National Technical University of Athens
School of Mechanical Engineering
Section of Manufacturing Technology

Robotic Arm Manipulation for Object Detection & Grasping in Occlusion Environments Using Machine Vision & Neural Networks

Diploma Thesis

Pavlos Chionidis

Supervisor: Prof. Panorios Benardos (NTUA)

Table of Contents

01 Introduction

System Parameters & The Problem

Why simulate on 3D environment?

- Easily adaptable.
- Cost and worry free.

Why use CNNs rather than typical image processing?

- Wider span of item identification.
- Highly adaptable.
- More robust in diverse environments.
- Can handle the occlusion problem.

Why use an RGB-D camera?

- Low-cost solution.
- Dense information from a single unit.

How to implement an automated robotic system?

- Use additional sensors.
- Implement logic to drive the system.

Stäubli RX90L

- Six Degrees of Freedom.
- Maximum carrying capacity: 6 kg.
- Equipped with inhouse gripper.

Kinect V2

- RGB Camera $[1920,1080]_{w,h}$.
- Depth Camera $[512,424]_{w,h}$.

Methodology

Robotic Arm Manipulation

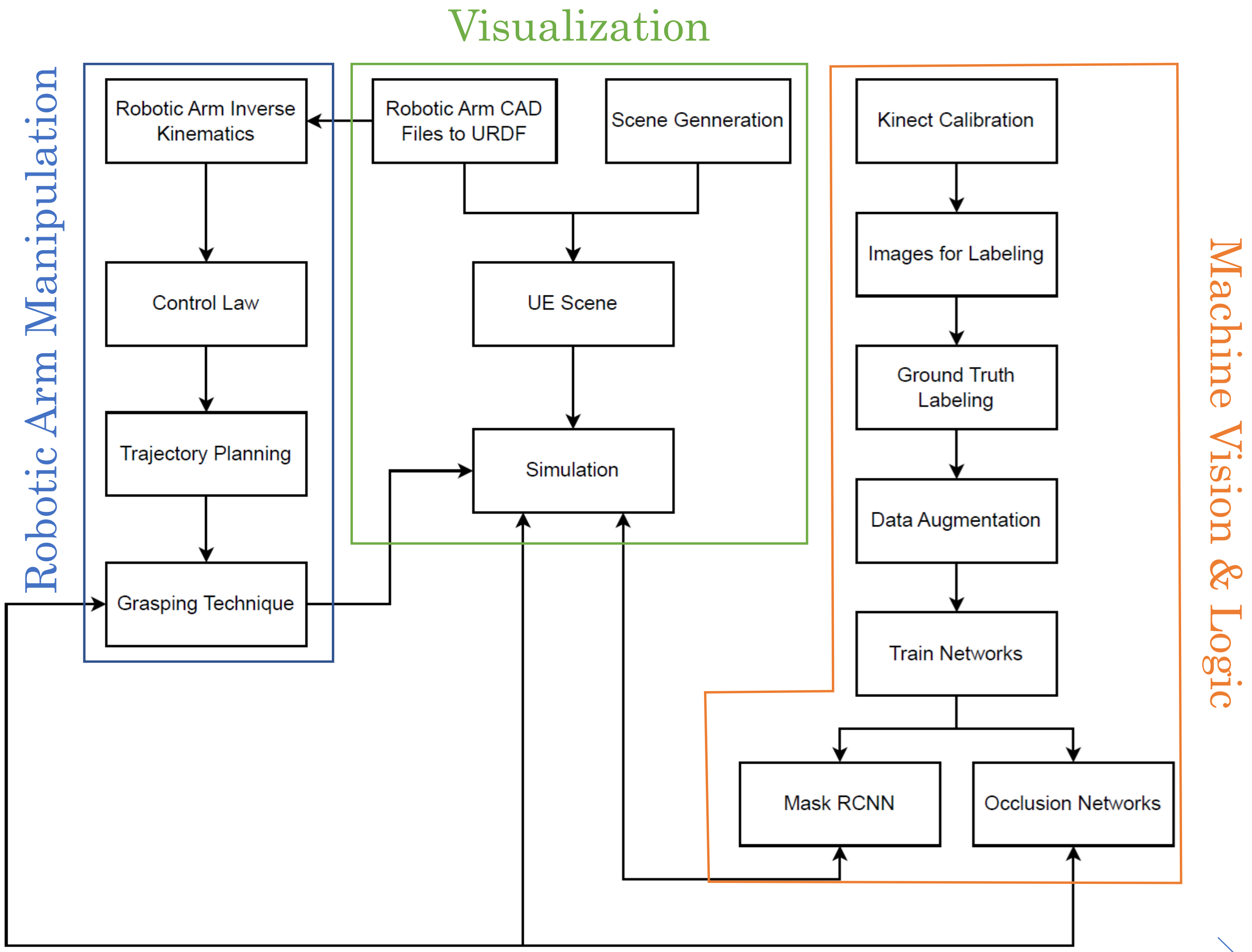
- How much to move?
- Where to move?
- How to move?

Visualization

- How to visualize the robotic arm?
- How to visualize the scene?
- How to simulate the model?

Machine Vision & Logic

- How to calibrate the Kinect?
- How to label images?
- How to train the networks?



02 Theory of Robotics

Transformations

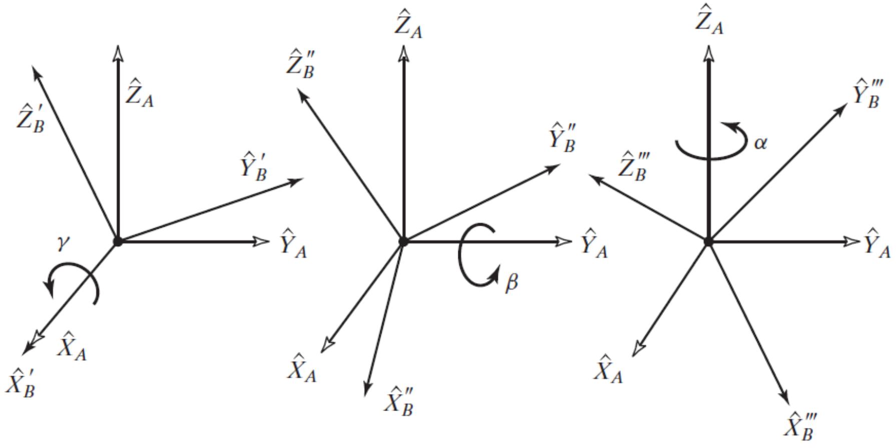
Description of a Body in Cartesian Space.

- Position: [x,y,z]
- Orientation: $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$

Transformation Matrix: $T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Rotation Matrix

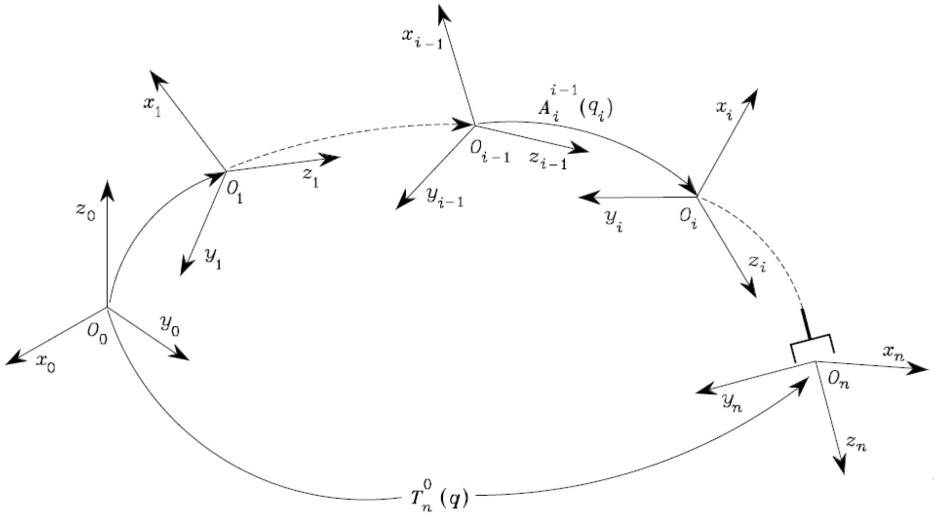
- Euler Angles (More Intuitive): Three angles.
- Euler Parameters (More Robust): Four Parameters



Euler Angles "XYZ".

