3D OBJECT LOCALIZATION USING FAST R-CNN BASED OBJECT RECOGNITION AND RGB-D SENSORS

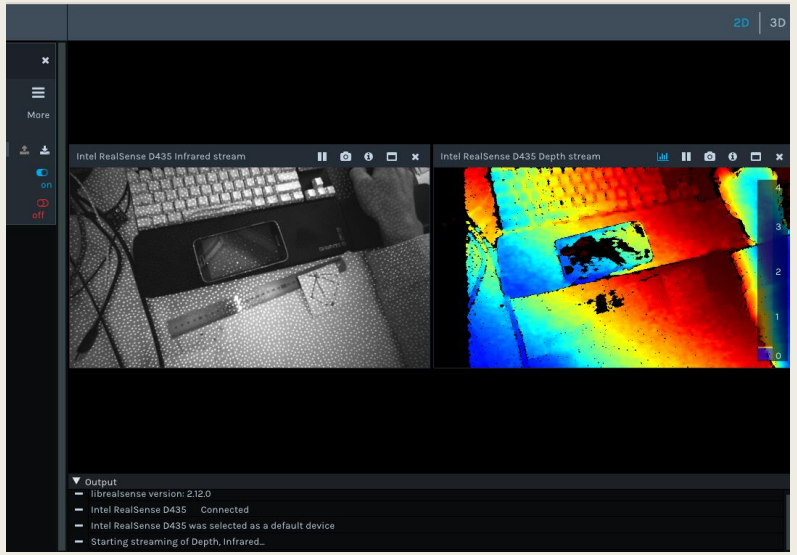
Evan Lou

Introduction – 3D Object Localization

- 3D object localization is the ability to locate object in 3D space for machines and computer programs.
- In field of robotics and autonomous driving, 3D object localization is widely implemented to achieve object avoidance, reconstruction of the environment, and navigation.
- Industry of augmented reality (AR) has also adopted 3D object localization to improve the quality of users' experiences when operating devices.

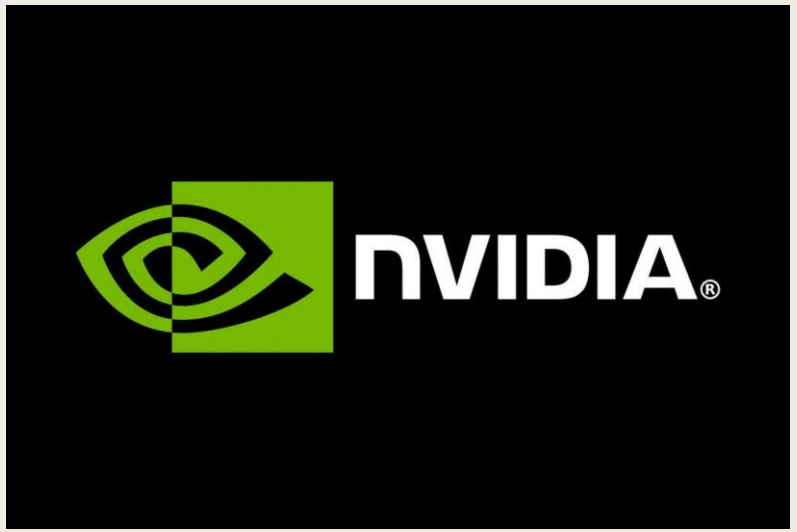
Question/ Goal

- This project attempted to achieve real-time, three-dimensional object localization through the implementation of RGB-D sensors and object recognition techniques. The computer program is built using python and C++, and it relied on OpenCV to construct a user interface.



Installing Sensors

- For both economic and compatibility reasons, I chose the Intel RealSense D400 series RGB-D cameras as my depth sensor in this project.
- RGB-D cameras, while providing me the depth data of the environment, also maintain a RGB image that I can reference back to, which is an advantage for RGB-D cameras over the more popular LiDAR sensors
- By installing the official Intel RealSense camera viewer, I am able to connect the sensor to my computer and extract raw frames from it.



