Final Project Report
Mutian Yan, Yinbo Chen
CSCI 5722
Ioana Fleming

# Home Moving Visualization Helper and Planning

## Induction

Our goal for this project is to help the user make a clear list of furniture that they have in their rooms and thus they could know how many moving trucks they should rent. This application would be potentially useful for those who move frequently or consider remodeling their homes. Based on the traditional computer vision concepts and modern deep learning techniques, we achieved our goal. The main architecture of the application is Mask R-CNN proposed by He et al. They came up with the idea of detecting the class labels, bounding boxes, and masks for all recognizable objects in the frame, in a timely parallel manner. It used the ResNet as a backbone and extended the depth of prior work published papers such as Fast R-CNN and Faster R-CNN. Instance segmentation problems can be thought of as the combination of object detection and semantic segmentation. Besides, it provides the fine boundaries for each instance. It fits our project well because we could use the mask to extract each object and operate each object individually.

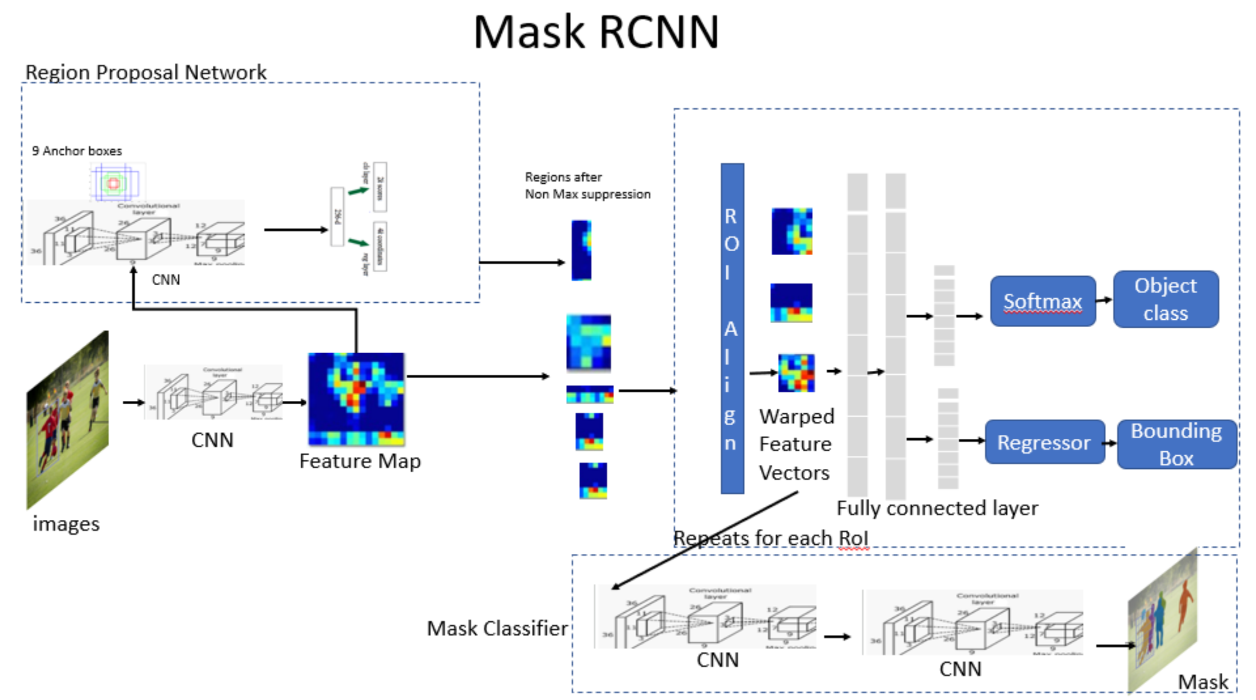
## Literature review

Traditional computer vision algorithms tackle the object detection task using some well-known feature descriptors such as SIFT, SURF, BRIEF, etc. The rise of feature descriptors (SURF, SIFT) in combination with traditional machine learning algorithms (Support Vector Machines, K-Nearest Neighbors) as the primary techniques for object detection problems in the 90s. In the era of data, the emergence and explosion in hype for deep neural networks have changed the computer vision field because of its empirical success in different image detection tasks.

CNN is popular in tasks such as object detection. CNN was used to reduce the size of images using convolutional layers and pooling layers. The reduced sized images then can be fed into fully connected layers. There are several similar object detection tasks around this topic, but each of them is focusing on a different aspect.