Open Kinematic Chain

$$^{0}T_n = {}^{0}T_1 \cdot {}^{1}T_2 \cdot \ldots \cdot {}^{n-1}T_n$$

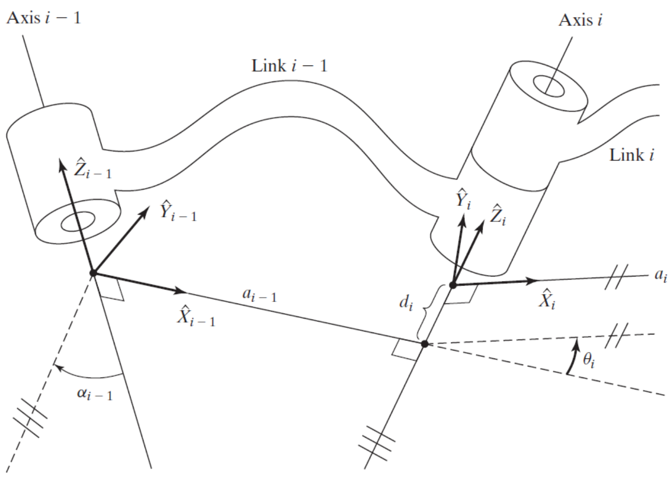


Open Kinematic Chain.

Kinematics

Transformation Matrices

- Denavit & Hartenberg Parameters
- Included inside the URDF File.
- Are dependent on rotation angles of the joints.



Joint frames between a link.

Forward Kinematics

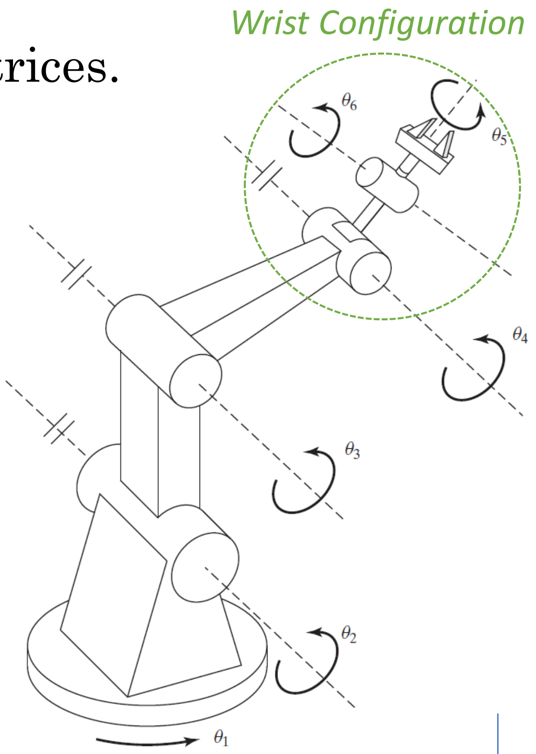
- What is the pose of the robotic arm for a specified joint rotation?
- Computed using transformation matrices.

Inverse Kinematics

- What are the required joint rotations for a specific end effector configuration?
- Hard to Calculate:

$$^{0}T_{e,desired} = X(q_1, q_2, \dots, q_n)$$

- More than one solution may exist.



Robotic Manipulator with wrist configuration.

Pieper Method

- ✓ Split the problem in two different 3DoFs Robotic arms.
- ✓ MATLAB: analyticalinversekinematics()

03 Theory of CNNs

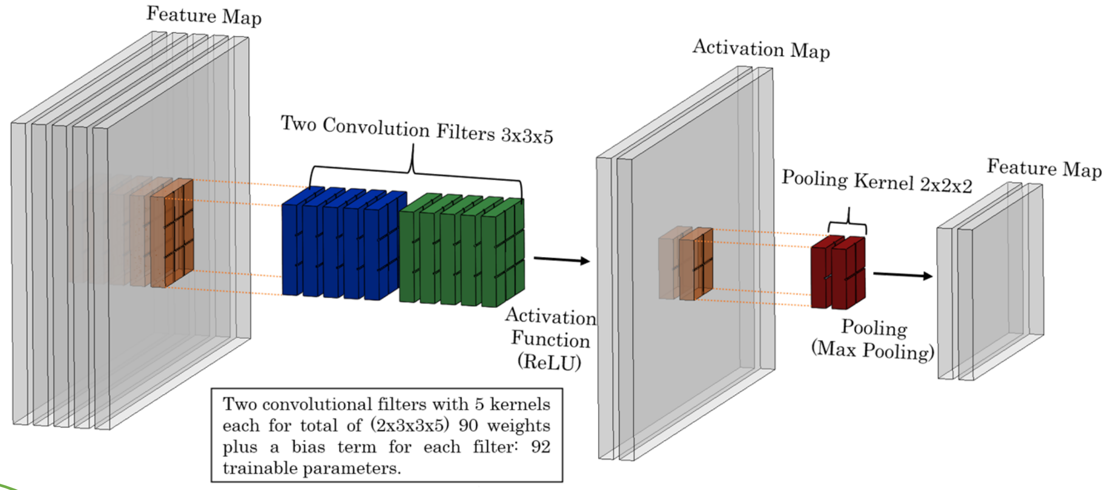
Convolutional Neural Networks

Feature Extractor (Backbone)

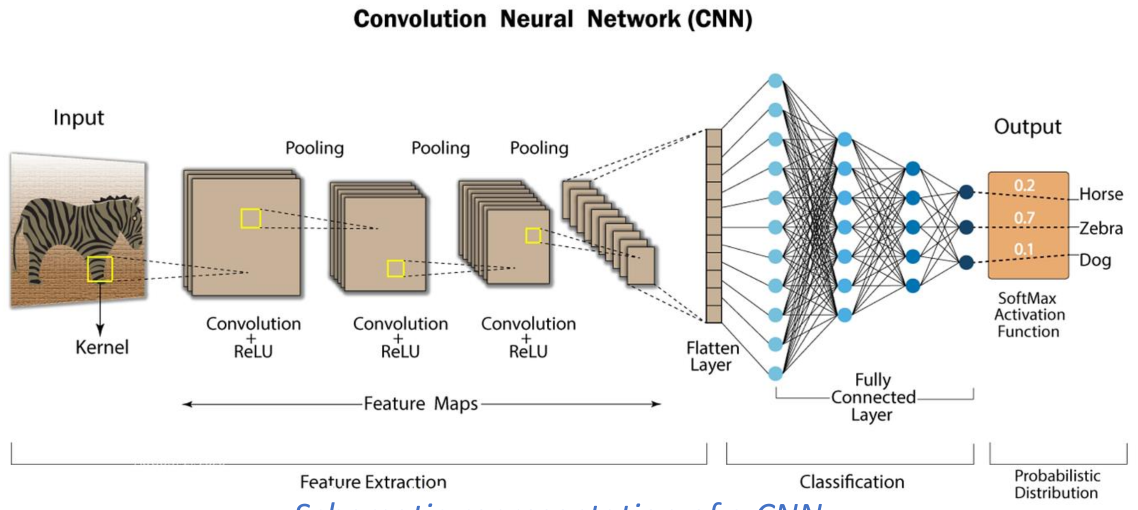
- Extracts features to identify each class
- Consist of many convolutional blocks

Convolutional Block

1. Convolution Operation
2. Activation Function (ReLU)
3. Pooling (Max Pooling)



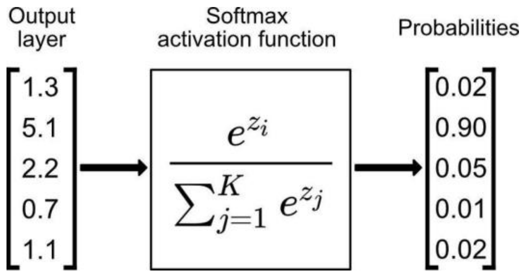
Example of a convolutional block.



