Topic: Object Segmentation

Group Members: Binbin Weng , Emily Laughead , Peiyan Wu , Rahul Javalagi

Motivation:

Image segmentation is a crucial part of understanding an image scene, and as a result it is a very active research area. Many methods have been developed in past years. We want to explore the different algorithms for image segmentation, and compare the effectiveness and performance between some of the methods. In this project, we explore the methods of Threshold Segmentation, Edge Detection Segmentation, K-Means Clustering, and Learning-based Segmentation. From the implementations of these methods, we hope to learn more about the concepts of scene understanding in computer vision.

Approaches:

Threshold Segmentation:

Threshold segmentation or Region Segmentation is a simple method of manipulating images to isolate objects within the image. There can be multiple different implementations of threshold segmentation utilizing various colorspaces, but the most common method, and the method we implemented, involves converting the image to grayscale and adjusting the pixel values based on a set threshold. Depending on the pixel intensities of the input, the ideal threshold can vary drastically from image to image.

For our implementation, we utilized Otsu's threshold. In short, Otsu's method finds the ideal threshold for separating an image into the background and foreground by separating the histogram of the image into two clusters. We then applied this threshold and multiple fractions of it to a gaussian blurred grayscale version of each image to yield the segmented image. The specific thresholds we used in our implementation were (Otsu's + Otsu's/2), Otsu's, and (Otsu's/2). The gaussian blur creates more separation and reduces graininess in the final image. From there, we inverted the image for better visibility.

Edge Detection Segmentation:

Edge Detection Segmentation is a way to manipulate images based on their edge pixels. In edge detection, edges and edge pixels are linked and combined to try and form the entire object. This can be performed with either local or global processing. This approach works very well in images with good contrast between foreground and background, and performs relatively poorly in images that have low contrast or smooth transitions. The method is also sensitive to noise.

For our implementation, we used the Canny Edge Detector Python library. The Canny Edge Detector is a form of optimal edge detection that addresses the fact that for edge detection, there is a tradeoff between noise reduction and edge localization. In this algorithm, the image is first smoothed with a Gaussian filter, the gradient magnitude and orientation are computed, the non-maximal suppression is applied to the gradient magnitude image. Finally, hysteresis thresholding is used to detect and link edges. It should also be mentioned that this algorithm can only work when the images are converted to greyscale.

K-Means Clustering with Subtractive Clustering:

K-Means Clustering is a method of dividing a set of data into K groups. It consists of two phases. The first phase is to find K centroids, and the second phase is to divide all the points to K groups. We implemented the method introduced by the paper, Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm, combined with the Laplacian Pyramid. The steps of this method is as the follows:

1. Do contrast sketching on the original image to improve the contrast by "stretching" its range of intensity values and generate the contrast stretched image.
2. Use the Elbow method to determine the value of K. The Elbow method is calculating WCSS (within-cluster sum of square) for different values of K (from 1 - 10), plotting WCSS with values of K, and selecting K at which the decreasing speed of WCSS becomes small.
3. Do Laplacian Pyramid to downsize the contrast stretched image to generate the downsized image

4. Use subtractive clustering to initialize K centroids on the downsized image. For the basic K-Means, we normally randomly select K centroids at the beginning. But the initial K centroids may affect the final results. To avoid the randomness, we manually select K initial centroids with subtractive clustering.
5. Do K-Means with the K initialized centroids on the contrast stretched image and generate segmented image
6. Apply the median filter on the segmented image to remove noises.