Semantic Segmentation focuses on segmenting image regions that share similar texture, such as grass and road. While Instance Segmentation concentrates on segmenting countable things. Since our main goal is to detect the objects inside rooms, our task is classified as the Instance Segmentation problem. We take advantage of the state-of-the-art region-based Mask R-CNN method published by He et al. There is another trending task, Panoptic Segmentation, first introduced by Kirillov et al. It segments so-called things which refer to countable things (for example, car) and so-called stuff whereas amorphous and uncountable regions (for example, sky, road). Panoptic Segmentation task leveraged the previous two tasks and, to a great extent, pushed the segmentation task towards more realistic real-life scenarios. It would be beneficial to forward our application to be able to recognize the backgrounds as well, thus the application can customize a set for furniture display styles based on the surroundings.

Mask R-CNN
It has been the state of the art in instance segmentation since 2017. It is a region-based approach that is different from the YOLO and SSD approach. Below is its architecture diagram. From the left side to right, you can see the input is images and output are object classes, bounding box, and mask.



Procedures

Setup: (Including the system requirement, environment, and packages installation)

System requirement: Windows 10 with an Nvidia graphics card.
Environment:

1. The submission code runs locally using Anaconda environment with all the required packages installed, which is written in the requirement.txt file.
2. We have also tried on Google Colab. Feel free to use it but the Mask R-CNN implementation works only with Tensorflow 1.15 from our experience.

Packages installation:

1. Pre-installed  a cuda_9.0.176 for win 10
2. cudnn-9.0 win10-x64-v7
3. Python3.6-3.7
4. Microsoft Visual C++ Build Tools

5. cocoapi
6. Pre-trained model weights (mask_rcnn_coco.h5), the notebook program will automatically download the dataset from Github if you don't have it in your working directory.

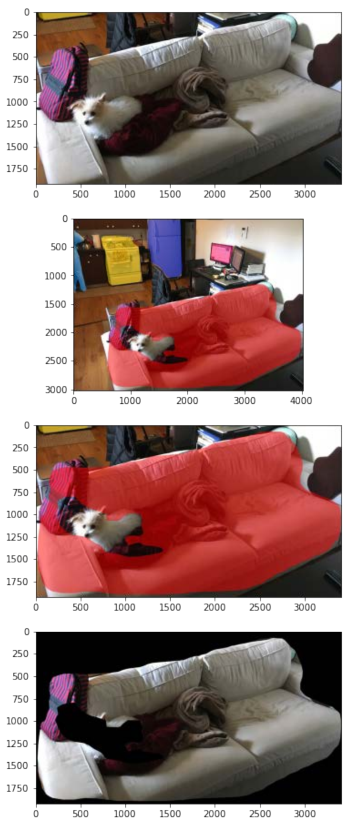
Components installation:

1. pip installs a required document that contains numpy, scipy, cython, h5py, Pillow, scikit-image, tensorflow-gpu (version =1.5), jupyter, keras(version = 2.1.6) and pandas.
2. Pip install PythonApi and perform a make and a setup
3. Python install the RCNN setup under the root folder

Note: the component tensorflow may not work on a MacOS platform. For testing purposes, changing tensorflow-gpu to tensorflow-cpu (version =1.15) .

There are three stages in this application.