Object Recognition

- To perform real time object detection, I utilize Faster R-CNN object detection architecture, models from TensorFlow, and support from Nvidia's compute unified device architecture (CUDA) toolkit.
- I import the Faster R-CNN model from the TensorFlow library pretrained using the COCO training dataset. I load the model into the memory and perform inferences on the frames captured by the camera using the TensorFlow session.
- Finally, to design an interface that allows users to directly interact with the program, the program import from OpenCV library is used to extract the results from the inferences and display them on the screen.

Different Object Recognition Architectures

R-CNN

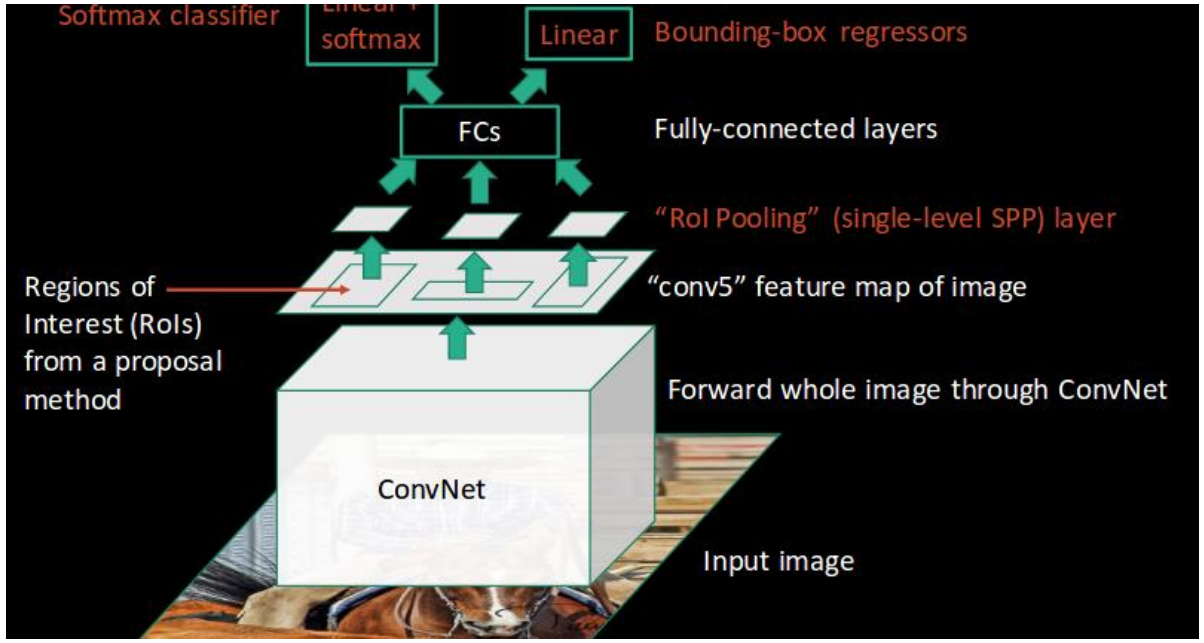
- R-CNN select about 2,000 regions of interest of an image and feed them directly into a CNN to compute the result. It is significantly faster than a normal CNN, however, the architecture is too slower to be implemented in real-time.