Schematic representation of a CNN.

Classifier (Head)

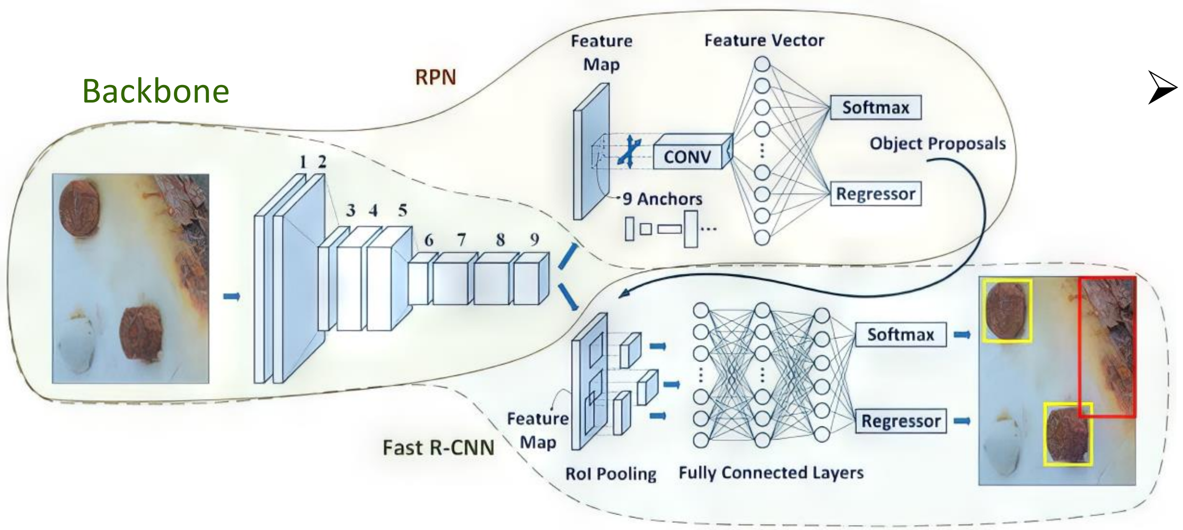
- Flattens the Feature Map
- Passes through Fully Connected Layers
- SoftMax Function to Extract Probabilities of Classes



Object Detectors - Faster RCNN

Faster RCNN

- Feature Extractor (Backbone)
- Region Proposal Network (Object Detector)
- RoI Pooling Layer
- Classifier with Box Regressor (Head)



Faster RCNN schematic representation.

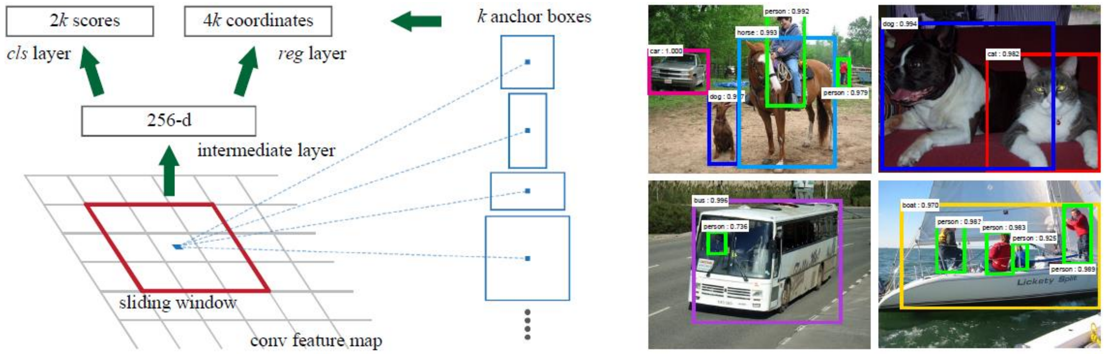
Region Proposal Network

Classifier (cls)

- Detects whether an object is within the anchor box.
- Binary output.

Box Regressor (reg)

- Generates coordinates deltas for each anchor box [x,y,w,h]



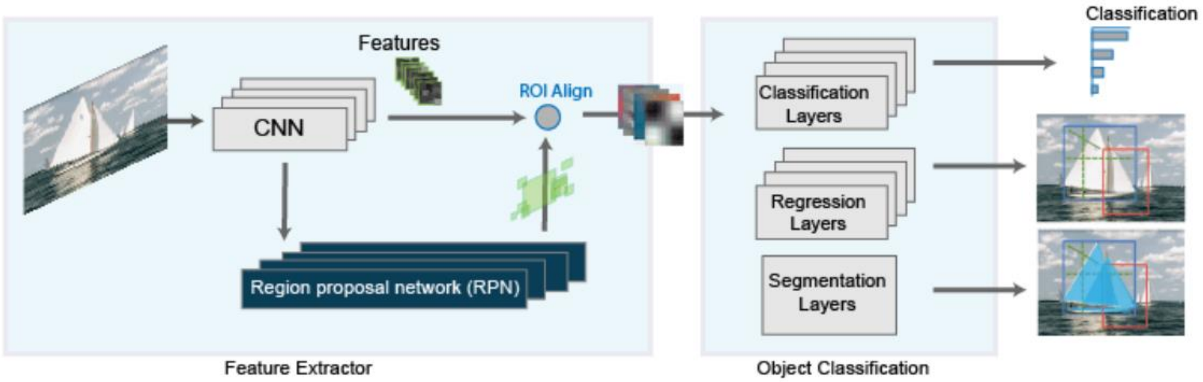
RPN schematic representation.

Anchor Boxes are set in the initiation stage of the network and remain constant. The Box Regressor outputs deltas of the Anchor Boxes coordinates to match the object sizes. The RoI Pooling Layer is responsible for creating fixed sized features (pooling operation of variable size) for the classifier.

Object Segmentors - Mask RCNN

Mask RCNN

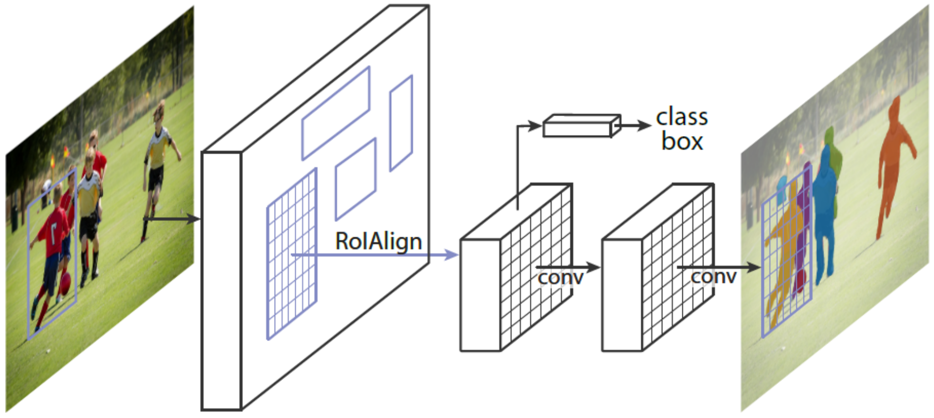
- Feature Extractor (Backbone)
- Region Proposal Network (Object Detector)
- RoI Align Layer
- Classifier with Box Regressor (Head)
- Segmentation Network (Mask RCNN Head)



Mask RCNN schematic representation.

Segmentation Network

- Consists of convolution blocks
- Output is a binary feature map (mask)



Segmentation Layers.

Mask RCNN Builds upon the Faster RCNN network and facilitates segmentation of objects by using the Segmentation Network.