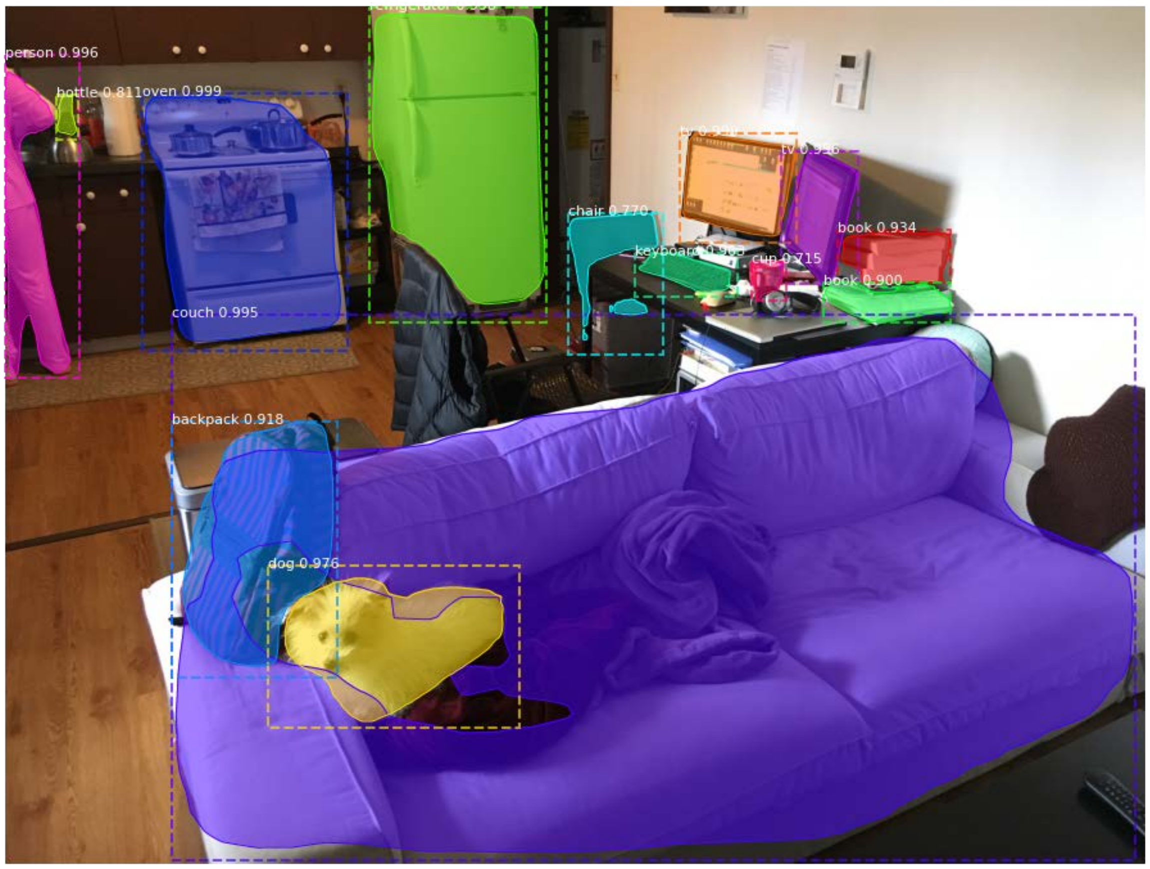
Stage 1: Detect each instance in the frame with class labels and masks in images or videos
The Mask R-CNN used ResNet as its backbone and used Feature Pyramid Network (FPN) to extract the multi-scale features by downsampling and upsampling. It then used Region Proposal Network (RPN) to scan over the FPN features to propose Region of Interest (ROI). From the below output images**,** the whole image is processed by all the ROIs linearly. It will only advance the procedure only if the current region contains an RoI object. Contrary to the previous mechanism to pass the object to the mask classifier, Mask R-CNN proposed RoI Align which used bilinear interpolation, to achieve better pixel-to-pixel alignment accuracy in the process of generating masks. Mask R-CNN then has a classifier and a bounding box regressor to produce the class labels and refined bounding box. Simultaneously, a mask generator takes the regions selected by the ROI classifier and generates them.  Due to the limit of our computation resources and lack of a labeled image dataset, our application used the pre-trained classes with weights. We captured some real-life indoor images and videos to test our implementation.



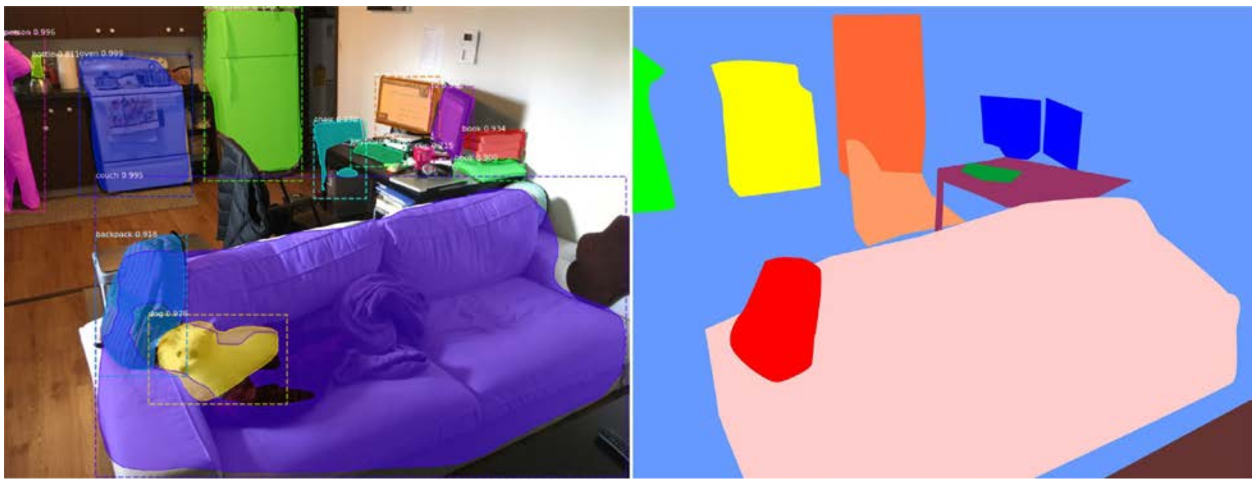
Stage 2: Conduct statistical approach to list all the objects recognized
Once the Mask R-CNN method outputs the class labels, bounding boxes, and confidence score for classification. We modified the mask by applying a different filter on and breaking the iterations for each image into parts so that we could retrieve each individual object. With the goal of counting all objects with their class label, we store the data into a dictionary.

