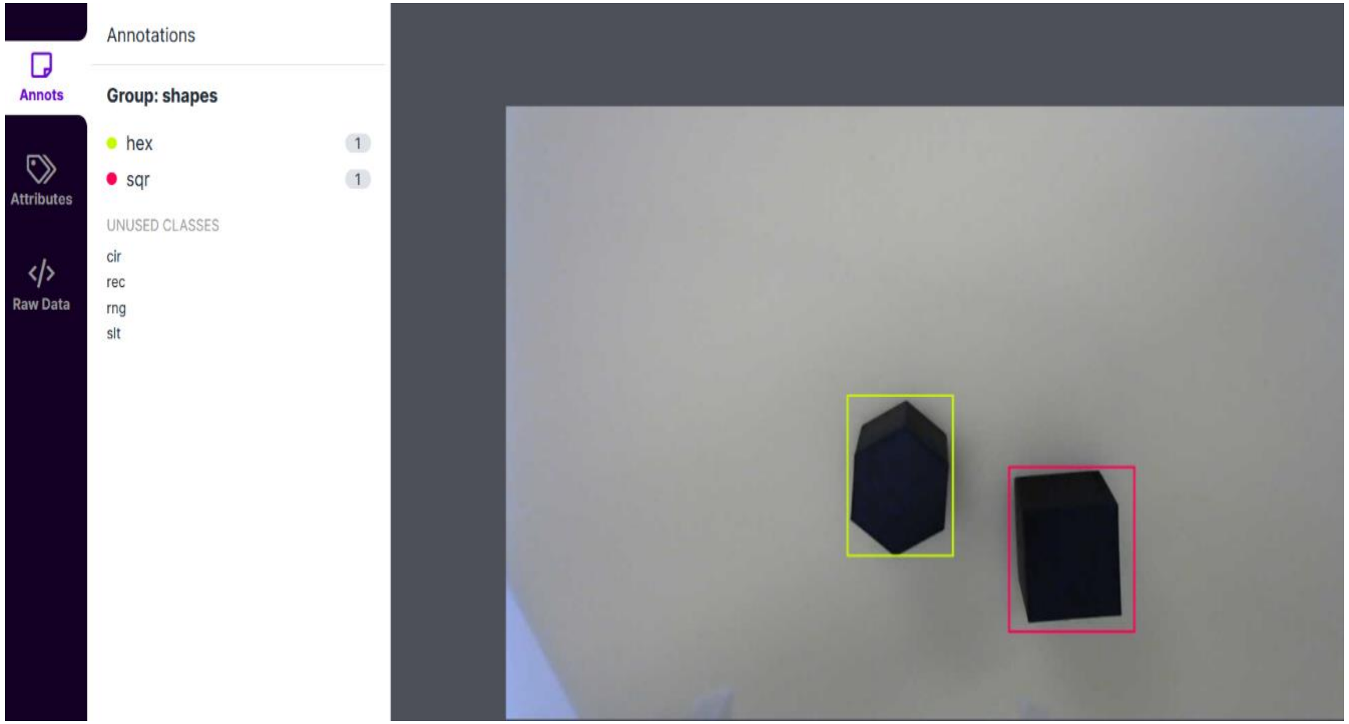


Random shot of objects group. Left image shows the camera view, and right image shows the environment.

Dataset

Dataset Labelling

After that dataset are uploaded to **roboflow** for objects labelling.



Objects Labelling

Dataset

Dataset Preprocessing and Augmentations

TRAIN / TEST SPLIT

Training Set	87%	207 images
Validation Set	8%	20 images
Testing Set	5%	11 images

PREPROCESSING

Auto-Orient: Applied

Resize: Stretch to 320×180

AUGMENTATIONS

Outputs per training example: 3

Flip: Horizontal, Vertical

90° Rotate: Clockwise, Counter-Clockwise, Upside Down

Rotation: Between -15° and +15°

Images pre-processing, augmentation, and train/test set.

Project Implementation

Object Detection

1. Training

YOLOv4 model and **YOLOv4** tiny model are trained.