04

Control Law & Trajectory Control

Control Law & Grasping Logic

Camera Transformations

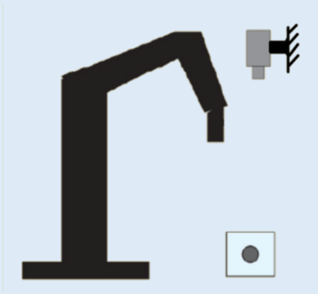
- Transformation Matrix from Base to Camera: ${}^{b}T_c$
- Transformation Matrix from Camera to Object: ${}^{c}T_o$

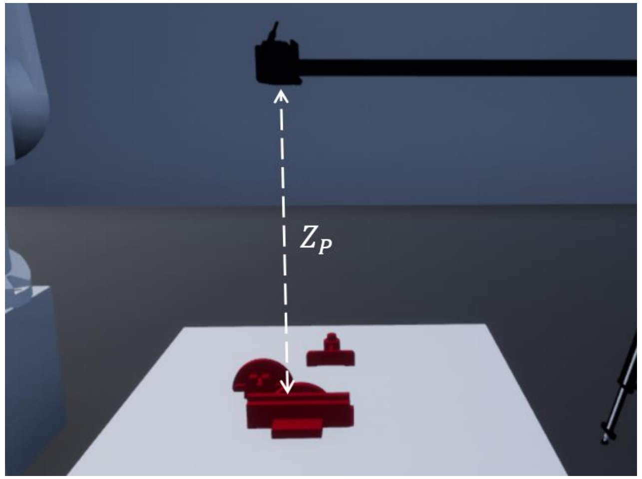
Position of the object is calculated using the following equations:

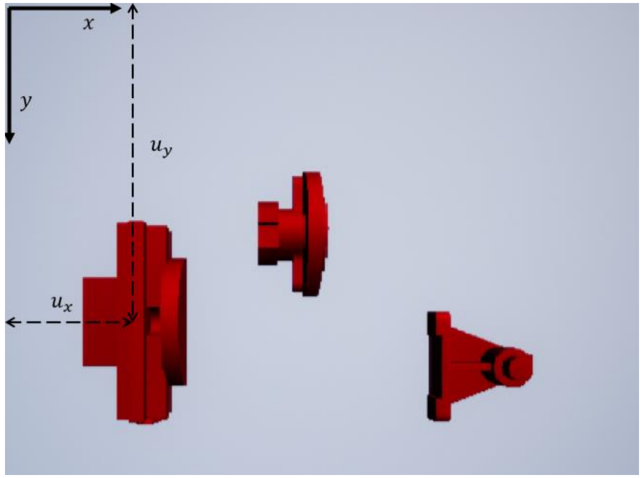
$$X_P = \frac{u_x - c_x}{f_x} \cdot Z_P$$

$$Y_P = \frac{u_y - c_y}{f_y} \cdot Z_P$$

Stationary eye & hand system







Example of calculated distances.

Grasping Logic

- Grab the highest object that is not occluded.

Grab configuration:

- Vertical to table
- Grab from the shortest axis passing through the center of area.

Inverse Kinematics

Required Configuration: ${}^{b}T_o = {}^{b}T_c \cdot {}^{c}T_o$

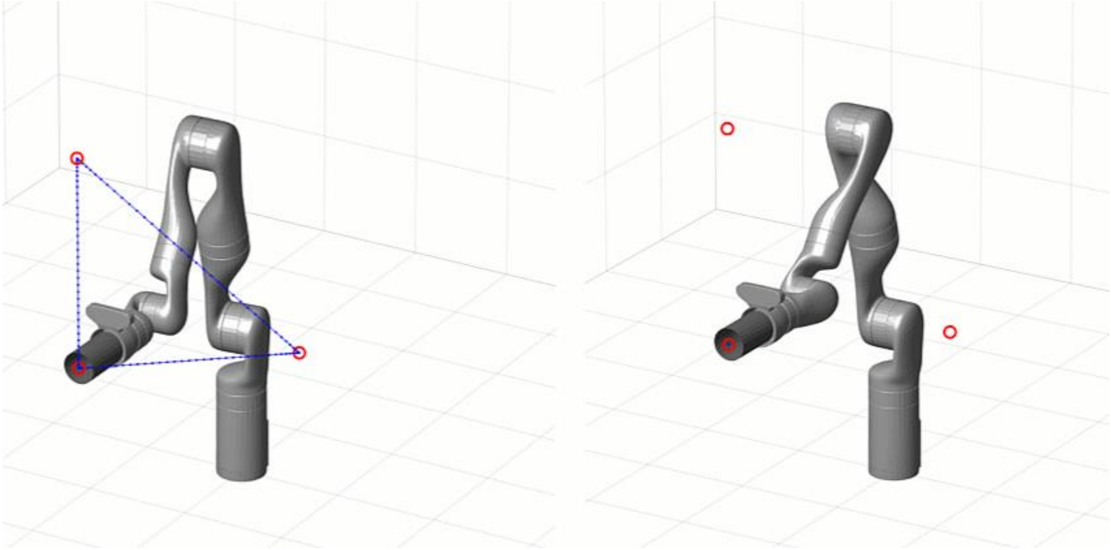
✓ Use of the inverse Kinematics solution to calculate the required joint rotations.

$${}^{b}T_o = X(q_1, q_2, \ldots, q_n)$$

Trajectory Control

Type of Control

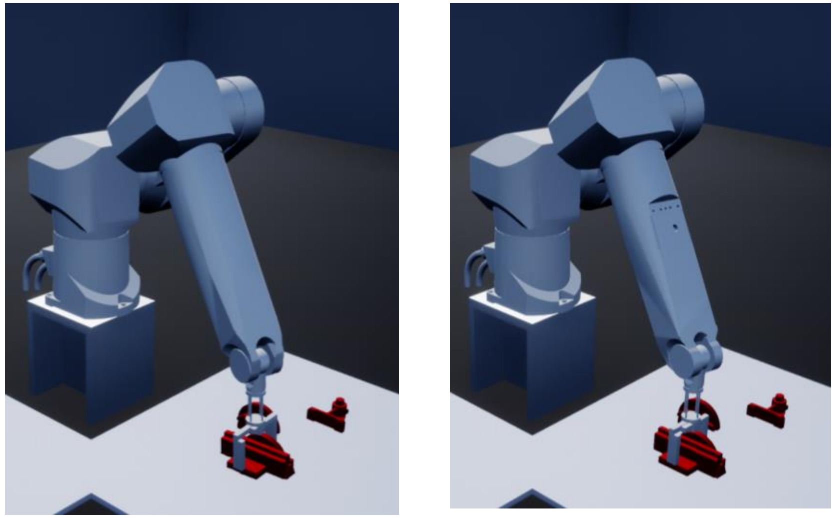
Cartesian Space	Joint Space
✓ Precise Trajectory	✓ Easy to Compute
× Computation Intensive	✓ No gimbal locks
× May lead to gimbal locks	× Not Precise Trajectory



Difference between cartesian space control and joint space control.

Pose Selection

- ➢ One end effector configuration, many possible robot poses.
- ✓ Sum of squared difference of joint rotation between initial pose and end pose.
- ✓ Use Polynomial joint trajectory.



