# Object Detection and Motion Planning

Shreyas Chandra Sekhar
*Department of Robotics Engineering*
*Worcester Polytechnic Institute*

***Abstract*—Surgical environment as seen by from Master console of Surgical Robots are way different from that of the outside world. It is quite complex and challenging to get samples of real time surgical environments due to privacy restrictions. To over come this, Domain Randomization techniques was used to detect objects in simulated environment. Many variations of object shapes, sizes, textures were used for training. Multiple distraction like change in camera angle, variation of background were introduced in sampling to simulate real time distractions. Experiments were conducted in increasing order of complexity of the environment and number of objects. Randomizing the environment helps transferring the learning from simulated to actual environment with high accuracy. On successful prediction of desired objects using our prediction model, Inverse Kinematic was used to pick and place the object of interest from its current to target location. Various trusted models like YOLO, Fast-RCNN, Faster-RCNN, MobileNetV1 and MobobileNetV2 were explored for this research. MobileNetV2 was simple and also memory efficient for this particular use case. These experiments were conducted in simulated ROS environment to speed up data collection, labeling and to simulate multiple environments. ROS further provided flexibility of extending experiment as needed using actions and services. NVIDA TX2 was used as hardware, which has PASCAL GPU with 8GB RAM. This was sufficient to handle both, ROS simulation and prediction model. This research was intent to be implemented in da Vinci environment, however due to high memory foot print of the simulator, this experiments were conducted in much smaller environment with similar features. The vision of this research is to apply the same technique of object identification and path planning to da Vinci Surgical robot.**

## I. Introduction

Surgeries are always unpleasant necessity to anyone undergoing it. Recovery from surgeries are normally very painful, tedious and can take around four weeks to get back to work. It may further takes several months to get back to normal life depending up severity of the surgery. This majorly depends upon the size and region of the surgical wound [1] [2]. An average time length with an abdominal incision is about one to two months or even just six weeks. It's the deeper tissues called the fascia needs to heal. The skin heals fairly quickly, but the deeper tissues take a little bit longer [3]. Those tissues are basically what make up your abdominal wall.

The impact of surgery can be reduced by with size of surgery wound, reducing blood loss and time of surgery. With the increase of awareness and use of Surgical robots, studies brings in many interesting facts that surgical robots are much precise and precise. They require much narrow opening for inserting surgical robots which can be just just few centimeters [4] [5] [6]. Surgical robots gives better access and view on the subject.

This paper provides the first step on object identification in simulated environment that can be transferred to real time environment with confidence. Domain Randomization is active filed of Computer Vision research with many proven approaches [7]. A different environment than the da Vinci, surgical simulation (AMBF) was used for this research. This was done to first focus on proving the concept of Domain Randomization in a well known environment [7] and later apply the same concept to AMBF [8]. AMBF is da Vinci simulated environment created at AIMs lab at WPI.

ROS environment was used to simulate objects of various shapes, sizes textures. Using a simulated environment makes the sampling, training and testing easy than the real time environment. These environment can further be extended for making the robot self learning by applying deep reinforcement learning algorithms. These techniques has been used by many Atari [9] and robotics control [10], [11]. Reinforcement learning explores entire User Space of the robot which can make it dangerous with real time environment and simulations makes it safe, fast and cheep. It further needs millions of attempts and trial to get a learning model. Simulations makes it simple and viable approach. There are always physics challenges to bring all real time limitations in simulation. Even with strong system identification, the real world has unmodified physical effects like non-rigidity, gear backlash, wear-and-tear, and fluid dynamics that are not captured by current physics simulators. These differences, known collectively as the reality gap. To address this challenge, Domain Randomization samples data from various background during training period. Deep learning frameworks are normally supervised. One main challenge is to label huge number of samples which would be used by the deep learning framework. Generating huge number of labeled data are complex in real time environment but are manageable in simulated environment.

There are many well proven State of object identification deep learning frameworks that uses pixel values of the images [12], [13], [14], [15], [16]. [17], was one of the promising algorithm for this application which was used in this paper. Different types of objects, textures and backgrounds were considered in the samples during training of the network. Inverse kinematics was used to pick and place the object from one location to another in the simulated environment.

Previous experiments and research in this filed normally

relies on high camera precision, multiple sensors and sensor fusions. These techniques are effective, however expensive due to the requirement of high precision hardware. On the out hand our model relies more on bringing variations in sampling. A variety of background images, distractions, object textures bringing in all the uncertainty as in reality. It further relies on the prediction model than hardware. Though this takes longer in training, predictions at the edge is much faster and cheaper as the model is trained and adapted for all real time variations and not too sensitive to hardware.

## II. Object Detection Challenge

### A. Challenge Overview

The motivation of this research was to detect objects in real time as in [18] kit. However due to the memory foot print and complexity, this research has been sub-modeled into object detection and motion planning in a similar, less complex environment. This research would be extended to implement the same concept in da Vinci simulation [8].

There has been many well proven object identification algorithms [12], [13], [14], [15], [16], which uses mesh model and 3D metric feature representation [19]. Color co-occurrence diagram is used to map the 3D object test time [20]. These traditional method uses depth of the camera image for classification. This relies mainly on the quality of the picture and need expensive camera and sensors. Our approach relies on Deep learning frame work and the the variation of sample data for identification and accuracy. As the accuracy approaches satisfactory results, this methodology will be applied to [8].

## III. Literature review

The task of image classification has been well addressed in many papers. Forward Kinematics (FK) and Inverse Kinematics (IK) has been well researched and a proven concept. This paper focus on image identification in simulated environment and make the robot arm pick and place object. Strength of this research is to use Domain Randomization [21] by taking various samples with changing background and apply it on surgical robot. This research was originally targeted to perform similar classification with object pick and place in [8] da Vinci surgical robot simulation. However, due to high memory footprint of the simulation, training need a bigger hardware. So this was conducted in a similar ROS robot with lessor memory foot print.

Endoscopic vision of da Vinci surgical robot are quite different that of outside world. Open data sets such as MNIST provides has very good samples for general image classifications, however these do not addresses the needs in surgical Robotics World. This leads to a need of collecting relevant data set as per this application. As these images in 3D, many of the latest Computer Vision techniques were considered as reference. 3D structure from single images by capitalizing on the power of Deep Convolutions Neural Networks [22]. It provides Generative Adversarial Networks (GANs) and DCNNs in order to reconstruct the texture and shape from single images. This lead us consider similar approach for object identification in da Vinci Research Kit [18]. At inference time, [23] method uses motion parallax cues from the static areas of the scenes to guide the depth prediction. The depth prediction was a concept guided us in approaching this research. [24] address the problem of learning to suggest the single best frame across the video for user annotation. This address the classification of moving clip and can be used further in the enhancement of this paper.

## IV. Impact Statement

The proposed work is innovative because it captures the various possible ways to simulate the objects as in surgical robot environment. It considers objects of different shapes and textures. It also uses background randomization which brings in lots of uncertainties as in real environment. This research also composes of performing useful task of pick and place objects using object detection. This simulated environment is very close to that of da Vinci simulated environment [8]. So, this learning can easily be transferred.

Our method focus on sampling objects with various posses and background. We collect image samples of various textures, color, size and shapes that would closely resembles the objects in da Vnci simulated environment.