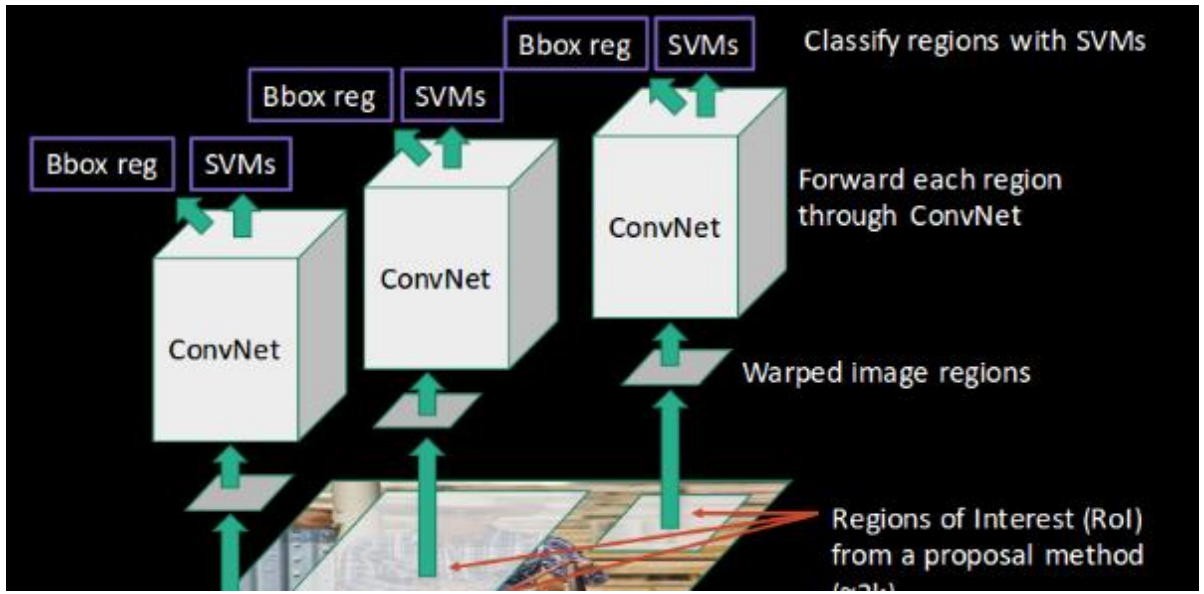
Fast R-CNN

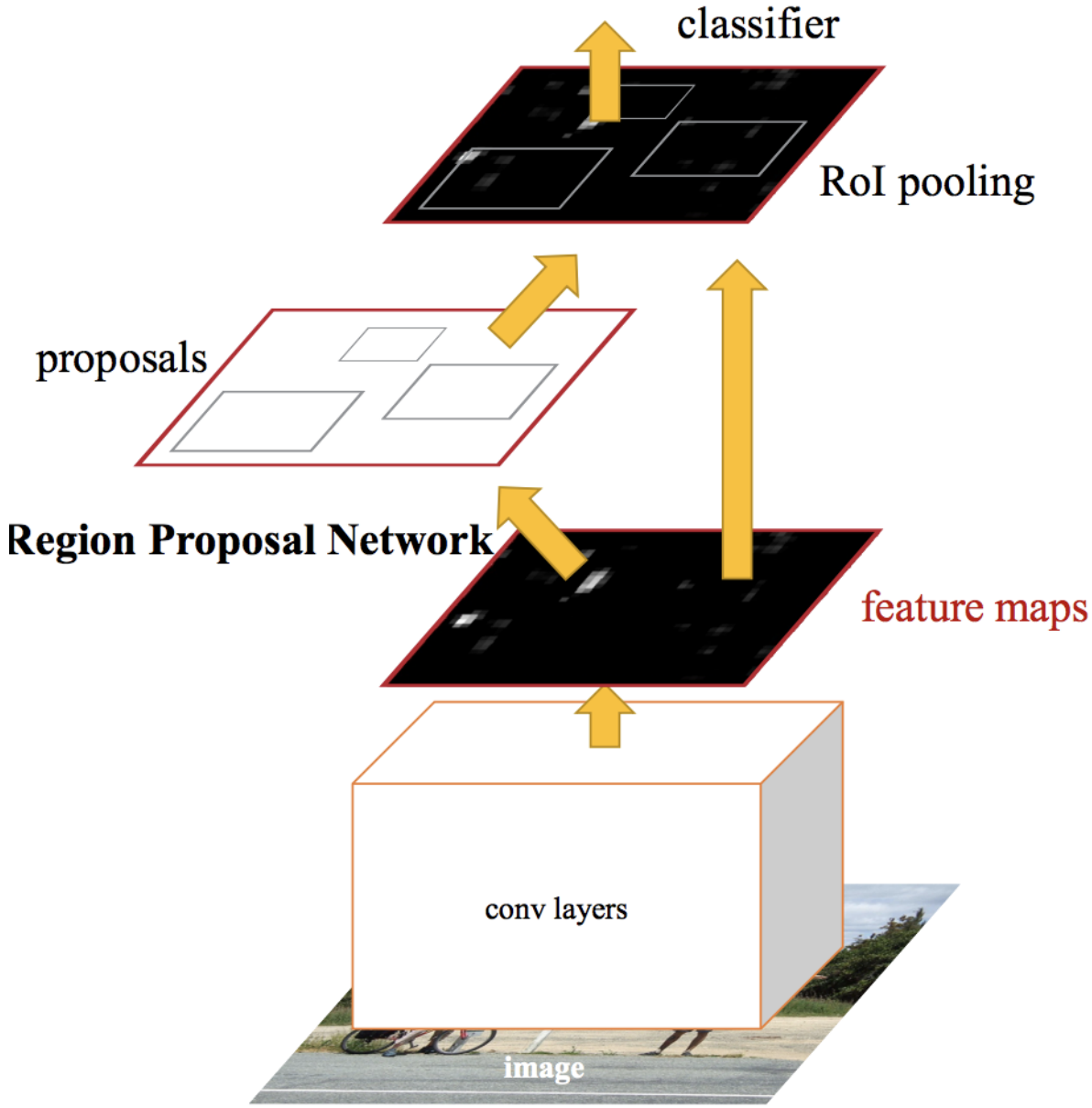
- Instead of feeding 2,000 regions directly into the CNN, Fast R-CNN feeds only the input image and generates a convolutional feature map. It then extract regions of interest based on the feature map using selective search.