	YOLOv4 darknet	YOLOv4 tiny-darknet
Mean average precision mAP	95.83 %	95.88 %
Number of iterations	2000	12000

2. Testing

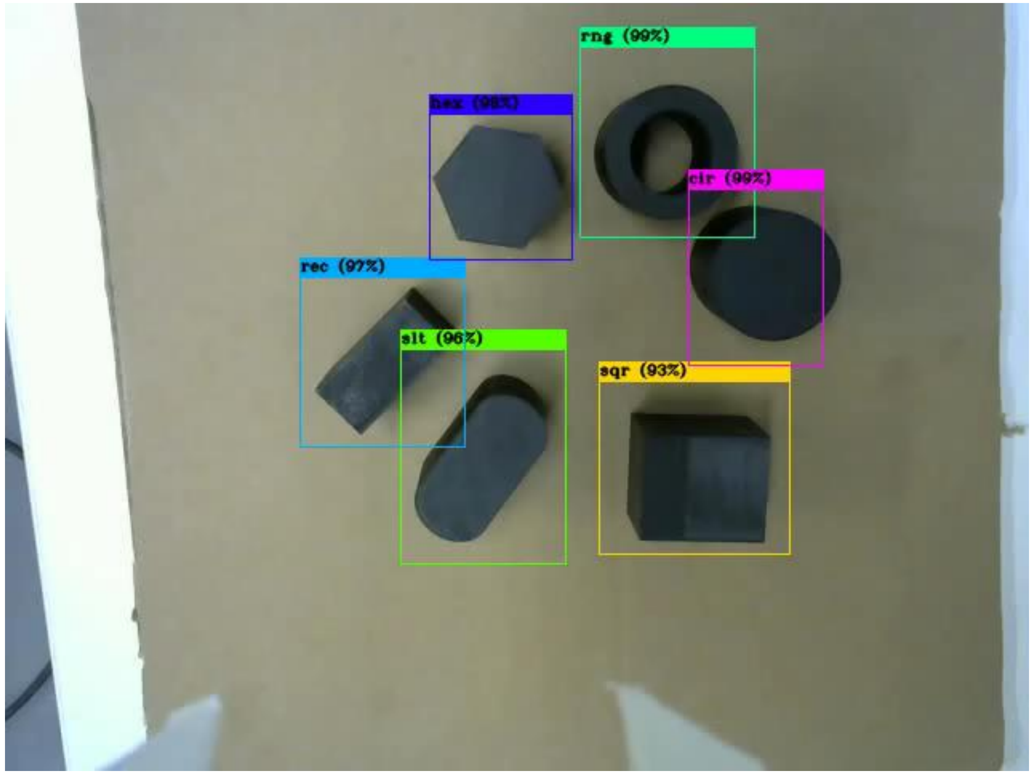
Network input size	YOLOv4 darknet		YOLOv4 tiny-darknet	
	FPS	Overall Accuracy	FPS	Overall Accuracy
416×416	7.2	High	7.5	Normal
256×256	15	Low	15	High
512×512	15	Low	15	Low