## V. Framework Details

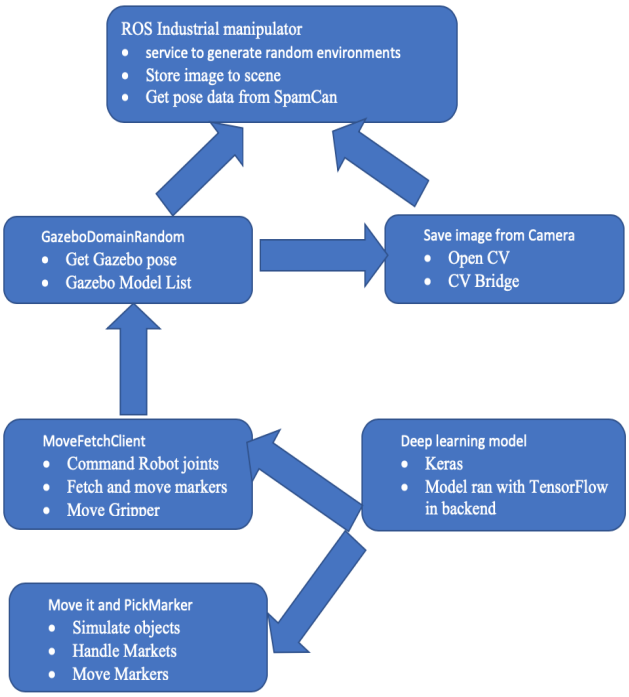



Figure 1: Architecture diagram

This architecture of this research implementation has several subsystems. Research was conducted in Ubuntu 16.04, NVIDA Jetson TX2 which has 8GB RAM, 6 PASCAL CPU. It used 2TB external SATA SSD Hard drive connected though SATA external cable. ROS kinetic was used with openCV for object collection. It used Keras as training API

with Tensorflow as its backend. TensorBoard was used to monitor training trends.

A 3D SpamCan environment was created with a table to place to objects and trash bin as target. ROS Industrial manipulator [25]package was used for this. It provides robust and flexible manipulation of industrial robot with high performance. GazeboDomainRandom plugin was used to randomize the environment with various background and camera view. It further generate virtual datasets for an object recognition task, as it will automatically generate the bounding boxes for the object we seek to recognize in every generated pictures. Wrapper class was used to access the gazebo systems and retrieve the current pose of the objects in the scene. XMLGenerator and XMLObjectTags packages were used for this purpose. RGBCamera was used to get the 3D pose of the objects of interest for training. CV Bridge package was used, which converts between ROS Image messages and OpenCV images. Many samples were generated with varying Object color, environment lighting, camera orientations object shapes and sizes. MoveFetch-Client rapper was used to find the locations of the object for sampling as well for pick and place. This gave a flexibility not to always set the Fetch robot in a non-intrusive pose, but also create training materials with random poses of the robot arm, adding extra occlusions to the training.

MoveFetch class was created reads the current pose of the object using the response of the geometric service. It reads it from corresponding res message. We send as service request a geometry pose, a joints array and a movement type. These values will be interpreted to generate one movement or another, based basicaly on the movement type. The service response is just a boolean to indicate if the service was successful.

RViz Markers [26], [27] were used simulate all the objects. It proves C++ API wrapper for displaying shapes and meshes in Rviz. It also provide helpful methods for debugging and manipulation. The markers are made to follow the robot end effector pose once they are picked up and made to follow free fall into the bin on reaching the destination.

MobileNetV2 model was imported using Keras APIs. Its input layer was modified as the size of the input image. Tensorflow was used as its backend. We create a Convolutional layer to add to the end of the original model, so that the output of the predictions are two values, which are the X and Y coordinates of the Spam object. We start the training with all the inputs from a launch file.

- Single Object varying only X axis
- Single Object varying X and Y axis
- Two objects Random poses fixed background
- Two objects Random poses changing background

Here are the sequence of steps that happens with every validation:

- Reset the Environment: We call the plugin and reset the environment with, in this case, a new position of the Spam. Then, we wait for the objects in the scene to settle.
- Get the Camera Image: We save a screenshot of the settled scene.
- Make prediction: Based on the image, predict where the Spam object is.
- Get the real Spam and Table Pose.
- Publish Markers: With all this info, we publish markers to visualize the predicted Spam pose, the real Spam pose, and the Table pose.
- Move Fetch Arm: If activated, the Fetch arm will use the predicted position of the Spam to try and execute a grasping sequence.

### *A. Environment Simulation*

This research was conducted in simulated environment using ROS. It provides easy approach in varying the environment setup and in creating sample data and object formats. It also provides easily accessible services to modify object orientation and camera positions as required. Simulation allows this to be done really fast and as random as possible. In reality, it wouldn't be feasible to have thousands of variations in each object of the scene, moving lighting rings, or a random object positioning system. The environment was build in 3D space with relevant services to modify object orientations, camera movements and handle robot joints. Gazebo plugins was used to manipulate the environment settings and appearance.

### *B. Object sampling*

Variety of object sampling and collection plays a key role in making the the prediction robots and generalized. Its not just the number of samples but also the variation of objects and the environment back ground plays a key role [9]. Among all the issues in prediction, the one that's most challenging is the fact that depending on the lighting conditions. The environment can change so much that your model won't be able to predict correctly, even if nothing else has changed. We feed the Training model with loads of images of a randomly changing environment. This will abstract the environment, independent of the lighting, textures, position of the objects, camera, lights, etc.

### *C. Object Collection*

To start, the environment was sampled with single object and by fixing one of the coordinates. Background randomize service was created to randomize the background. The image samples were stored with the given pose. A gazebo plugin was created to randomize the environment. OpenCV was used to target the object of interest with its coordinates.

Randomization includes the below:

- Change the object's position and orientation randomly in the scene, always inside the table area.
- Change the object's color.
- Change the lights of the scene and their position orientation.
- Change the Main camera's position orientation

- Move the fetch robot as we sample. This will create random poses of the robot arm, adding extra occlusions to the training.

Here are some sample random environments and objects used in this research.

XML labeling for this image was created, with the data of the Pose3D, image size, image file name. A script was created to access gazebo systems and retrieving the current pose of the objects in the scene. Gazebo Domain Randomization rapper was used to generate XML file for each image which has relevant details about the image size [21].

### *D. Model Description*

Multiple object detection model was considered for this research. We used MobileNetV2 [17] for this research due to its adaptability to this use case. Input shape was modified to 96x96x3 to fit into the model. The Model was also tried with 128 and 224 input sizes with and without dropouts. The Weights were trained from scratch for the initial experiment and they were retrained as the the number of objects was increased. Most of the layers ReLu activation. In practice, we found that using random weight initialization works as well in most cases.

Adam optimizer was used with momentum and diminishing learning rate.

### *E. Training model*

Keras APIs with TensorFlow backend were used in training the model.

30, 60, 90 epochs were considered for model training. 120,000 images samples were created. Two csv files for training and validation, respectively, were generated. Sample images were split to 80 percent of the images go to train, and 20 percent to validation. Adam optimizer proved to be most optimal and converging.

The same model was sampled with the objects much closer to the Fetch Robot so that it was in its arm range. This was done to retrain the simulation to find the Spam anywhere on the table.

There were many training ran with increasing environment complexity. Here are the series of different experiments conducted:

- Single object, changing just X coordinate, no changing background.
- Single object, changing both X, Y coordinate, no changing background.
- Two object, changing both X, Y coordinate, no changing background.
- Two object, changing both X, Y coordinate, with changing background and lighting.