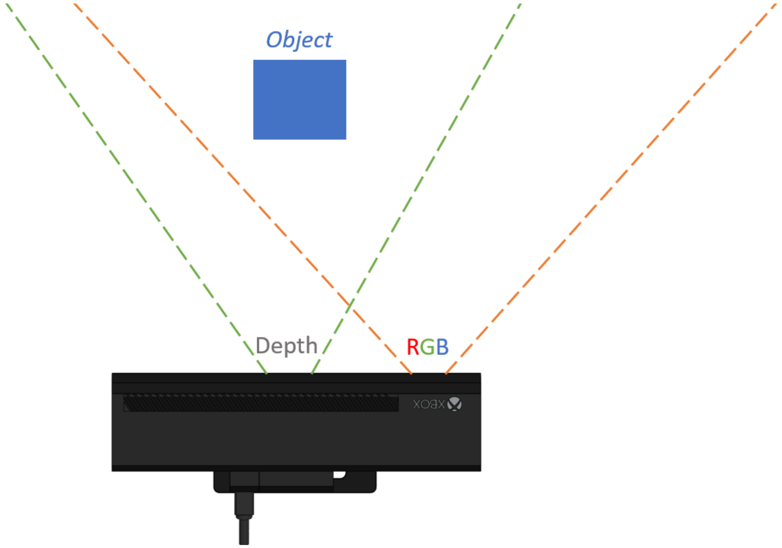
Example of same position and orientation but with different configurations.

05 Machine Vision

Kinect Calibration & Mapping

Problems

- Different resolution
- Different radial distortion
- Different Physical position

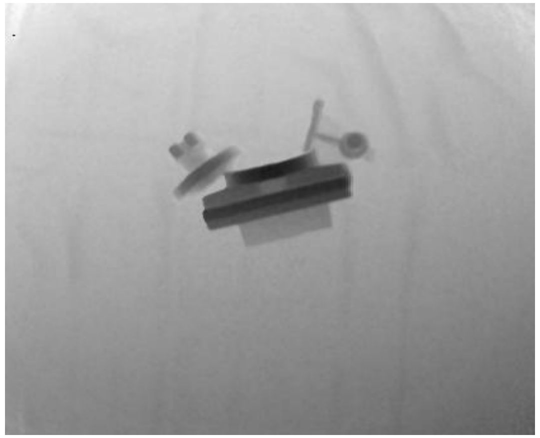


Example of sensors misalignment on the Kinect V2.

Calibration Procedure

- ✓ Use known objects for evaluation of camera properties
- ✓ Fix radial distortion.
- ✓ Calculate PAR = 3.05.
- ✓ Map images in y direction.
  $y_{mapped} = -15\ depth\ pixels$
- ✓ Map images in x direction.
  $x_{mapped}(depth)$

Mapped Images

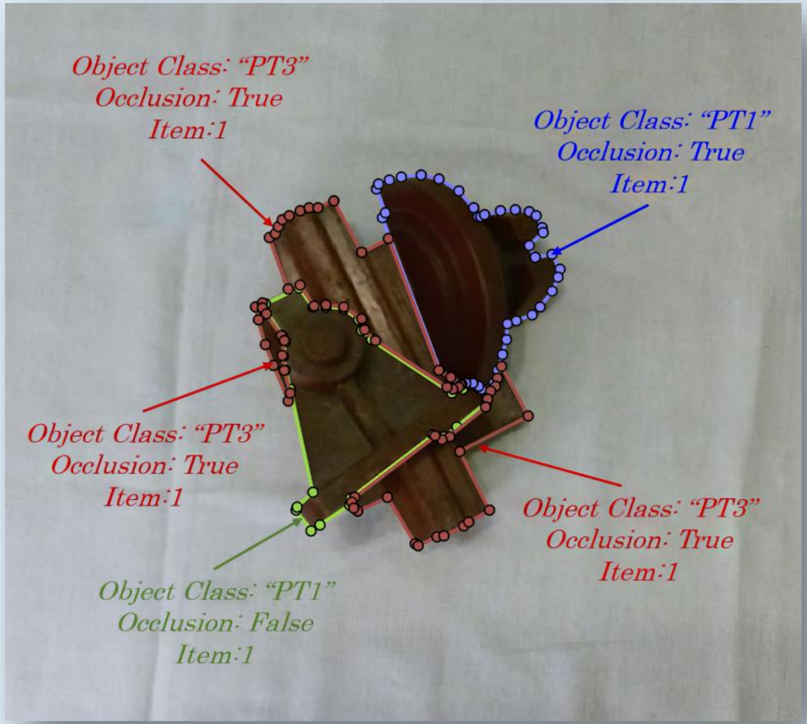


Mapped Images, Final Resolution of: $[417{,}340]_{w,h}$.

Image Labeling & Data Augmentation

Image Labeler App

- Three Item Classes
- Polygon Shaped Masks
- Occlusion Parameter: Logical
- Item Parameter: Integer



Methodology

- Pool of 10 different backgrounds
- Random rotation and translation

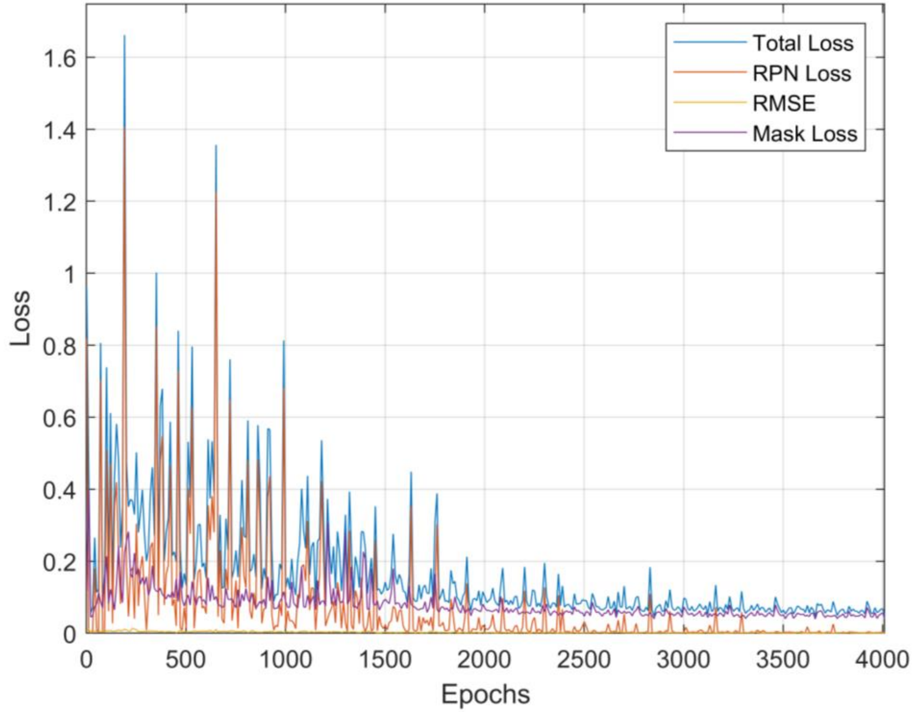
Why Data Augmentation?

- ✓ Increase the data size by 10-fold
- ✓ Generalize the model
- × Cannot be used in Occlusion ANNs

Mask RCNN Training

Pre-Trained Network

- ➢ Mask RCNN pretrained with COCO dataset.
- ➢ **COCO:** 200k labeled images, 80 object categories.
- ✓ Faster convergence of the Feature extractor & the RPN.



Training Parameters

- ➢ Augmented Images: 1140
- ➢ Real Images:114
- ➢ Training Algorithm: SGDM
- ➢ Positive IoU: [0.75,1]
- ➢ Anchor Boxes: 15
- ➢ Minibatch: 2

Loss Functions

- ➢ Box regressor (Head): Root Mean Squared Error.
- ➢ RPN: Binary Cross Entropy (cls) + Smooth L1 Loss (reg).
- ➢ Mask: Binary Cross Entropy (pixelwise).