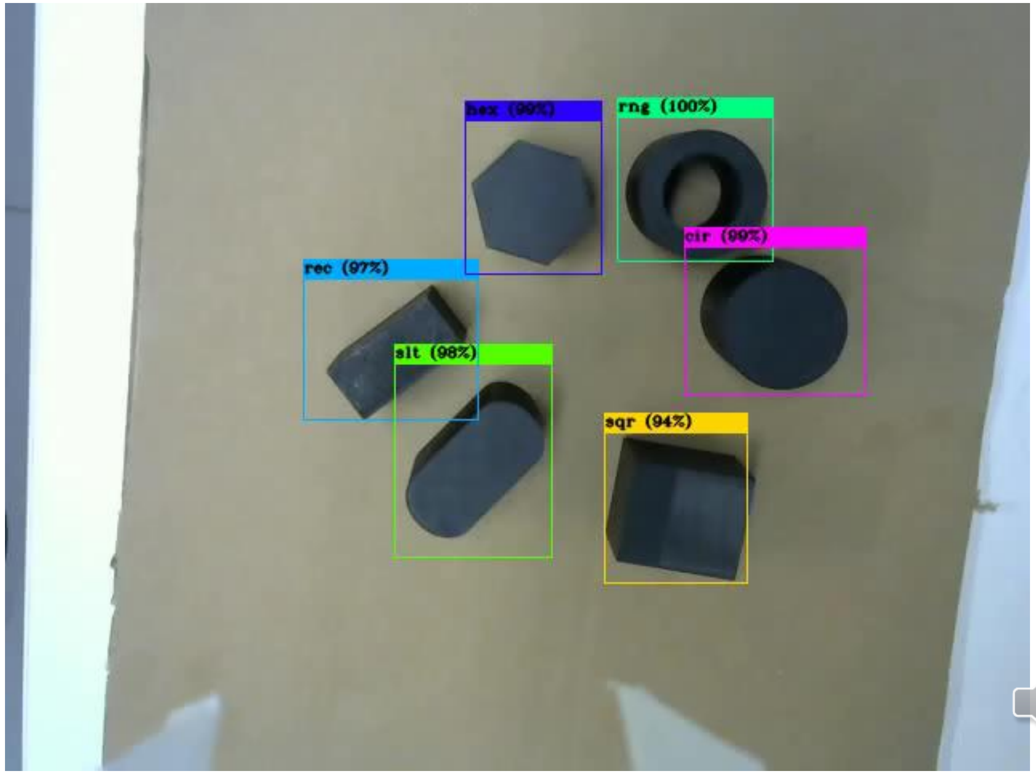
Object Detection

2. Testing

Problems arise due to illumination change, camera orientation, false positive results (detect the gripper as an object, detect background as an object).

Solution: Applying thresholding on detection confidence score.