Faster R-CNN

- Faster R-CNN adds another layer of network to generate regions of interest instead of using the slow selective search algorithm, making real-time object detection possible.

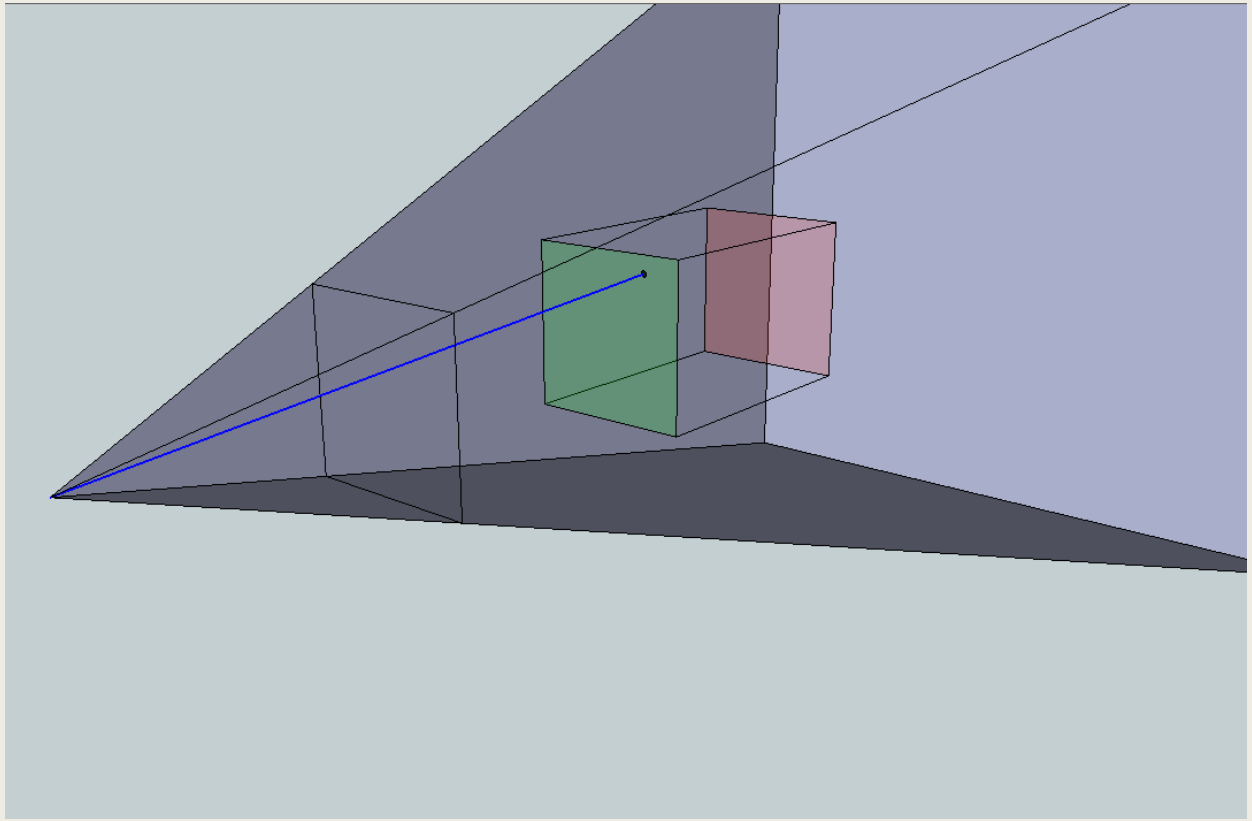

Softmax classifier
softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image


Bbox reg
SVMs
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

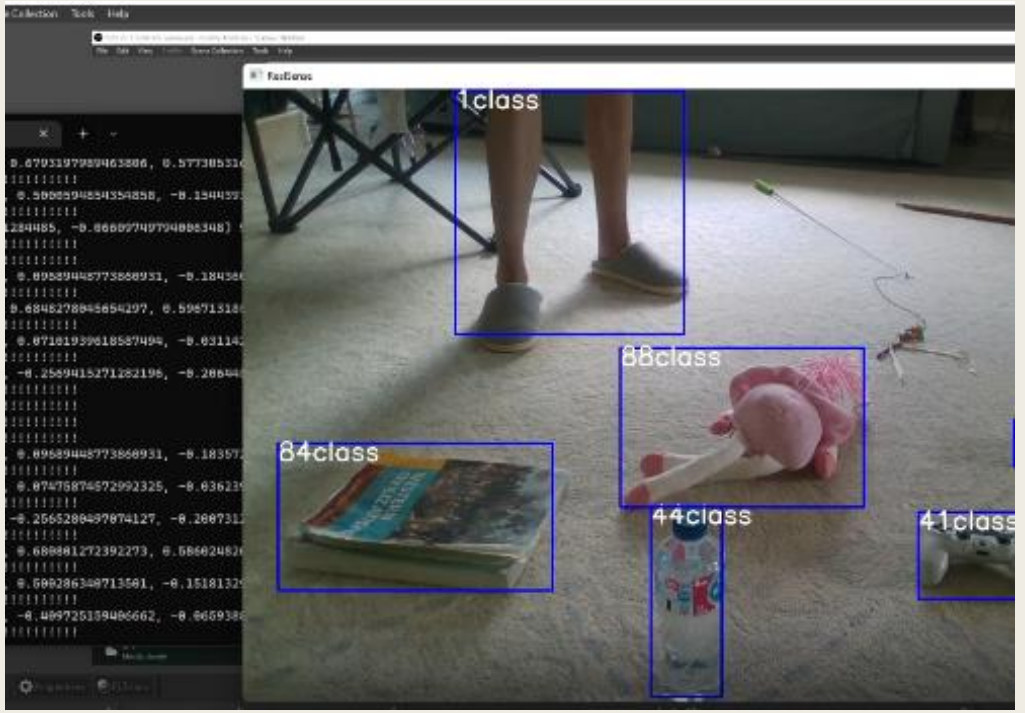
3D Object Localization and Adaption

- Ideally, the sensor provides 3D point cloud data of the surroundings, making the calculation simple because the computer can directly access the 3D location of the objects by referencing the point cloud.
- However, this method will not work in my case since the object detection program is built targeting 2D RGB images rather than 3D point clouds.
- Fortunately, the Intel RealSense SDK provides me an approach that converts the 2D coordinates of an object to its 3D location.