Stage 3: Evaluation
We calculate the classification accuracy and mask accuracy using the provided benchmark. Also, we test the accuracy on a set of rotated images.

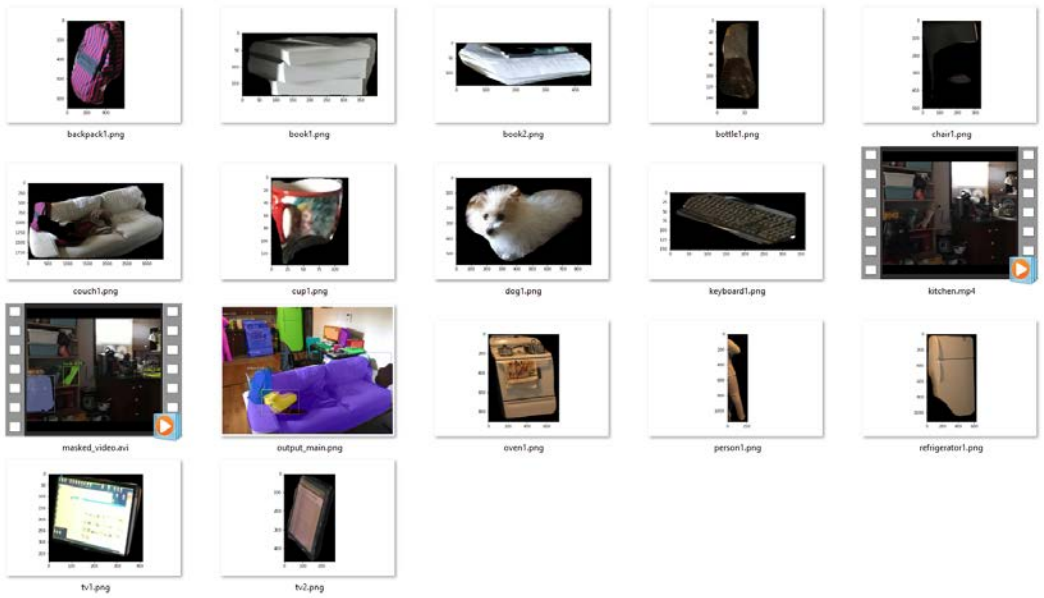
1. Perform on images



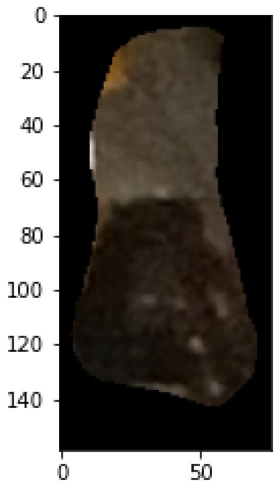
The output image from Mask R-CNN

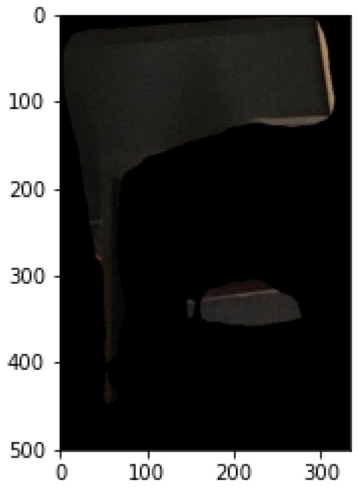


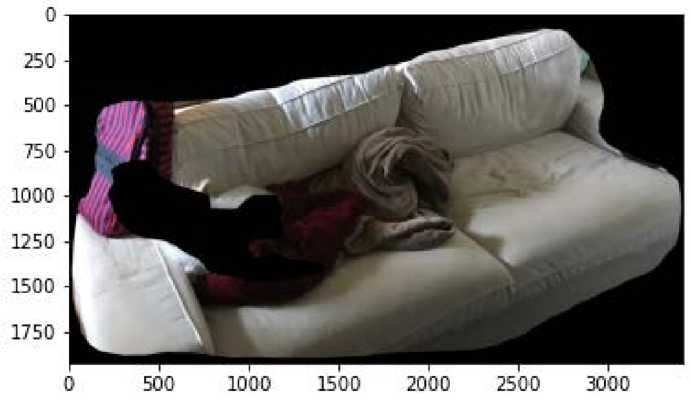
The output image v.s. The ground truth image