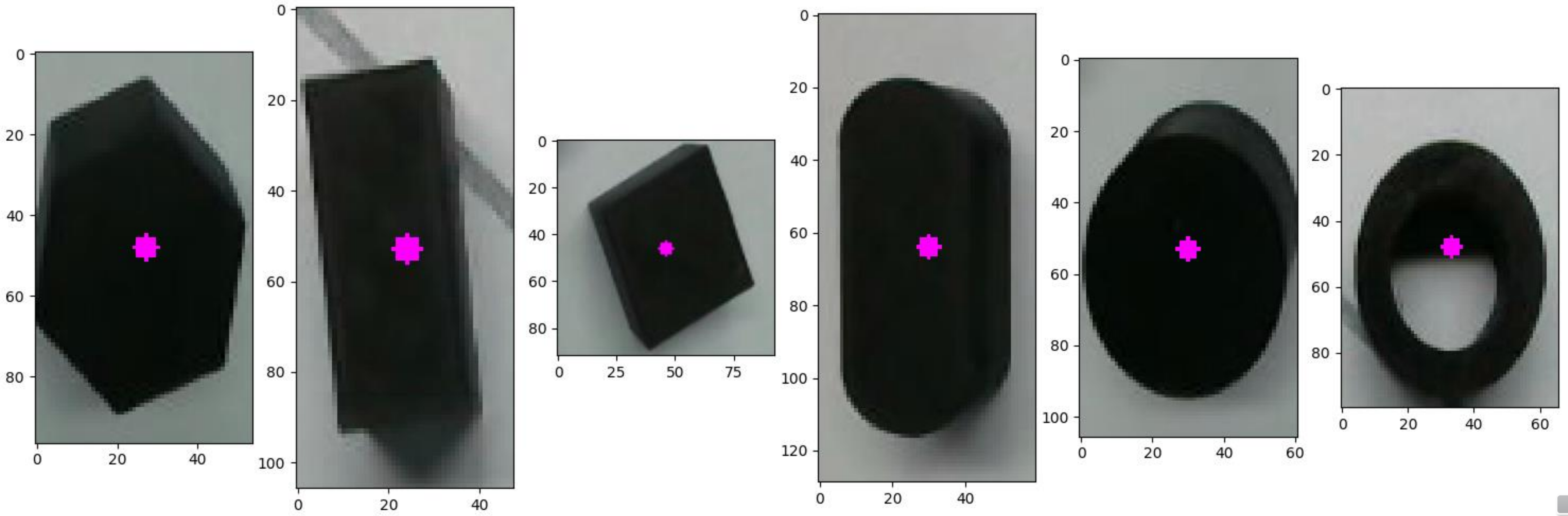




Object Center & Orientation

1. Object Center

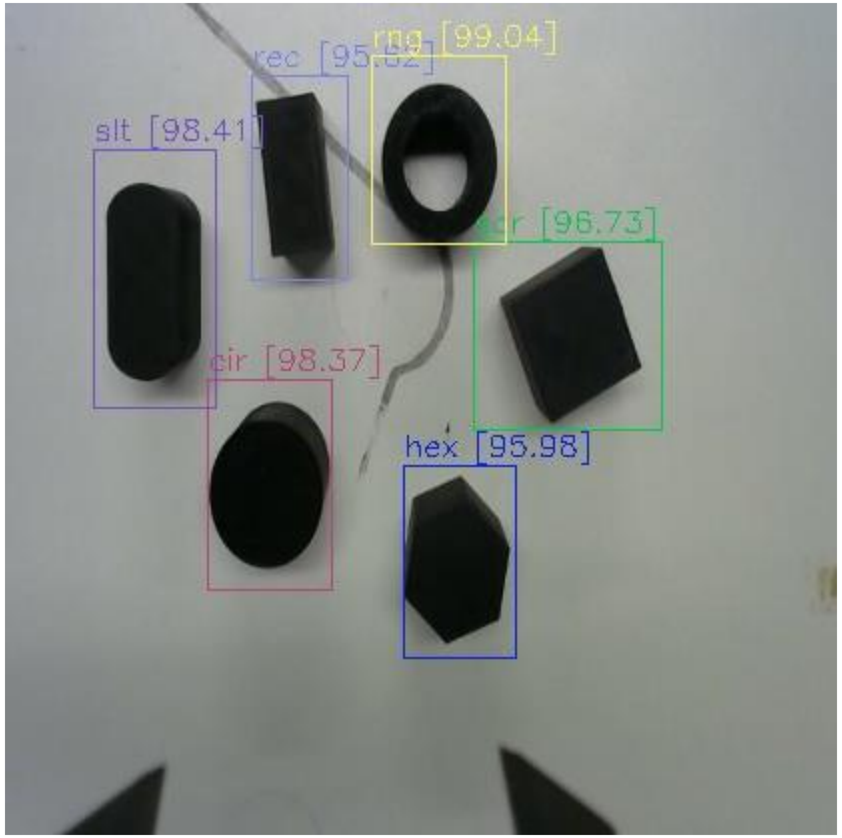
After detecting the object, object center is located w.r.t to image frame using the bounding box obtained from the detection.



Object Center & Orientation

1. Object Center

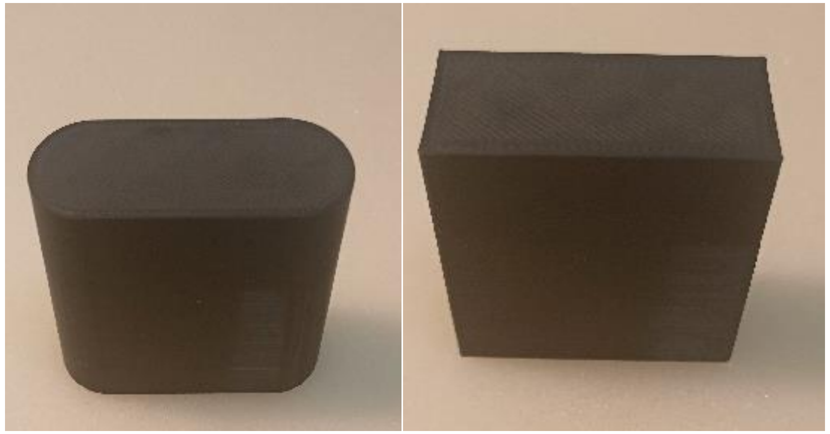
After detecting the object, object center is located w.r.t to image frame using the bounding box obtained from the detection.