3D Object Localization and Adaption

- I first obtain the intrinsic data of the camera
- I then utilize the Intel RealSense built-in deprojection function, which deproject a 2D pixel into a 3D point by parsing the above information as parameters.
- The program can loop through the objects detected in the 2D image and output their 3D locations.




```
_intrinsics = rs.intrinsics()
_intrinsics.width = cameraInfo.width
_intrinsics.height = cameraInfo.height
_intrinsics.ppx = cameraInfo.ppx
_intrinsics.ppy = cameraInfo.ppy
_intrinsics.fx = cameraInfo.fx
_intrinsics.fy = cameraInfo.fy
#_intrinsics.model = cameraInfo.distortion_model
_intrinsics.model  = rs.distortion.none
# _intrinsics.coeffs = [i for i in cameraInfo.D]
result = rs.rs2_deproject_pixel_to_point(_intrinsics, [x, y], depth)
#result[0]: right, result[1]: down, result[2]: forward
```

EXPERIMENTS

Testing the Object Recognition Program Against COCO Validation Dataset

- My first experiment tests the accuracy and efficiency of my real time object recognition program, which utilizes the Fast R-CNN architecture and trained itself on COCO training data set. I test the model against the COCO validation set. The model demonstrated an average precision of 32% at an intersection over union (IoU) threshold of 0.5 and this value dropped to 17.5% when the IoU threshold was raised between 0.5 and 0.95. When the frames per second of the object detection program reached 27.5 the program ran smoothly in real-time.

ACCURACY AND EFFICIENCY OF FASTER R-CNN

Model	Region Proposals	Testing dataset	FPS	mAP@0.5	mAP@[.5, 0.95]
Faster R-CNN	50	COCO validation set	27.5	32.3	17.5

Testing The Accuracy of 3D Object Localization Program

- To test the accuracy of the localization program, I start the program under different lighting conditions and distances. Under distance less than or equal to 3 meters, the accuracy of the localization program is outstanding, producing on average an inaccuracy of 0.05 meters. At a distance in between 3 meters and 5 meters, the accuracy of the program is at 0.1 meters, and bad lighting conditions worsen the situation. At distances greater than 5 meters, no result is produced since the maximum capacity of the sensor is reached.

ACCURACY OF LOCALIZATION UNDER DIFFERENT CONDITIONS

Lighting condition	Distance from camera	Inaccuracy
Great	<=3m	±0.05m
Bad	<=3m	±0.05m
Great	>3, <=5m	±0.1m
Bad	>3, <=5m	±0.2m
Great/Bad	>5m	-

Challenges

Background Research and Implementation of APIs

- When I first started this project most of my time was committed to heavy background research online because this project involved many different libraries and repositories.
- I also spent a lot of time studying different architectures for real time object recognition.
- The debugging process during the installation of the SDK was also very demanding.

Combination of Object Detection and Localization

- The process of performing object detection prior to object localization greatly reduced human effort to handpick objects they wished to locate and provides smoother user experiences. However, the integration was very complicated and required heavy adaptation.
- I needed to find a medium that would help me link the detection program and the localization program, and it took a lot of time and effort for me to build such a medium, which is the deprojection process that maps the location of 2D pixel to its real 3D point.

Conclusion

- Overall, my approach to 3D Object Localization utilizes the Fast R-CNN architecture for object detection and Intel RealSense sensors. My approach achieves real time object localization with high accuracy under ideal conditions. It also achieved real time object detection with an accuracy of 32%.
- In the future, two key areas requiring improvement are the accuracy of the object detection program and the range of object localization. The accuracy can be improved by training the model with a better tuned training dataset. I will attempt to utilize a combination of cameras, instead of relying on a single RGB-D camera, to extend the range of localization. Combination of a LiDAR sensor and RGB camera could potentially be a choice, since LiDAR sensors usually produce more accurate depth data at a further range. I also plan to try to apply my approach to the field of autonomous navigation of robots.


THANK YOU