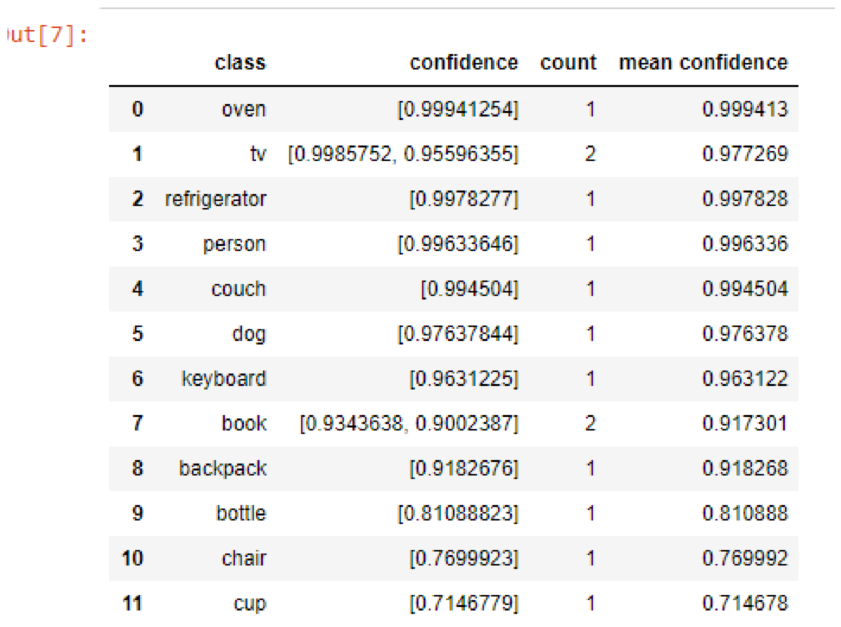


All the objects recognized image stored in the directory







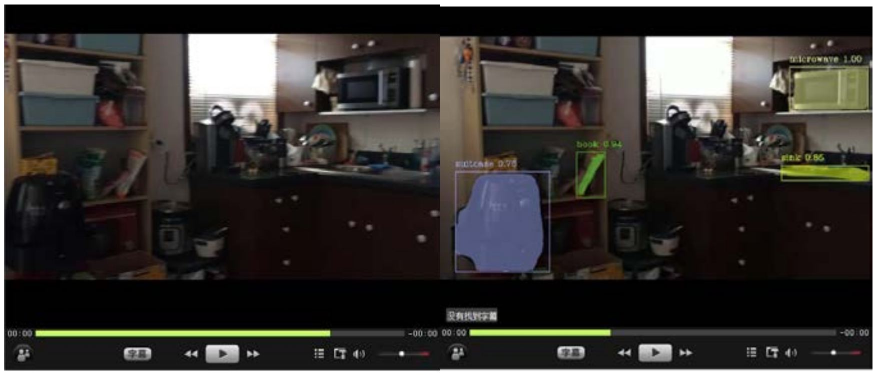


ut[7]:

|  | class | confidence | count | mean confidence |
|---|---|---|---|---|
| **0** | oven | [0.99941254] | 1 | 0.999413 |
| **1** | tv | [0.9985752, 0.95596355] | 2 | 0.977269 |
| **2** | refrigerator | [0.9978277] | 1 | 0.997828 |
| **3** | person | [0.99633646] | 1 | 0.996336 |
| **4** | couch | [0.994504] | 1 | 0.994504 |
| **5** | dog | [0.97637844] | 1 | 0.976378 |
| **6** | keyboard | [0.9631225] | 1 | 0.963122 |
| **7** | book | [0.9343638, 0.9002387] | 2 | 0.917301 |
| **8** | backpack | [0.9182676] | 1 | 0.918268 |
| **9** | bottle | [0.81088823] | 1 | 0.810888 |
| **10** | chair | [0.7699923] | 1 | 0.769992 |
| **11** | cup | [0.7146779] | 1 | 0.714678 |