Object Center & Orientation

2. Object Orientation

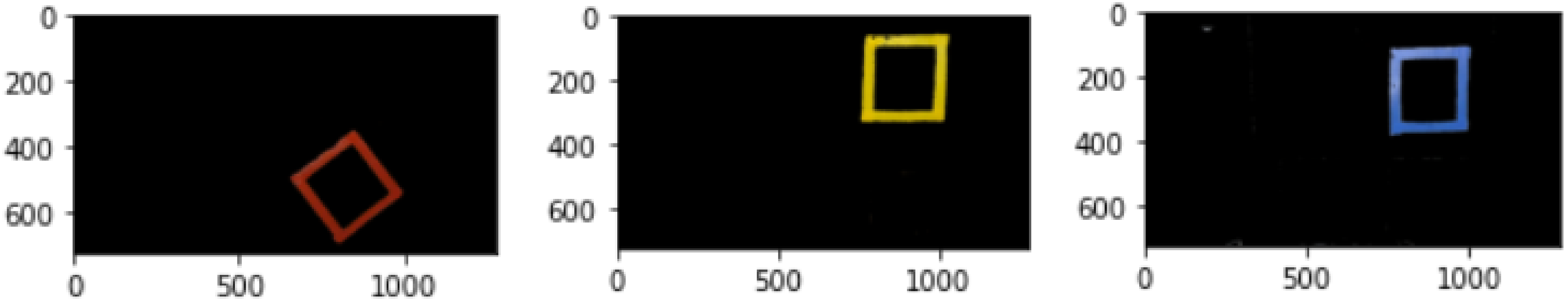
Determining the object orientation is extremely important for the picking task. It is quite easy to pick the circular shapes, but for the slot and parallelepiped which have different dimensions, the task might be challenging.



Solution: Gripper should be aligned with the orientation of the shape

1. Calculating angel of rotation around horizontal axis using small edge points.
2. Then add 90° degree as the gripper horizontal axis should be aligned with perpendicular to the short edge, which is the longest one.

Box Detection



Nested loop Vs. Mask with bitwise_and ?

This is how computer vision tools are powerful ! Using nested loop takes about 7 seconds for execution, while using mask with bitwise_and performs the segmentation in 1 second.

```
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        if (red[i,j]>=28) and (red[i,j]<=135)
        and (green[i,j]>=64) and (green[i,j]<=159)
        and (blue[i,j]>=144) and (blue[i,j]<=238):
            pass
        else:
            img[i,j] = 0
```

7 seconds

```
mask = cv2.inRange(img, lower, upper)
output = cv2.bitwise_and(img, img, mask=mask)
```

1 seconds

Box Detection

1. Color segmentation in HSI space

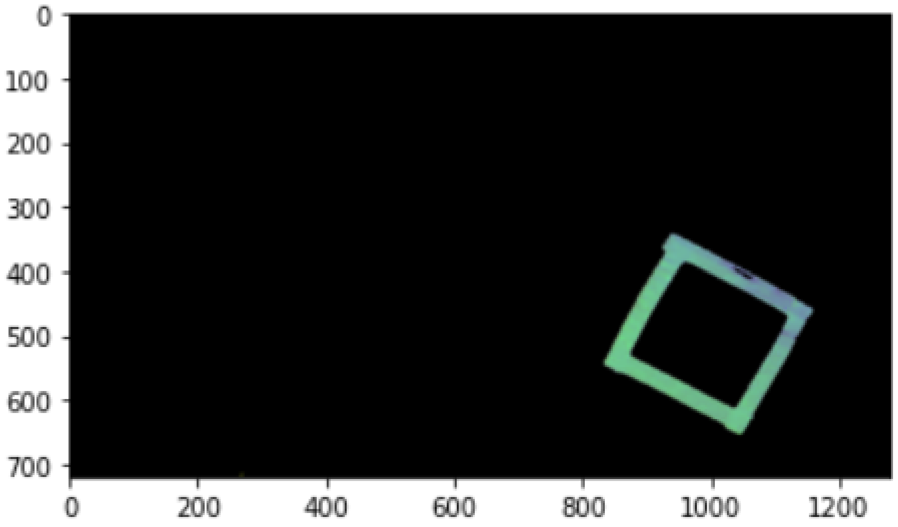
Color segmentation is implemented in HSI color space as the HSI separates the luminance and the chrominance and shows a reliable result compared to RGB space.