The environment was progressively added with additional objects to bring variation and distractions to the object collection. This was done to retrain the simulation to find the Spam anywhere on the table and with one detractor.

The environment was randomized to make it more robust in any lighting condition. There were huge number of image samples collected (120,000) with varying background and distraction. Different object samples were used to make the simulation as close to real time environment and to introduce randomization.

The collected images are labeled with XML file, and the images are reshaped to fit the training architecture.

Here are samples of training data with two objects in random poses and background.

### *F. Results*

There were several experiments conducted with varying number of epochs, batch sizes optimizes and momentum values.

Varying batch sizes shows

Some of the experiments results did not have high impact on the turning parameters. Tuning Batch size and Momentum had noticeable change in accuracy and losses. The experiments about was conducted with Adam optimizer, initial learning rate was set to 0.0001 with beta1=0.9, beta2=0.999. Training was ran for 30 epochs each with varying number of batch sizes and Momentum. The same experiments was ran with different types of sampling images.

- Single Object varying only X axis
- Single Object varying X and Y axis
- Two objects Random poses fixed background
- Two objects Random poses changing background

Training had 1,20,000 image samples and split 80 to 20 percent between train and validation.

All experiment were conducted with Mean Squared Error loss, Adam optimizer and image size of 96x93x3.

### *G. Results with varying batch size*

20 different experiments were conducted with varying batch size of 5, 10, 15, 20 and 25 on 4 different environments.

It was observed that batch size of 5 gave best accuracy and least loss.

Batch size of 5 with with single object varying just x-axis provided close to 100% accuracy and zero loss after 30 epochs. Batch size of 5 gave least case accuracy of 97% in Two objects with random background and Orientation environment. Accuracy gradually decreased in complex environments for a fixed batch size.

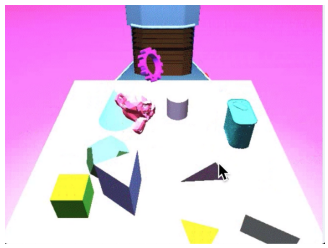
For a fixed environment, it was also observed that accuracy gradually decreased with increasing the batch size. Accuracy of Two objects with random background and Orientation environment showed 97%, loss of 0.1 for batch of 5 and an accuracy of 92%, loss of 1.2 with batch sized of 25.

### *H. Results with varying Momentum*

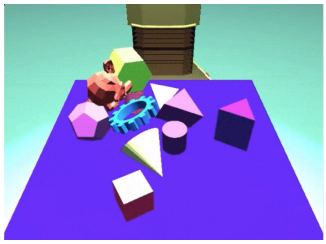
20 different experiments were conducted with varying Momentum of 0.6, 0.7, 0.8, 0.9 and 1 on 4 different environments.

It was observed that momentum of 0.9 gave best accuracy and least loss.