2. Perform on videos

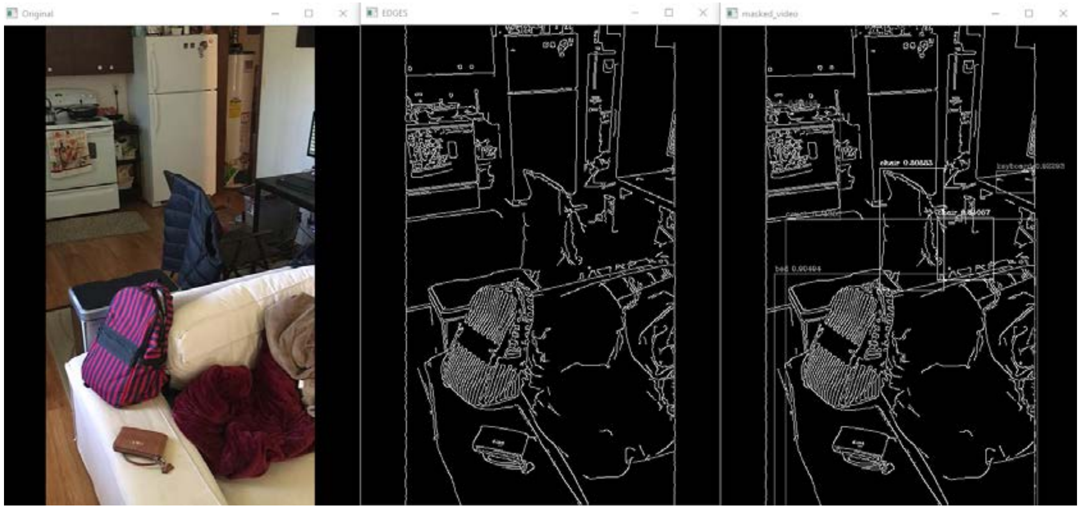
Our original purpose is counting numbers and recording the label of objects which appeared in a video to help moving home. Unfortunately, during the project, we found it's really hard to mark each object in a stream as well as telling the computer which object has already been counted. The approach is taken to count furniture in an apartment, where most objects overlap and some objects in the far distance are very small which are bare to see due to perspective view. So performing a video detection will be completely different from counting objects in an image.



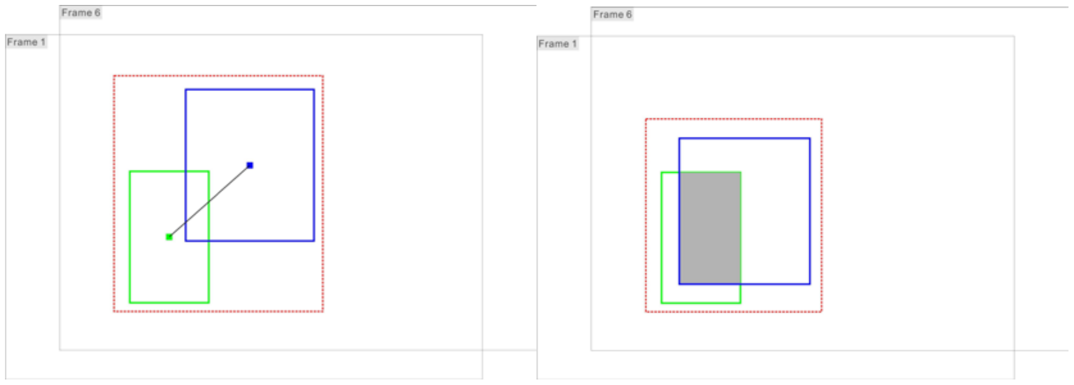
The left image shows the original video that includes a bunch of objects. The right image presents a result that four objects were detected by this application. However, two of them were misidentified. From the result we found, there are still many objects that have not been detected. It may be caused by the brightness of the video, some objects are hard to see. Also, due to the limitation of pre-trained data samples, some objects are not on the data list.