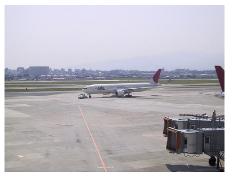
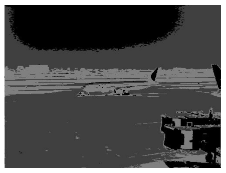
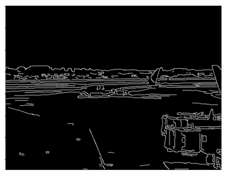
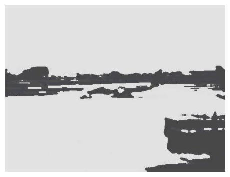
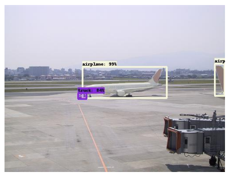
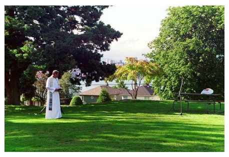
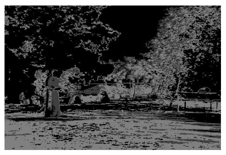
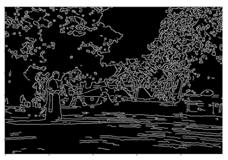
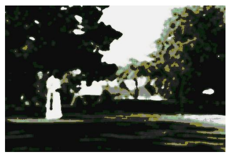
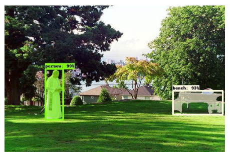
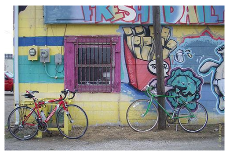
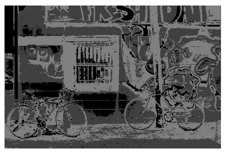
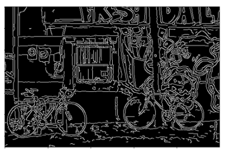
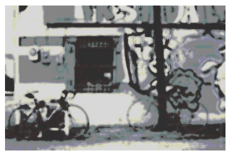
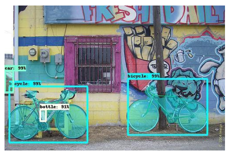
Mask R-CNN Segmentation:

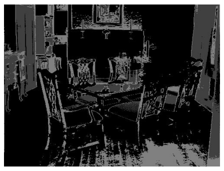
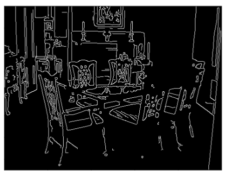
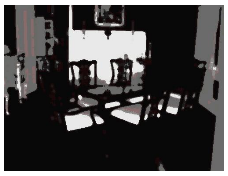
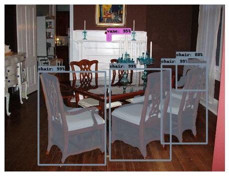
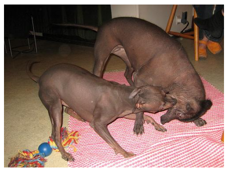
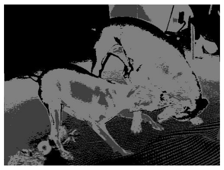
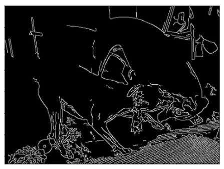
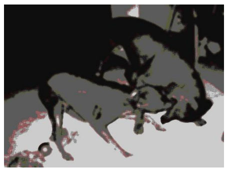
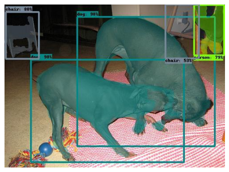
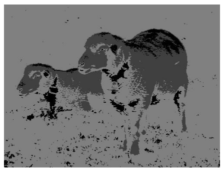
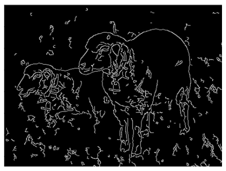
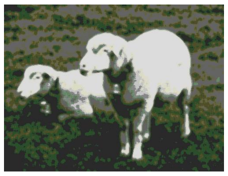
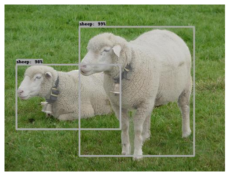
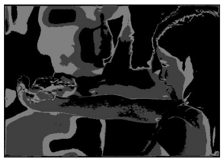
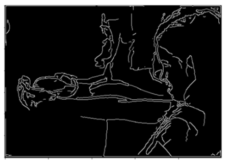
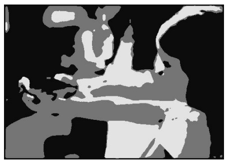
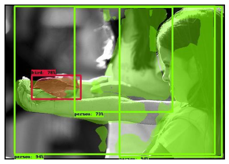
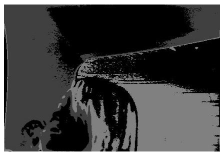
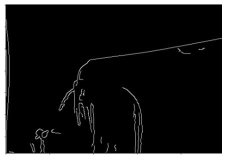
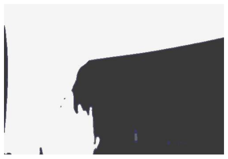
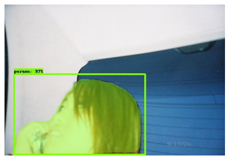
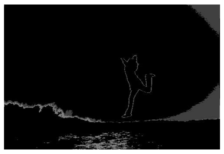
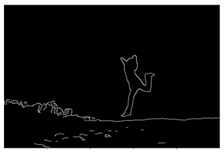
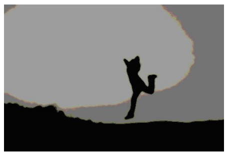
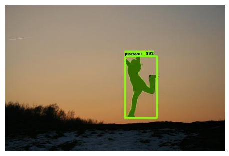
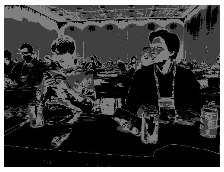
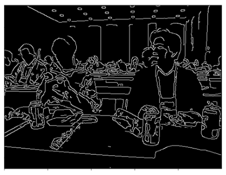
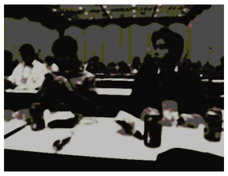
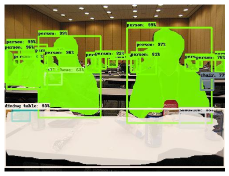
Mask R-CNN is a deep learning model for performing image segmentation and detection. It is a two-stage model that shares the same backbone convolutional layers between two stages to decrease computational cost. We used PyTorch to try implementing this model from scratch using DetNet59-FPN as the backbone. When evaluating, the model works as the following:

1. The backbone convolutional network produces a feature map pyramid from the input image.
2. The feature maps are then fed into a region proposal network (RPN) that proposes bounding boxes of different aspect ratios at each N by N pixel window on the feature maps along with an objectness score that predicts whether an object is present in that bounding box.
3. We then apply non-maximum suppression to the proposal bounding boxes to avoid highly overlapping proposals, and the N proposals that have the highest objectness scores are selected as final proposals. These proposals are called regions of interest (ROIs).
4. Regions on the feature maps are then pooled into sets of fixed-sized feature maps corresponding to each proposed ROIs.
5. The pooled feature maps are then passed into a detection branch and a mask branch. The detection branch predicts the classification of the object present in each bounding box, and predicts further refinements of the bounding boxes.
6. The mask branch predicts segmentation mask of the object present in each bounding box for every classification.
7. The final segmentation mask is then selected based on the predicted classification.

Unfortunately, our implementation of the Mask RCNN model is unfinished due to technical difficulties described in the challenges section below. After consulting with a TA, we used an off-the-shelf Mask RCNN model pretrained on COCO dataset to produce results for below comparisons.

Results:

No	Original image	Threshold Segmentation	Edge Detection Segmentation	K-Means	Mask R-CNN
1					
2					
3					

4					
5					
6					
7					
8					
9					
10					

Comparison:

From the results of Threshold Segmentation, it is clear that certain types of images work better than others. Specifically, images that have a defined foreground and background, like images 5, 6, and 9 were segmented very well. Images with similar pixel intensities, like images 7 and 8 lost quite a bit of detail and were not segmented well. Overall the RMSE between the images had a variance of at most about 0.01 and the PSNR between images had a variance of at most about 1. However, all RMSE values for this method were clearly higher, and PSNR clearly lower, when compared to the other segmentation methods.