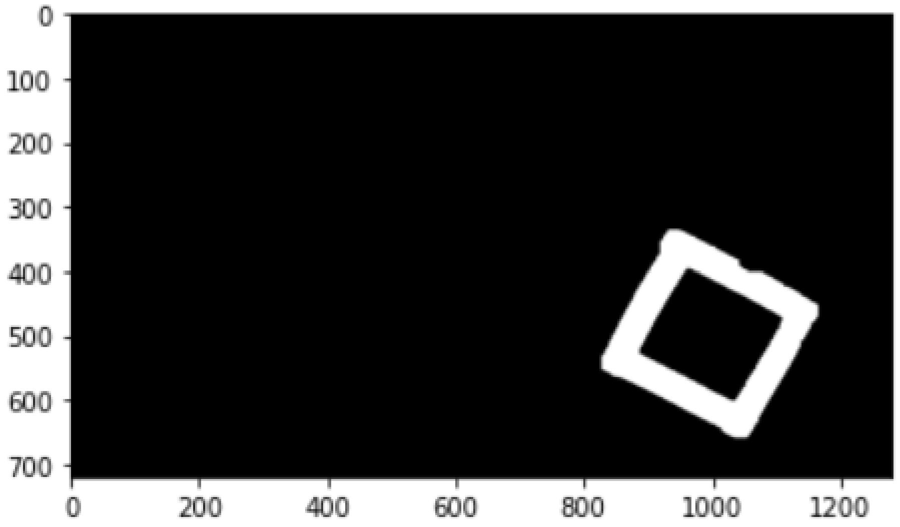
2. Morphology operations

Erosion >> kernel (5,5) , iterations = 1

Dilation >> kernel (5,5) , iterations = 5

Erosion >> kernel (5,5) , iterations = 1

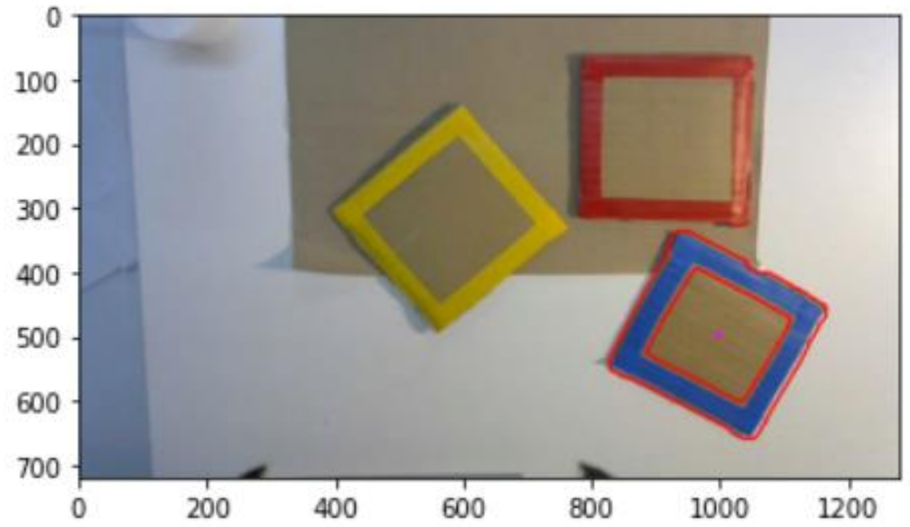


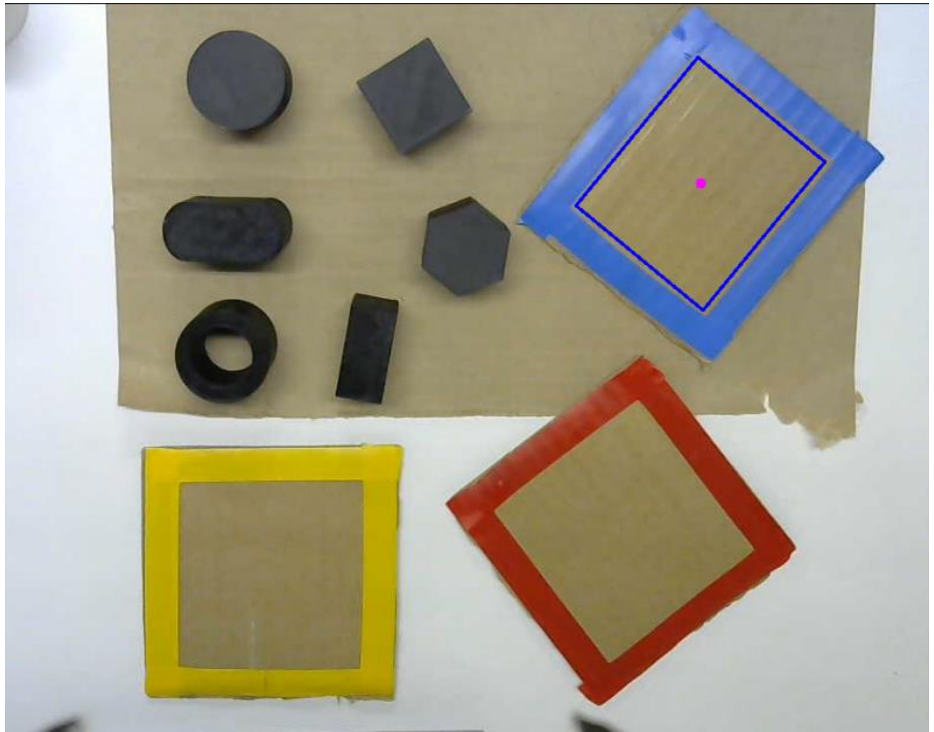


Box Center & Orientation

1. Box Center

Using contours finding algorithm, we detected possible contours for each colored-region. Better