(a) Random Environment 1



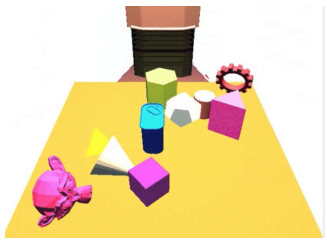
(b) Random Environment 2



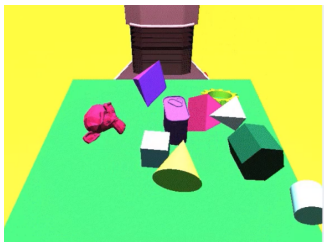
(c) Random Environment 3

(d) Random Environment 4

(e) Random Environment 5

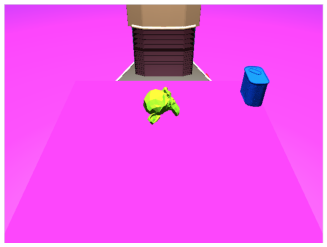


(f) Random Environment 6

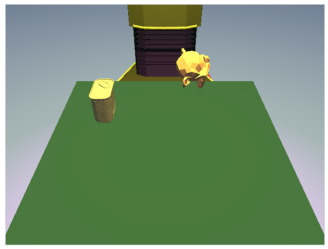


(g) Random Environment 7

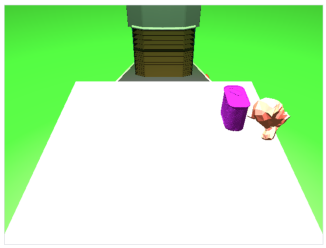
(h) Random Environment 8



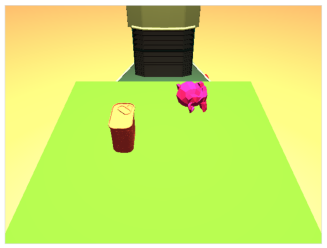
(a) Training Data 1



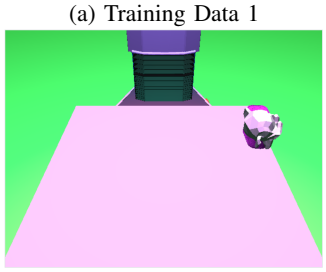
(b) Training Data 2



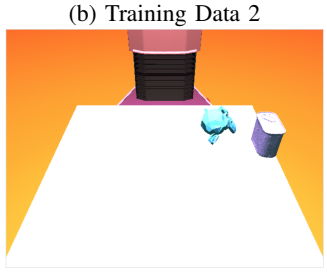
(c) Training Data 3



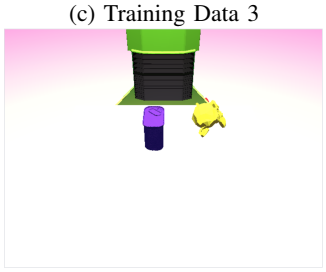
(d) Training Data 4



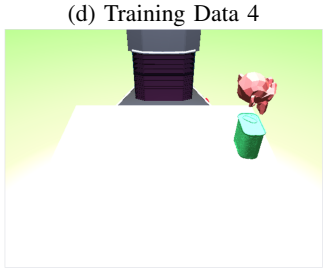
(e) Training Data 5



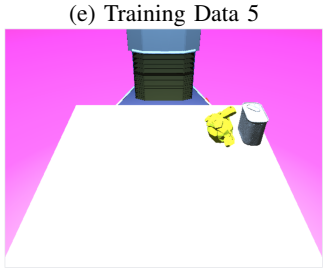
(f) Training Data 6



(g) Training Data 7



(h) Training Data 8

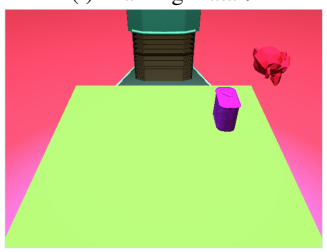


(i) Training Data 9

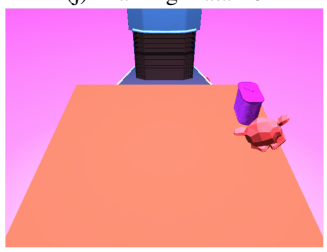
(j) Training Data 10

(k) Training Data 11

(l) Training Data 12



(m) Training Data 13



(n) Training Data 14

(o) Training Data 15

(p) Training Data 16

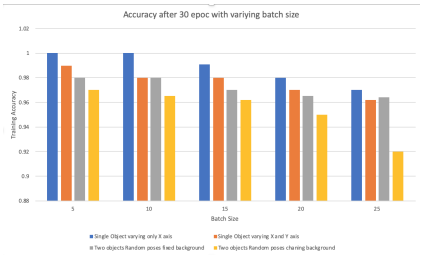



Figure 4: Accuracy for 30 epochs with varying batch sizes

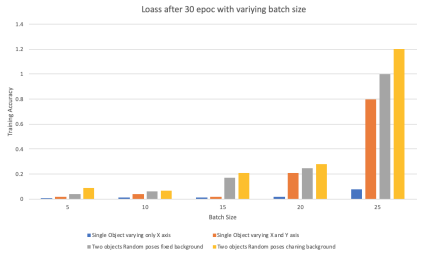



Figure 5: Loss for 30 epochs with varying batch sizes

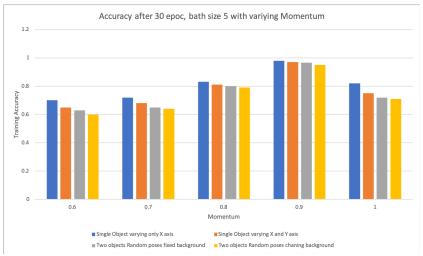



Figure 6: Accuracy after 30 epochs with varying Momentum

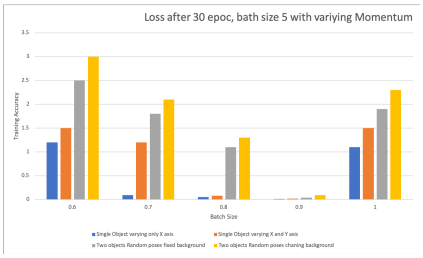



Figure 7: Loss after 30 epochs with varying Momentum

Momentum of 0.9 with with single object varying just x-axis provided close to 100% accuracy and zero loss after 30 epochs. Momentum of 0.9 gave least accuracy of 97% in Two objects with random background and Orientation environment. Accuracy gradually decreased in complex environments for a fixed Momentum.

For a fixed environment, it was also observed that accuracy gradually increased with increasing the Momentum until 0.9 and dropped beyond that.

Accuracy of Two objects with random background and Orientation environment showed 97% and loss of 0.1 with momentum of 0.9 and an accuracy of 92% and loss of 1.2 with momentum of 0.6.

### *I. Results with varying Image size*

Multiple training were ran on each environment with varying image sizes. All images were 3D X*X*3 with X being the image size in each axis and 3 being RGB color channels. Image sizes of 96, 128, 160, 192, 224 were trained on all the 4 different environments with varying complexities.

All other training parameters were maintained unchanged. Batch size of 5, momentum of 0.9, Adam optimizer was trained for 30 epochs.

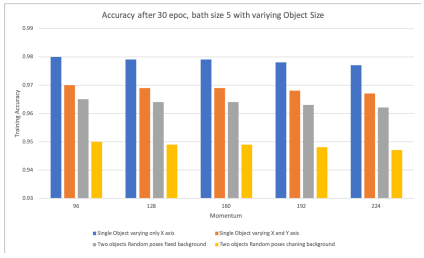



Figure 8: Accuracy after 30 epochs with varying Image size

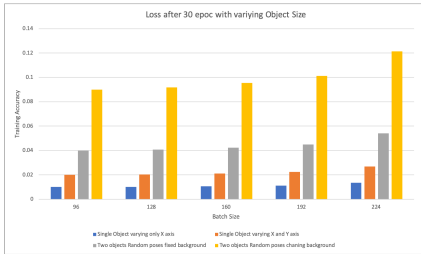



Figure 9: Loss after 30 epochs with varying Image size

It was observed that images size 96x96x3 showed best accuracy and least losses. Prediction accuracy was still under acceptable limits with biggest object size of 224x224x3. Accuracy ranges from 98% with single objects in 96 frame to 97.5% in 224 frame. Losses were close to 1% to 1.5% respectively.

Complicated environments with Two objects Random poses chaining background, showed an accuracy of 95% in 96 object frame compared to 94% in 224 object frame. Losses were 2% and 3% respectively.

Momentum of 0.9 with with single object varying just x-axis provided close to 100% accuracy and zero loss after 30 epochs. Momentum of 0.9 gave least accuracy of 97% in Two objects with random background and Orientation environment. Accuracy gradually decreased in complex environments for a fixed Momentum.

For a fixed environment, it was also observed that accuracy gradually increased with increasing the Momentum until 0.9 and dropped beyond that.

Accuracy of Two objects with random background and Orientation environment showed 97% and loss of 0.1 with momentum of 0.9 and an accuracy of 92% and loss of 1.2 with momentum of 0.6.

### J. Accuracy of Object Identification vs Pick and Place

There were two main goals of this research:

- Object detection in various environments
- pick and place of objects in these environments

Trained model after 30 epochs with batch size of 5, momentum of 0.9 using Adam optimizer were validated for 100 times in each environment. The accuracy of object identification, pick-place were studied.

It was observed that, Object detection was 100% accurate with single object moving just on x-axis. It had an accuracy of 97% of 100 trials in complected environment with 2 object with random background and orientation.

Pick and place could successful pick and place the objects from target to end location with an accuracy of 95% out of 100 trials. It gave an accuracy of 85% in complicated environment. It was observed that, Forward and Inverse Kinematics were successfully computed, however the robot arm smashed into the objects due to its inertia. This needs further turning and adjustment of Robot dynamics.

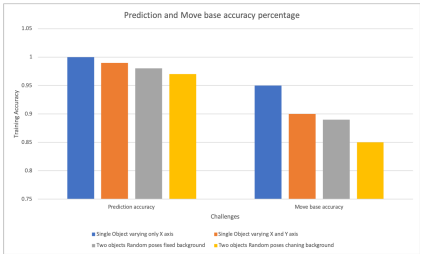



Figure 10: Accuracy of Object Identification vs Pick and place

### K. Prediction accuracy of different objects in various environments

There were different types of objects used for validation like Cone, Cube, Cylinder, Hexagonal Prism, Pyramid Rectangular Prism.

Trained model after 30 epochs with batch size of 5, momentum of 0.9 using Adam optimizer were validated for 100 times in each environment one each object.

It was observed that, Cone and Cube gave the best accuracy across all environments ranging from 99% in single object environment to 96% accuracy in environment with Two objects Random poses changing background.

All objects gave 99% accuracy in Single object environment with just changing x-axis. Variations of Pyramids showed lower accuracy relatively to other objects in all environments. Rectangular Prism gave an accuracy of 98% in Single Object varying only X axis environment and 94% in environment with Two objects Random poses changing background.

### L. Training Two objects Random poses changing background

Training environment with Two objects Random poses changing background showed very interesting running accuracy and losses. Training was ran for 30 epochs with 0.9

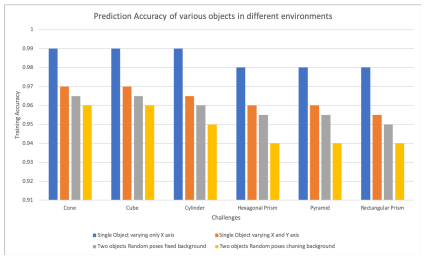



Figure 11: Prediction accuracy of various objects in different environments

momentum, batch size of 5 and a starting learning rate of 0.001 and gradually decreased it to 0. 1,20,000 data sets were used. Sample images were split to 80 percent of the images go to train, and 20 percent to validation.