Performance:

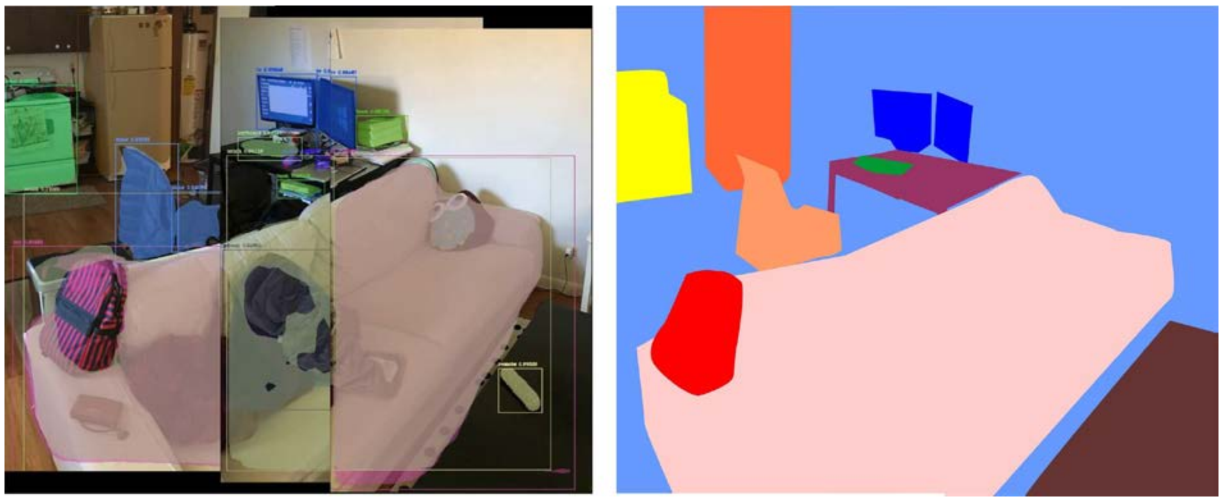
We tried to use the edges detection to detect each frame and compare it with the adjacent frame.



We experimented to compute and connect the center of the nearest object’s’ bounding boxes with a threshold by distancing between the adjacent 5 frames to project which regions have already been detected. It works while the bounding boxes appear near the center area of each frame without a big change in shape but it’s terrible when objects are partially detected or along with the edge of the image.



We also attempt to use the OpenCV tracking library, considering each bounding box coordination as an input tracking region to perform multiple objects tracking then compare automatically updated tracking regions with the newly detected bounding boxes area. Choosing the new bounding boxes as newly detected objects if they overlapped. But this process is expensive and time-consuming. Currently, we haven't implemented it. As a challenge, we will test it in the future. None of the methods performed gave us continuous and reasonable feedback. We can’t perform an accurate result of a whole stream evaluation, so we segmented the stream to multiple scenes and made a comparison with ground truth as a figure shows below.



Evaluation:

We set up a scene with 10 objects which shows on the right side of the figure above. Then we conducted a stream detection in the scene area. We extracted three random frames in the range of stream's segmentation which covered all 10 objects then combined them which displays on the left side of the figure above. The data shows below.

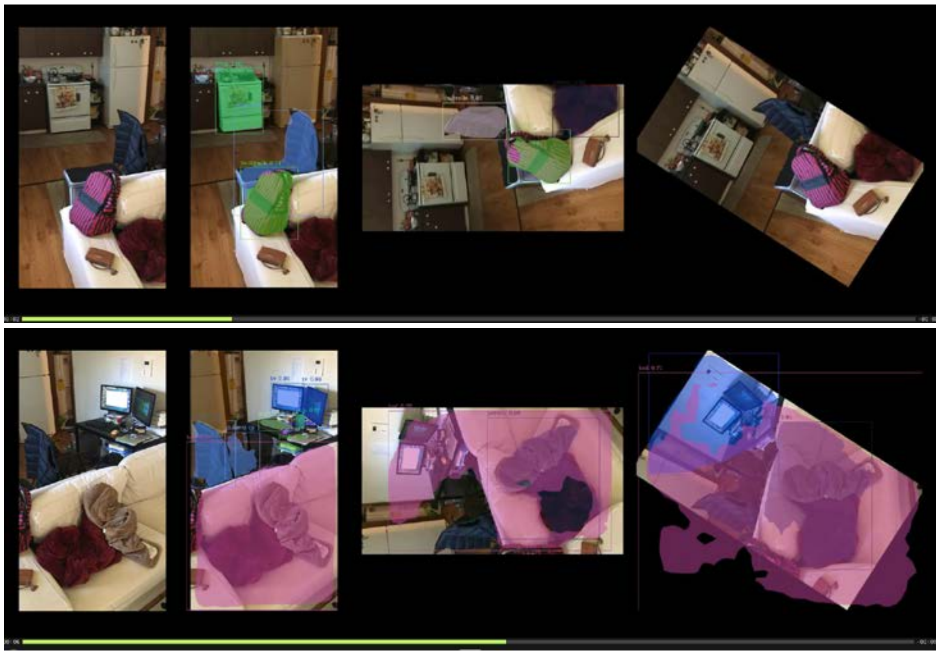
| | class | confidence | count | Mean confidence |
|---|---|---|---|---|
| 0 | oven | 0.88413 | 1 | 0.88413 |
| 1 | couch | 0.71043 | 1 | 0.71043 |
| 2 | bed | 0.83484 | 1 | 0.83484 |
| 3 | chair | 0.80803, 0.84074 | 2 | 0.82439 |
| 4 | keyboard | 0.86683 | 1 | 0.86683 |
| 5 | refrigerator | 0.89840 | 1 | 0.89840 |