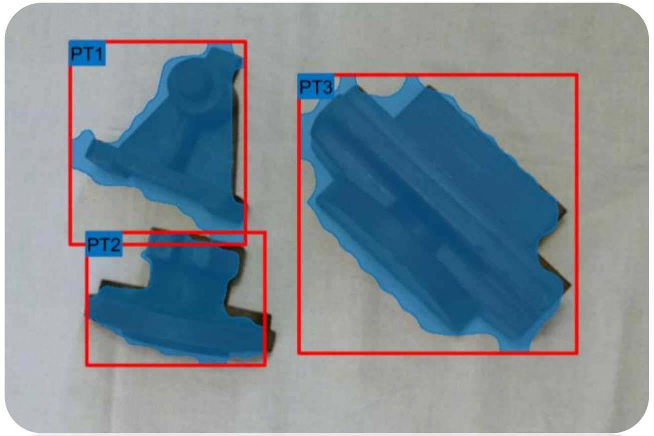
Total Loss	0.0223
RPN Loss	0.0016
RMSE	0.0004
Mask Loss	0.0153

Mask RCNN Testing

No Occlusion

- No occlusion between parts.
- 15.7% increase in total loss.

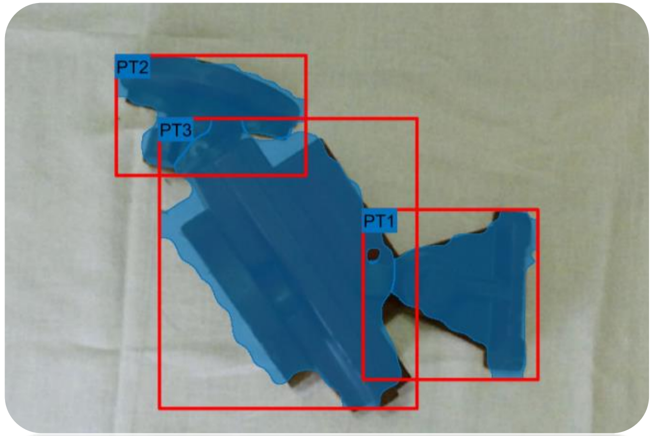
Total Loss	0.0258
RPN Loss	0.0022
RMSE	0.0005
Mask Loss	0.0179



Mild Occlusion

- Up to 25% of part area may be occluded.
- 57.8% increase in total loss.

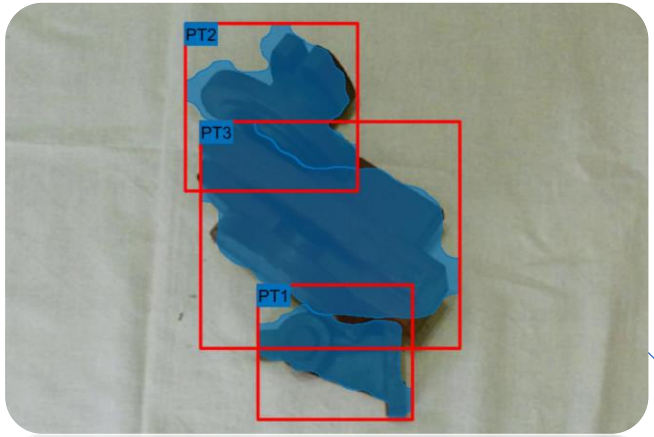
Total Loss	0.0352
RPN Loss	0.0031
RMSE	0.0005
Mask Loss	0.0251



High Occlusion

- More than 25% of part area is occluded.
- 133% increase in total loss.

Total Loss	0.0520
RPN Loss	0.0032
RMSE	0.0009
Mask Loss	0.0381



06 Occlusion ANN

Occlusion Features

Background Height

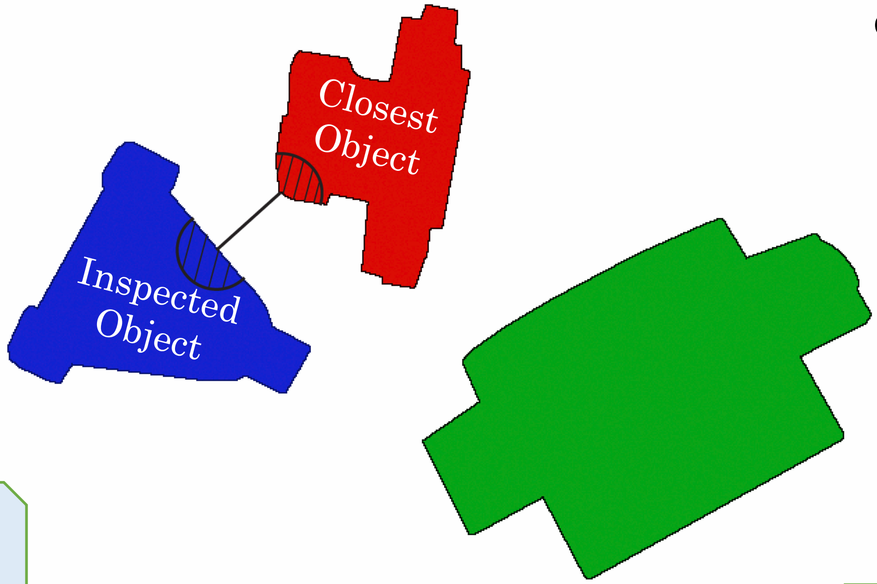
- Average height of the background.

Object Height

- Average height of the mask.

Opened Perimeter

- Mask perimeter divided by Background Height.



Height Difference Between Closest Object

- Height Difference between those two objects in a region around the closest points.

Distance Between Closest Object

- Distance in pixels between the mask of the inspected object to the mask of the closest object.

Opened Area

- Mask area divided by Background Height.

Occlusion ANNs

Parameters

- Six inputs (no normalization).
- One network for each object class.
- One "binary" output.

Training

- Training/Validation/Testing: 80/10/10%
- Algorithm: Resilient Backpropagation
- Activation Functions: "tansig"
- Evaluation criterion: MSE

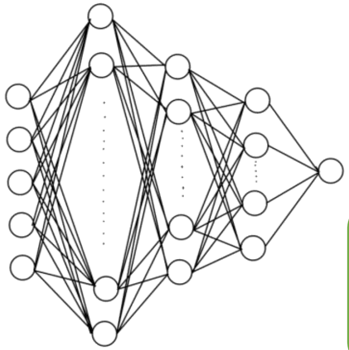


Table: Training & Testing Results.

Object Class	Hidden Layers	Training	Validation	Testing
PT1	[50,50,25,25,10,10,5,5]	2	1	1
PT2	[100,100,50,50,25,25,10,10,5,5]	0	0	2
PT3	[100,100,50,50,25,25,10,10,5,5]	4	1	1

Total Testing Performance: 90.5%

Table: Type of error of Occlusion ANNs.

Object Class	False Positive	False Negative
PT1	1	3
PT2	0	2
PT3	1	5

The networks underestimate the likelihood of occlusion. Parameters like "Closest Object Distance" and "Height Difference Between Closest Objects" give misleading inputs. Correction occurs in the grasping logic.

Occlusion ANNs are dependent on the performance of Mask RCNN and on the quality of the depth data.

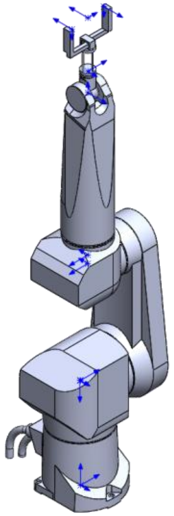
07 Virtual World Simulation

Parameters

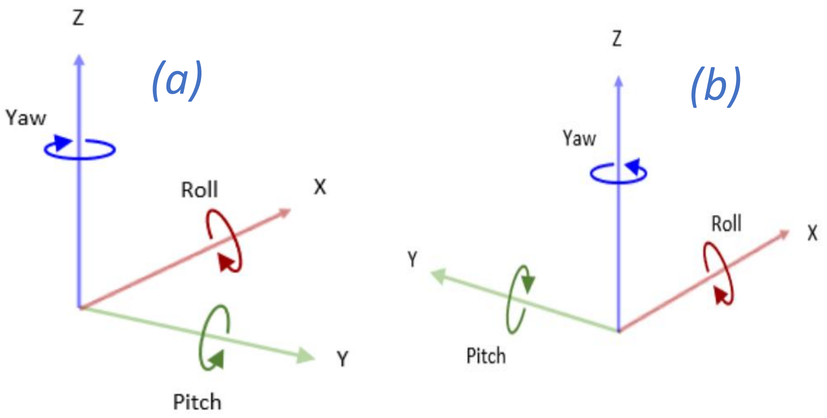
Unreal Engine

- Photorealistic
- RGB-D Cameras
- Includes Events & Collisions

Robotic Arm Model



- Assembly in SOLIDWORKS
- SW URDF Exporter add on.
- MATLAB to UE coordinate system.