Accuracy gradually kept increasing with epochs and plateaued between 8 to 14 epochs at 80% accuracy and increased close to 97 percent 22 epoch. Loss was always decreasing over epochs.

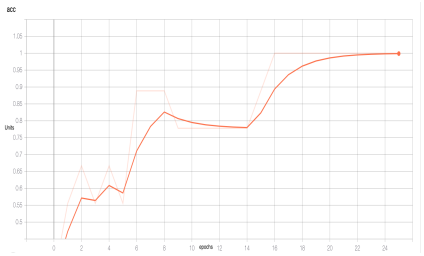



Figure 12: Accuracy of Two objects with random background and Orientation

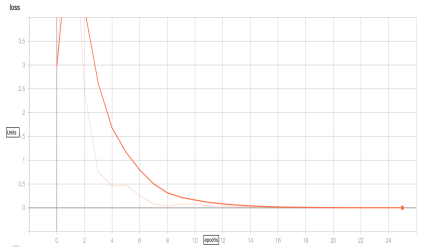



Figure 13: Loss of Two objects with random background and Orientation

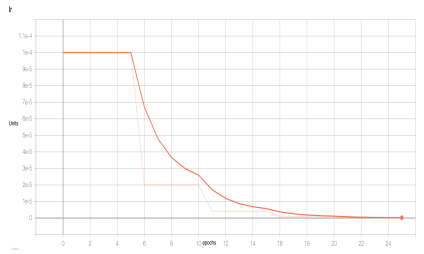



Figure 14: Learning Rate of Two objects with random background and Orientation

Validation loss kept decreasing with epochs which indicates no over-fitting with batch size of 5.

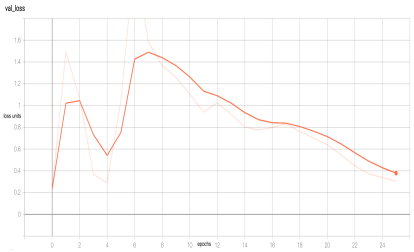



Figure 15: Learning Rate of Two objects with random background and Orientation

## VI. Current Research and Future Work

Current Research included experimenting domain randomization with multiple object shapes, background color and orientation. This research had two parts. First was to object identification in the simulated by using MobilNetV2 [17]. Second part of the research was to command the robot arm to pick and place the identified object in the seen. With the research and experiments explained, the objects in the seen could be identified with high accuracy. Inverse Kinematics was used to successfully pick and place the objects. Various types of objects and background were used for validation and verification.

There were few occasions at which the robot end effector displaced the target object though it was properly identified. This would need further optimization of Pick and Place algorithm.

As seen from the samples, object identification and pick-place by the robot can be successfully performed. Future work would be to apply this learning and approach in da Vinci medical robot simulation [8]

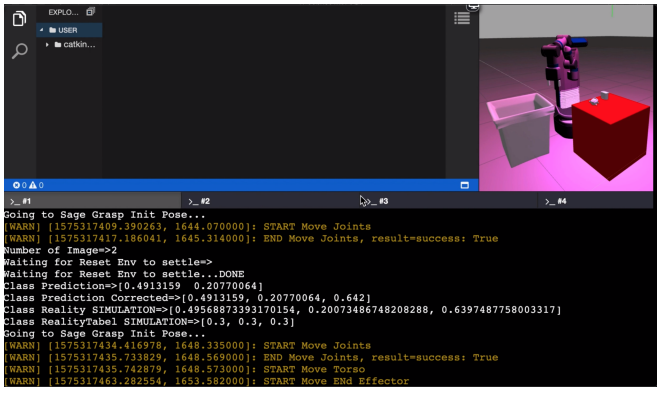
## VII. Conclusion

Research was successfully conducted which included various types of objects and domain randomization was adapted to reduced the gap between simulation and real environment. Object recognition was ran on various environments with increasing complexity. Accuracy of more than 95% could be achieved in a random environment with multiple objects. Accuracy of more than 85% could be achieved in object pick and place. As a next step, this methodology would be applied to medical robot environment [8].

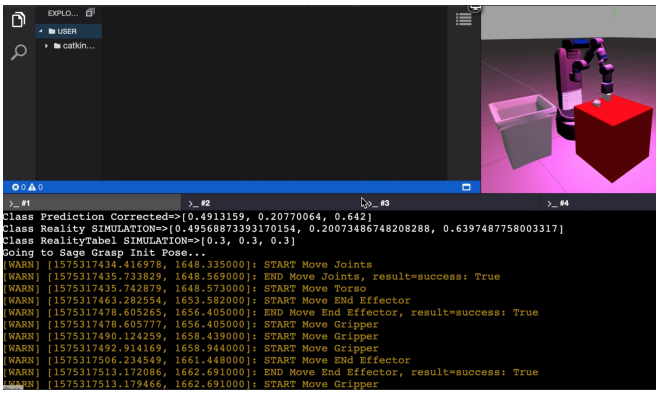
## VIII. Appendix

Here are End to end cycle in 4 different environments.

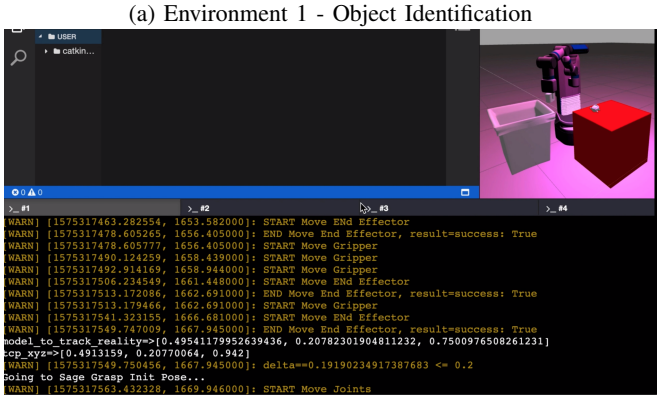
- Reset the Environment
- Make prediction: Based on the image, predict where the Spam object is.
- Pick up object from source
- Compute Inverse Kinematics to find the join trajectory.
- Move Fetch Arm to destination
- Drop object at destination