| | class | confidence | count | Mean confidence |
|---|---|---|---|---|
| 0 | tv | 0.99848,0.88178 | 2 | 0.94013 |
| 1 | couch | 0.94116 | 1 | 0.94116 |
| 2 | book | 0.88168, 0.84936,0.75625 | 3 | 0.82909 |
| 3 | mouse | 0.85563,0.86571 | 2 | 0.86067 |
| 4 | person | 0.84704 | 1 | 0.84704 |
| 5 | backpack | 0.78348 | 1 | 0.78348 |

| | class | confidence | count | Mean confidence |
|---|---|---|---|---|
| 0 | tv | 0.98487 | 1 | 0.98487 |
| 1 | book | 0.78138 | 1 | 0.78138 |
| 2 | couch | 0.77018 | 1 | 0.77018 |
| 3 | bed | 0.83597 | 1 | 0.83597 |
| 4 | remote | 0.99559 | 1 | 0.99559 |

From the data, we found that many objects are identified multiple times. It also exists that the same object is identified as two different objects in one frame. To consider this scenario, we do

need more accurate algorithms to conduct object detection in future studies. The good news is 8 of 10 pre-set objects are identified through this approach. The remaining two undetected objects are either partially appeared in the video or blocked by others.
For robust testing, we performed a video detection with three different directions: Normal, Rotate 90 CCW, and Rotate 45 CCW. We made a screenshot from the video under the same timeline. The pictures display below(from left to right are the original, normal position, rotate 90 and rotate 45).



Mask R-CNN based on the rise of feature descriptors but it is limited to the numbers of pre-trained samples. From this test, it only has the highest detection accuracy when a detected object is under a normal position. Under others, it barely succeeds. The reason for the classification failure and mask mismatch might be caused by not enough variety of training data. For future object detection tasks, we would include more synthetic data with more varieties such as various directions to improve the robustness and accuracy.

Conclusion:
Mask R-CNN has a very high accuracy for most of our test images and it can finish running small size images or videos in a relatively small amount of time. Due to our capacity of computing, we would not be able to test on the raw image or video footage.

Limitation:
1. Because each image is only captured from an angel, it is impossible to measure the relative distance between objects and their dimension.
2. Since it’s an instance segmentation problem, segmenting overlapping objects tend to miss the boundary of the overlapped object.
3. The rotation of input images will affect task accuracy significantly.

Future Work:

1. Use GAN to generate the corresponding images from another angle if only one image is feasible to get.
2. Adding a camera interface for the real-time Instance Segmentation.
3. Develop better algorithms to solve the counting objects' problems, such as overlapping objects and repeated counting on one object.

References:

1. He, Kaiming, et al. “Mask R-CNN.” ArXiv.Org, 2017, arxiv.org/abs/1703.06870.
2. Lin, Tsung-Yi, et al. “Feature Pyramid Networks for Object Detection.” ArXiv.Org, 2016, arxiv.org/abs/1612.03144.
3. matterport. “Matterport/Mask_RCNN.” GitHub, 9 Mar. 2019, github.com/matterport/Mask_RCNN.
4. “COCO - Common Objects in Context.” Cocodataset.Org, 2019, cocodataset.org/#home.

Link to demo video with code:
Google Drive

1. Demo video

https://drive.google.com/file/d/1AOSrYYqNghb_U0U2xsKP8XAdi_b4GggP/view?usp=sharing

2. Comparison video

https://drive.google.com/file/d/1T--nGhJAT-zVpdZynsbVV91syP9_KCPw/view?usp=sharing

3. Code

https://drive.google.com/drive/folders/1l2abFgHFo0zb471o96kat6K7z-9rxNfU?usp=sharing

Include a summary of work separated between members

Mutian Yan: Literature Review, perform the object segmentation on images.

Yinbo Chen: Perform object segmentation on videos.