results are obtained by thresholding contours area and getting rectangular contour. Center is calculated through **moments** function which use the first moment of area technique.



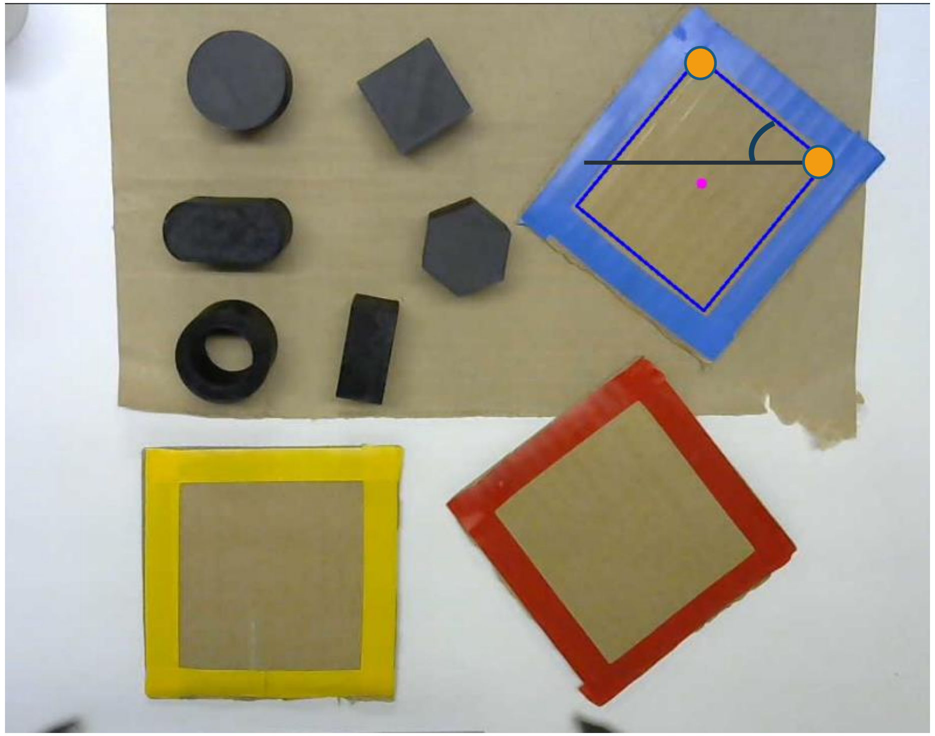


Box Center & Orientation

2. Box Orientation

To make the robot gripper orientation matches the colored-region orientation, angle is calculated via arc tangent (slope).