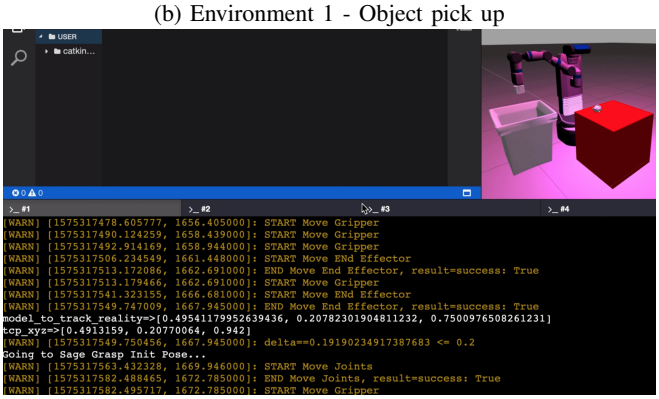

(a) Environment 1 - Object Identification



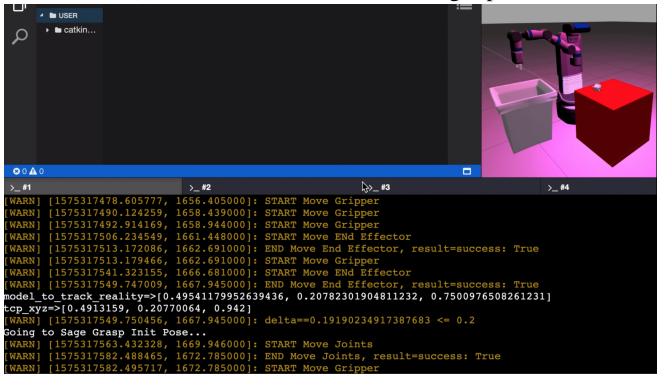

(b) Environment 1 - Object pick up



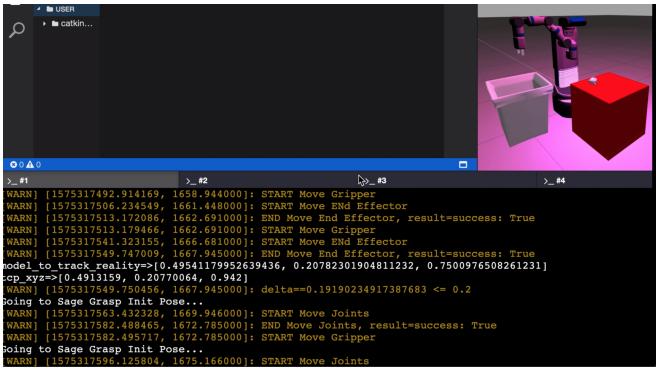

(c) Environment 1 - Move to target pose



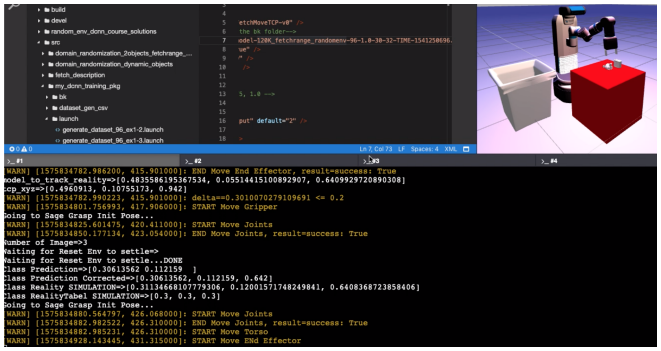

(d) Environment 1 - Reached target pose



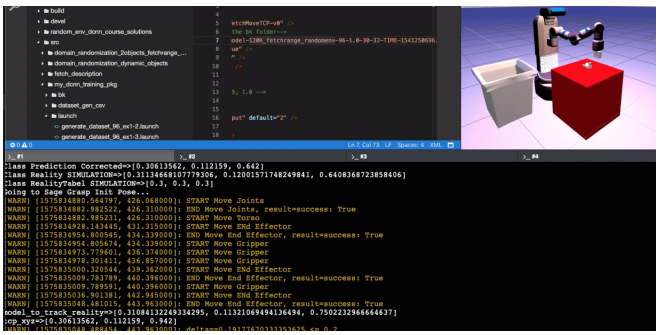

(e) Environment 1 - Object dropped



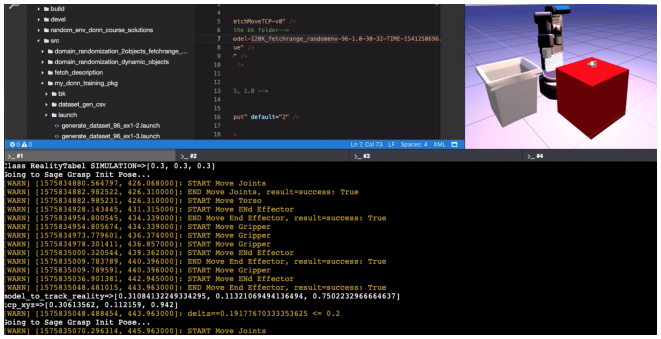

(f) Environment 1 - Ready for next cycle



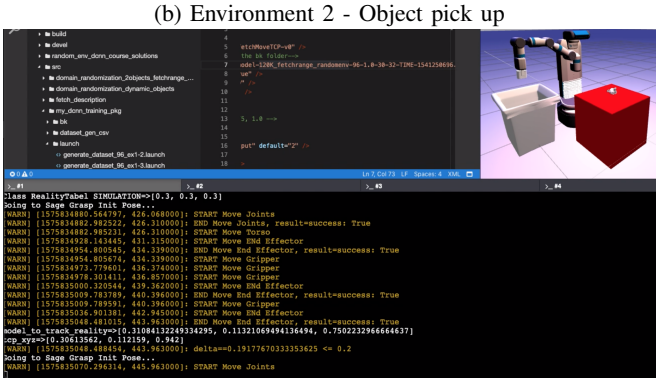
(a) Environment 2 - Object Identification