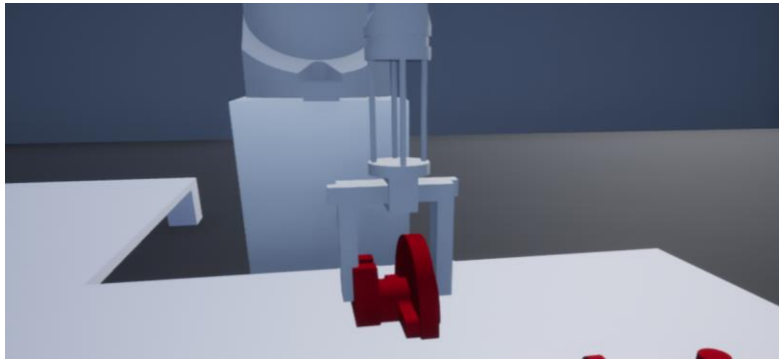
Coordinate Systems: Unreal Engine (a), MATLAB (b).

Grasping

- Lock the transformation matrix of end effector and the object:

$$^{e}T_{obj} = {}^{e}T_{b} \cdot {}^{b}T_{obj}$$

- Use collision events to visualize grasping – gripper closing.

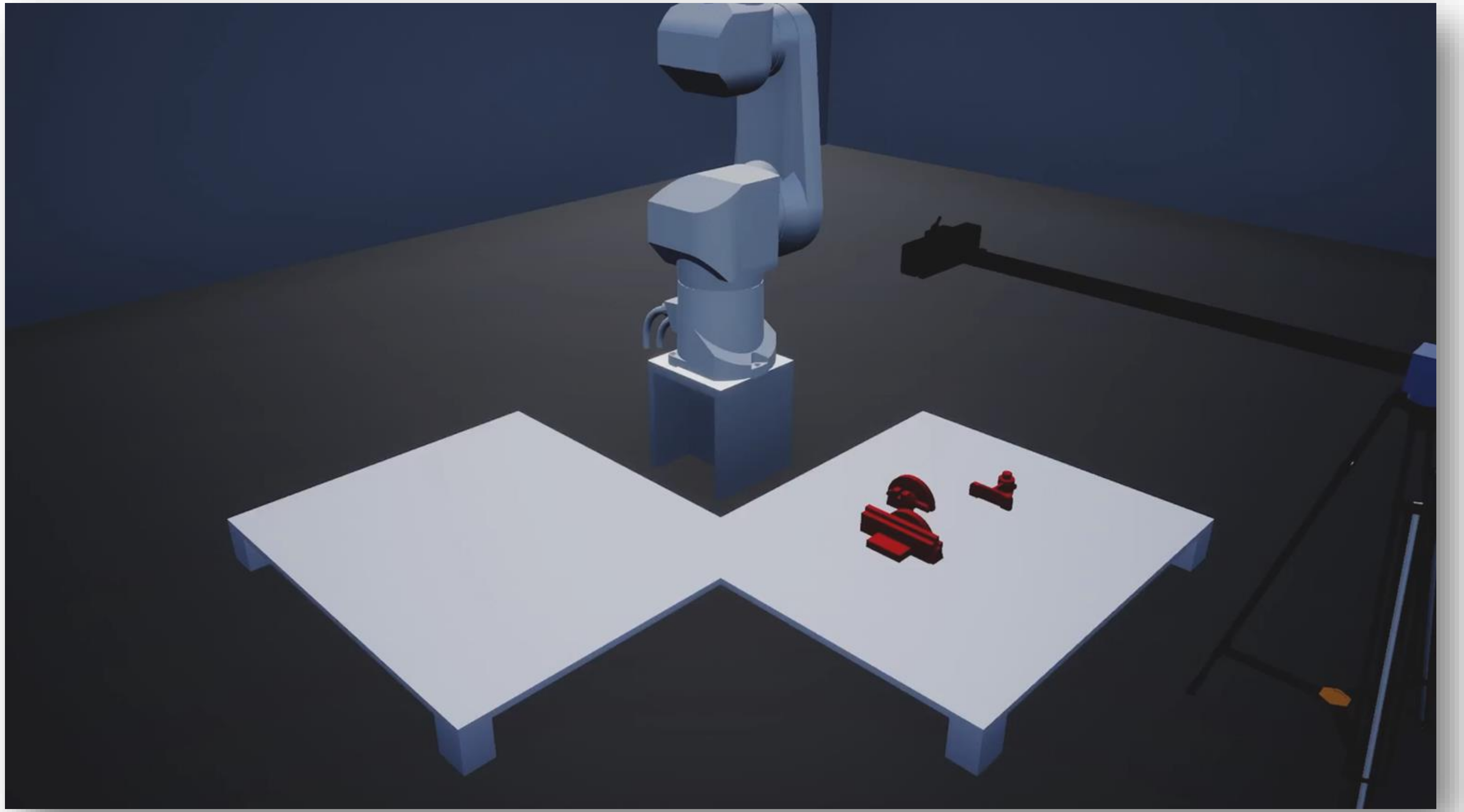


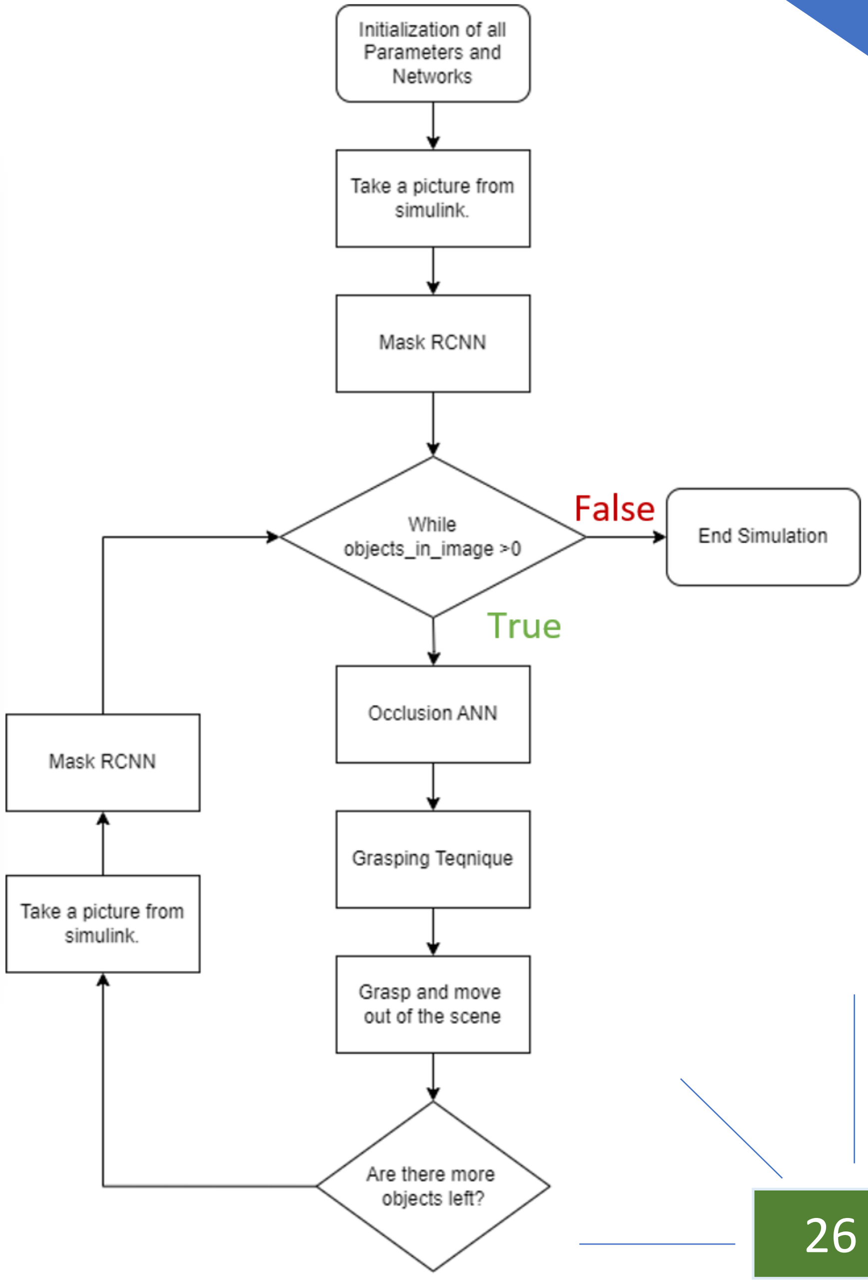
Example of grasping visualization.