Box_angle = np.arctan2(upper_right_y - upper_left_y, upper_right_x - upper_left_x)



Localization w.r.t Robot Base

After localizing the object/box center in image frame, it should be transformed to robot coordinate base frame.

1. Camera Intrinsic Matrix ${}^{C}_{O}T$

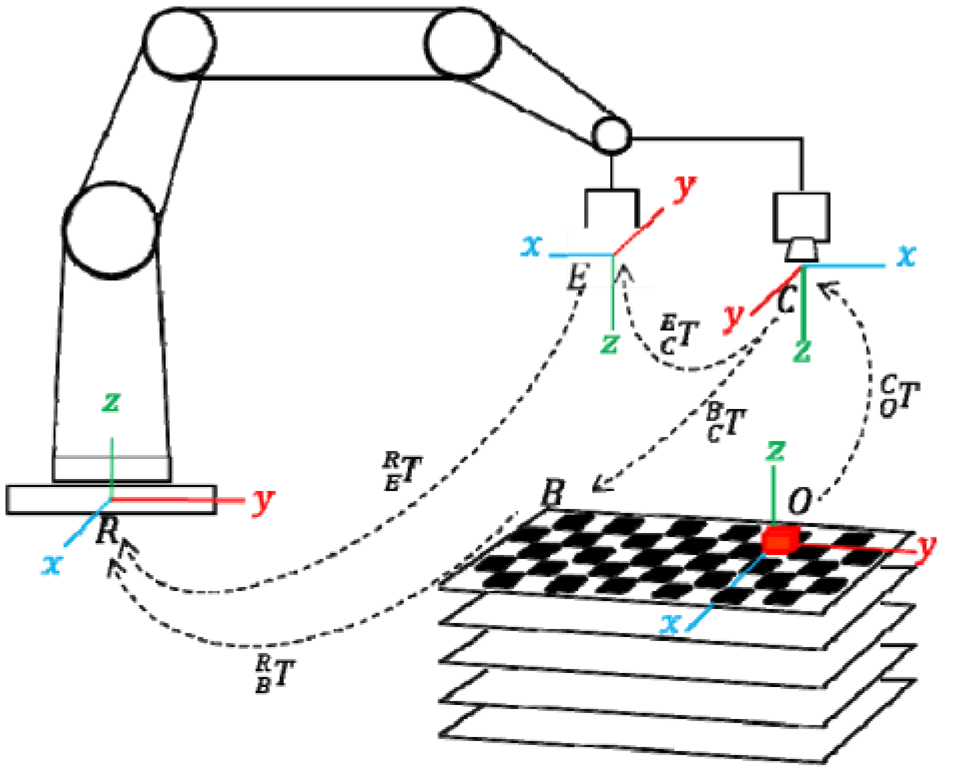
Multiplying the pixel frame vector by camera intrinsic matrix transforms the pixels to camera frame.

2. Eye-In-Hand Camera Calibration ${}^{E}_{C}T$

Eye-In-Hand matrix transforms from end-effector frame to camera frame.

3. Transformation ${}^{R}_{E}T$

Transformation from robot base frame to end-effector frame is calculated using robot current state (position and orientation).



Source: Object Localization and Depth Estimation for Eye-In-Hand Manipulator using Mono Camera

Hardware Implementation

KUKA iiwa

In this stage, we control KUKA iiwa robot with ROS for the real implementation.

Demo

This the project demo (4 min) in which the whole process is of objects and colored-box detection is implemented.

Slot shape == **Blue** region

Ring shape == **Yellow** region

Rectangle shape == **Red** region

We are looking forward to receiving your prestigious feedback for possible improvements/developments that could be applied as we are so enthusiastic to publish a research paper in this track.

Thank You !