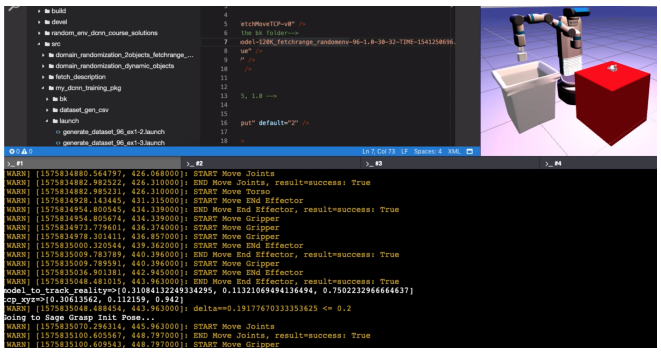
(b) Environment 2 - Object pick up



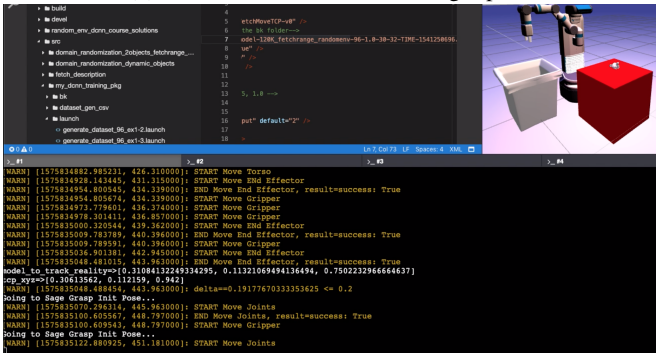
(c) Environment 2 - Move to target pose



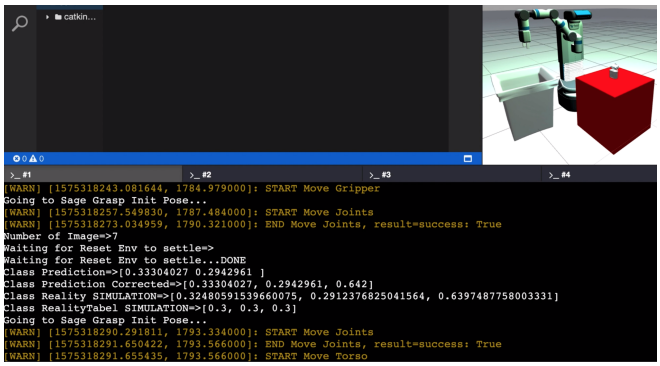
(d) Environment 2 - Reached target pose



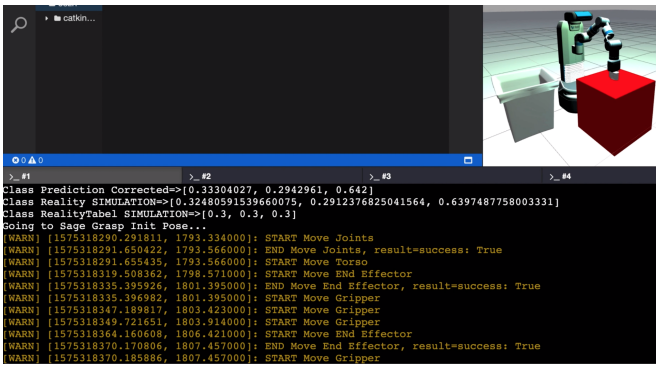
(e) Environment 2 - Object dropped



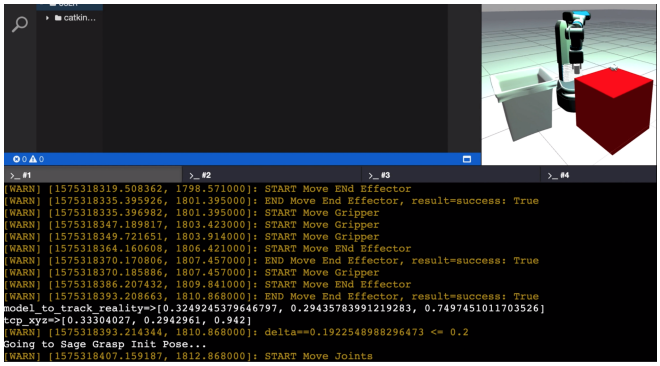
(f) Environment 2 - Ready for next cycle