MATLAB-Simulink

- Use MATLAB for the logic.
- Use Simulink – Unreal Engine for the visualization.

Simulation

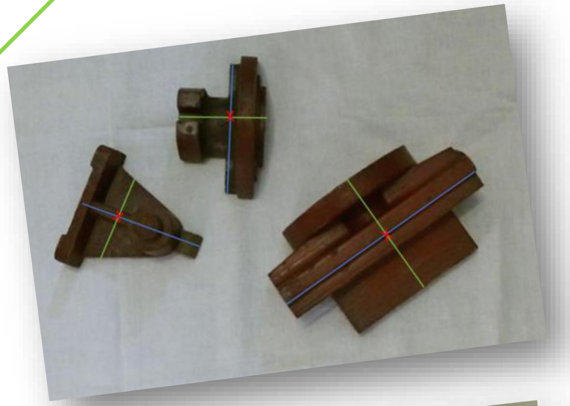


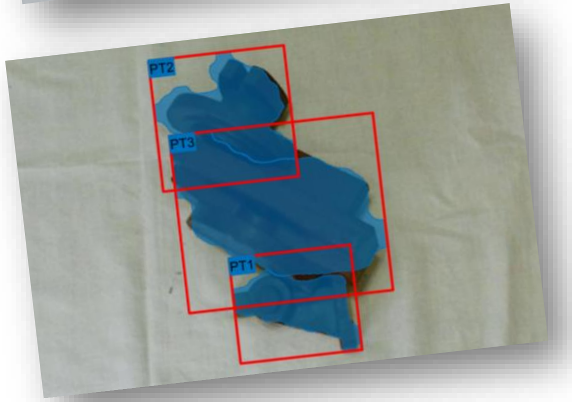


Initialization of all Parameters and Networks
Take a picture from simulink.
Mask RCNN
While objects_in_image >0
False
End Simulation
True
Occlusion ANN
Grasping Teqnique
Grasp and move out of the scene
Are there more objects left?
Take a picture from simulink.
Mask RCNN

08 Conclusions & Future Work

Conclusions





Robotics

Implementation of the Control Law involving the inverse kinematics & features from the machine vision system. Usage of joint space trajectory and an angle based grasping method.

Machine Vision

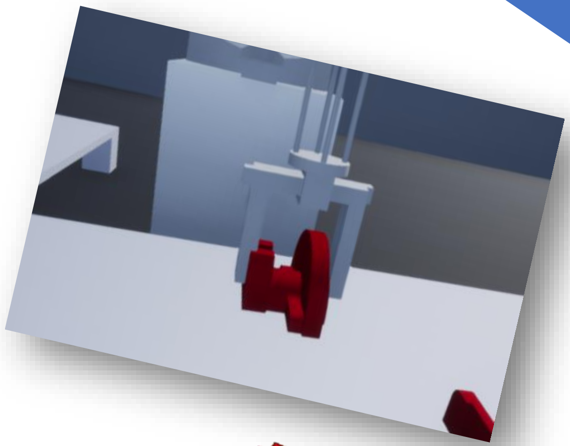
Successful employment of Mask RCNN to handle the occlusion problem with a testing loss of 0.038.

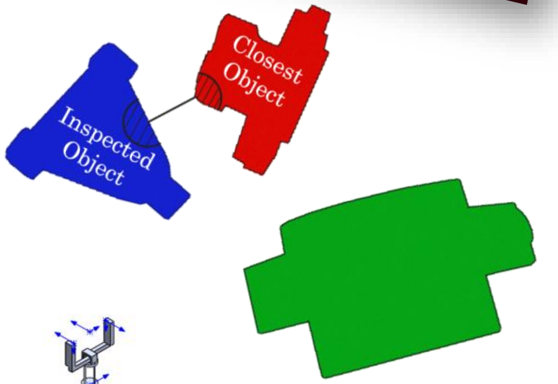
Occlusion Problem

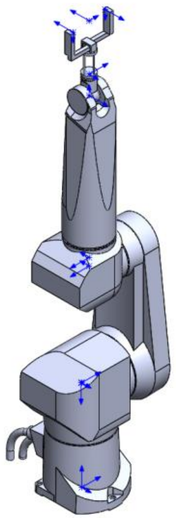
Implementation and evaluation of a Neural Networks to handle the occlusion problem. Performance in testing 90.5%.

Visualization

Full system simulation in a virtual setting inside the MATLAB - Unreal Engine Framework.

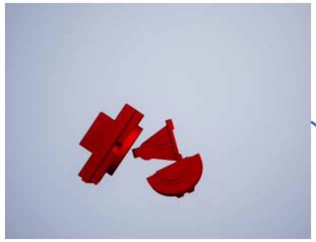


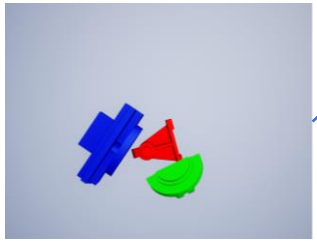




Future Work

Computer – Generated Images





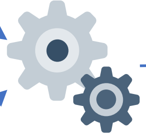
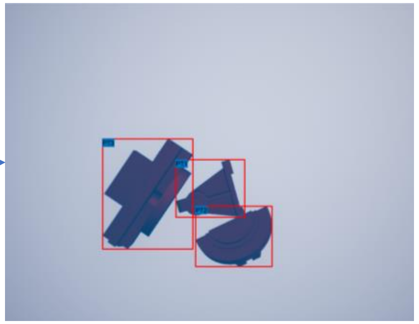


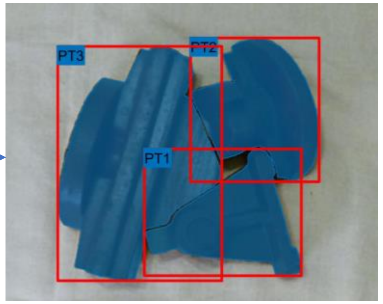
Image Processing



- ➢ Physics Engine: Blender/UE
- ✓ Increase productivity

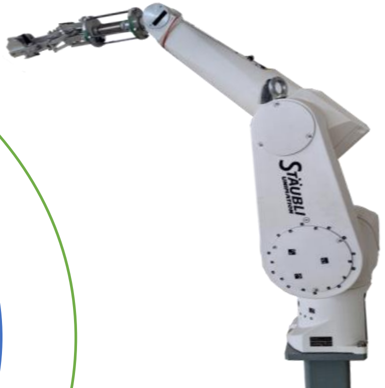
Mask RCNN with RGB-D images

Mask RCNN



- ➢ Python - MATLAB
- ✓ Increase Performance

Real-World Implementation



- ➢ Serial Communication
- ➢ Control with V+ Language
- ➢ System Calibration
- ➢ Simulation

Thank you for your time!