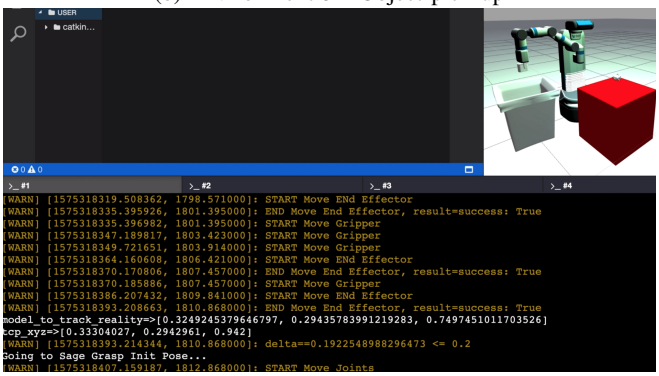

(a) Environment 3 - Object Identification



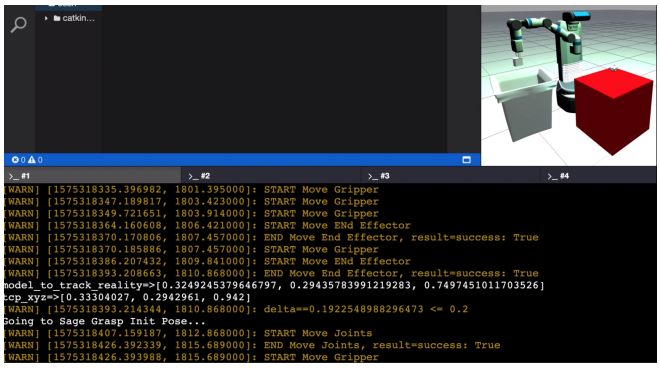

(b) Environment 3 - Object pick up



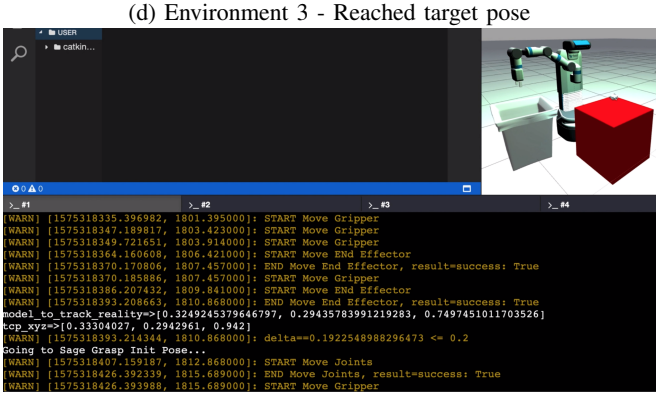

(c) Environment 3 - Move to target pose



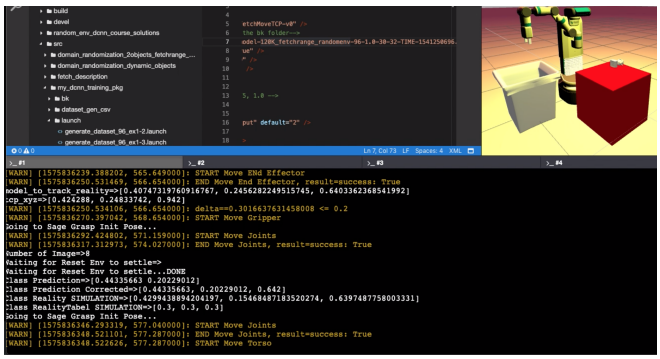

(d) Environment 3 - Reached target pose



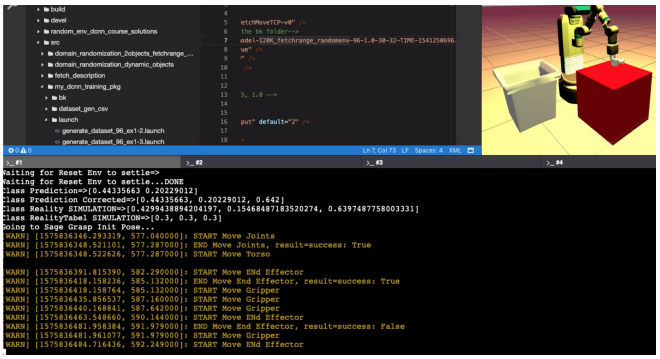

(e) Environment 3 - Object dropped



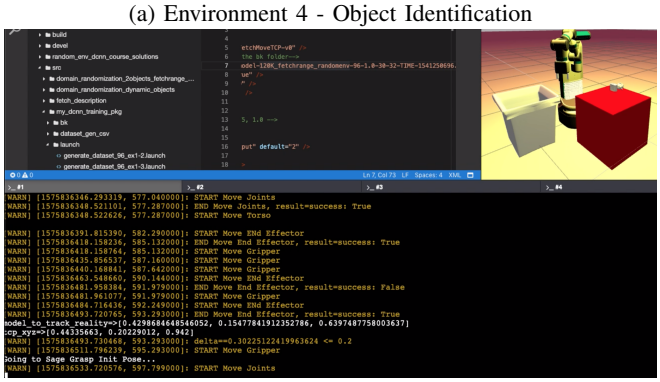

(f) Environment 3 - Ready for next cycle



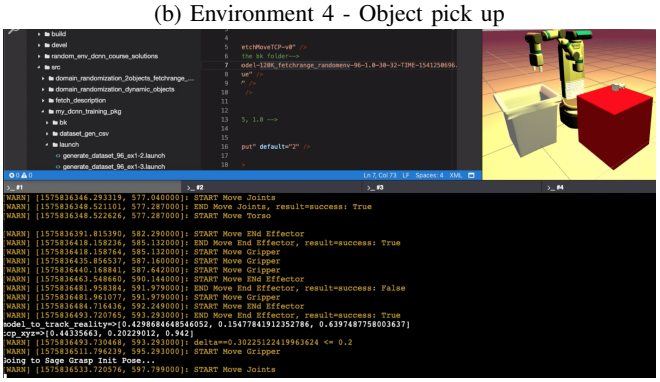

(a) Environment 4 - Object Identification



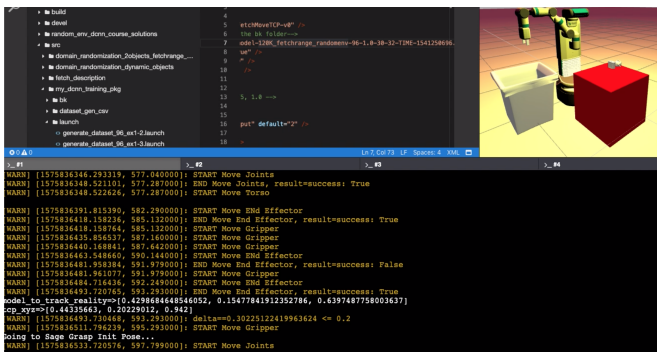

(b) Environment 4 - Object pick up



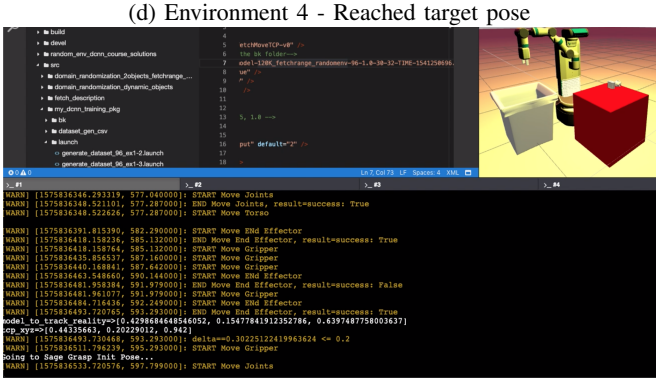

(c) Environment 4 - Move to target pose




(d) Environment 4 - Reached target pose




(e) Environment 4 - Object dropped




(f) Environment 4 - Ready for next cycle

## REFERENCES


[1] D. J. Campsen. Getting life back to normal after surgery. youtube. [Online]. Available: https://healthcare.utah.edu/the-scope/shows.php?shows=0_jcd5tg17

[2] M. F. Jennifer Whitlock, RN. Timing of recovering after surgery. youtube. [Online]. Available: https://www.verywellhealth.com/recovering-after-surgery-3156847

[3] M. Clinic. Timing of recovering after surgery. youtube. [Online]. Available: https://www.mayoclinic.org/tests-procedures/robotic-surgery/about/pac-20394974

[4] J. H. U. S. o. M. Department of Surgery, "The surgical recovery index," April 2004. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/15026915

[5] O. J, "Minimally invasive abdominal surgery," *Minimally Invasive Abdominal Surgery*, July 2000. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3117521/

[6] S. Endosc, "Sri," April 2004. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/15026915

[7] A. R. J. S. W. Z. P. A. osh Tobin, Rachel Fong, "Domain randomization for transferring deep neural networks from simulation to the real world," https://arxiv.org/abs/1703.06907, 2017.

[8] W. Y. G. R. . G. F. Munawar, Adnan, "A real-time dynamic simulator and an associated front-end representation format for simulating complex robots and environments," IROS, IEEE, 2019, Dec, 2019.

[9] D. S. A. A. R. J. V. M. G. B. A. G. M. R. A. K. F. G. O. e. a. Volodymyr Mnih, Koray Kavukcuoglu, "Human-level control through deep reinforcement learning," Nature, 518(7540):529–533,, 2015.

[10] T. D. VSergey Levine, Chelsea Finn and P. Abbeel, "Endto-end training of deep visuomotor policies," Journal of Machine Learning Research, 17(39):1–40,, 2016.

[11] P. A. M. I. J. John Schulman, Sergey Levine and P. Moritz, "Trust region policy optimization," ICML, pages 1889–1897, 2015.

[12] R. G. A. F. Joseph Redmon, Santosh Divvala, "You only look once: Unified, real-time object detection," https://arxiv.org/abs/1506.02640, May, 2016.

[13] A. F. Joseph Redmon, "Yolo9000: Better, faster, stronger," https://arxiv.org/abs/1506.02640, Dec, 2016.

[14] T. D. J. M. Ross Girshick, Jeff Donahue, "Rich feature hierarchies for accurate object detection and semantic segmentation," https://arxiv.org/abs/1311.2524, 2014.

[15] R. Girshick, "Fast r-cnn," https://arxiv.org/abs/1504.08083, 2015.

[16] R. G. J. S. Shaoqing Ren, Kaiming He, "Faster r-cnn: Towards real-time object detection with region proposal networks," https://arxiv.org/abs/1506.01497, 2016.

[17] M. Z. A. Z. L.-C. C. Mark Sandler, Andrew Howard, "Mobilenetv2: Inverted residuals and linear bottlenecks," https://arxiv.org/abs/1801.04381, Jan, 2018.

[18] A. D. G. F. R. T. P. Kazanzides, Z. Chen and S. DiMaio, "An open-source research kit for the da vinci surgical system," in robotics and automation," *ICRA*, June 2014. [Online]. Available: http://zihanchen.com/assets/papers/ICRA14PKAZ.pdf

[19] M. M. Alvaro Collet and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," The International Journal of Robotics Research, 30(10):1284–1306, 2011.

[20] D. K. Staffan Ekvall and F. Hoffmann, "Object recognition and pose estimation using color cooccurrence histograms and geometric modeling." Image and Vision Computing, 23(11):943–955, 2005.

[21] A. R. J. S. W. Z. P. A. Josh Tobin, Rachel Fong, "Gazebodomainrandom," https://github.com/Near32/GazeboDomainRandom, Dec, 2016.

[22] I. K. S. Z. Baris Gecer, Stylianos Ploumpis, "Generative adversarial network fitting for high fidelity 3d face reconstruction," *Advanced Robotics*, November 2019. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Gecer_GANFIT_Generative_Adversarial_Network_Fitting_for_High_Fidelity_3D_Face_CVPR_2019_paper.html

[23] F. C. R. T. N. S. C. L. W. T. F. Zhengqi Li, Tali Dekel, "Learning the depths of moving people by watching frozen people," *CVPR*, April 2019. [Online]. Available: https://arxiv.org/abs/1904.11111

[24] J. J. C. Brent A. Griffin, "Learning to select the guidance frame in video object segmentation by deep sorting frames," *CVPR*, March 2019. [Online]. Available: https://arxiv.org/abs/1904.11111

[25] ROS. manipulator. youtube. [Online]. Available: https://robots.ros.org/category/manipulator/

[26] ——. Marker. youtube. [Online]. Available: http://wiki.ros.org/rviz/DisplayTypes/Marker

[27] D. Coleman. Rviz visual tools. youtube. [Online]. Available: https://github.com/PickNikRobotics/rviz_visual_tools