From the results of Edge Detection Segmentation, it is definitely evident that some images are far better for this method than others. The best performing images were ones like image 1, image 7, and image 10. Images like image 6 and image 9 which had low or similar contrast in parts performed worse. Overall, this method performed better than we were expecting, and did decently well on all the provided images. Please see below the RMSE and PSNR results for the ten images.

From the results of K-Means, we can find that the method is segmenting the pixels based on their values. Pixels with similar colors are segmented to the same groups. For example, for image 4, the frame of the chairs is of the similar color as the floor, so the chair frame is segmented in the same group as the floor. From these results, we can also see that if the object is of a single color or a few similar colors, the object is more likely to be segmented in one group, like the image 1, 5, 9; if the object is of different colors, different parts of the object may be segmented to different groups, like image 2, the face and cloth of the person are in different groups.

For the results of the learning-based method (Mask RCNN), since the model directly outputs bounding boxes, segmentation masks, and classification labels for each recognized object, unlike the results from other methods, there is little need for humans to identify what each segmented layer represents. The advantage of a learning-based segmentation model is that for a model that's trained on a large dataset, the model produces very accurate results on most non-specialty images, as can be seen from our results. However, it also has disadvantages. The model can only produce segmentations and classifications for objects that belong to a set of predefined classes. For example, the model we used was trained on the COCO dataset, which doesn't define classes like trees, ground, or sky. As a result, if we want to also have a segmentation for the trees and sky like in image 2, our learning-based model has no way to produce such segmentation, and the K-mean method would give a much better result. Furthermore, a learning-based model requires a lot of computational resources to train, while more traditional methods are less computationally expensive.

In order to compare the results of the different models, we calculate the Root Mean Square Error (RMSE) and Peak to Signal Noise Ratio (PSNR). RMSE measures how much the segmented image is deviated from the original image. PSNR is the proportion between maximum attainable powers and the corrupting noise that influence likeness of image. Since the output of the Mask RCNN model is not a single image, but class labels, bounding boxes and masks, this metric is not suitable for measuring the quality of result from Mask RCNN and thus not calculated for it.

	RMSE (Lower is better)			PSNR (Higher is better)		
Image number	Threshold Segmentation	Edge Detection Segmentation	KMeans	Threshold Segmentation	Edge Detection Segmentation	KMeans
1	0.0152982	0.0013717	0.0008632	66.284	76.757	78.769
2	0.0147954	0.0013860	0.0013197	66.430	76.712	76.926
3	0.0150585	0.0013342	0.0010944	66.353	76.870	77.739
4	0.0147632	0.0013987	0.0015798	66.439	74.595	76.145
5	0.0147196	0.0013501	0.0011904	66.452	76.826	77.374
6	0.0150011	0.0014177	0.0013038	66.370	76.614	76.979
7	0.0143010	0.00124445	0.0012124	66.577	77.180	77.294
8	0.0148708	0.00140574	0.0008967	66.408	76.631	78.294
9	0.0140806	0.0012560	0.0011731	65.323	77.126	76.116
10	0.0145420	0.0013224	0.0014896	66.505	76.865	76.400

From a quantitative standpoint, KMeans performed the best, due to it having the lowest RMSE and highest PSNR values. However, from a qualitative standpoint, the Mask RCNN model was the most accurate in identifying and masking the objects in each image.

Challenges:

One of the biggest challenges we encountered in this project is the implementation of Mask RCNN. Although we were able to fully understand the architecture of the Mask RCNN network after reading all the RCNN-family, ResNet and FPN papers and implementing most of the model, we were not able to debug it fully due to insufficient understanding of PyTorch. We encountered a fatal bug when testing the region proposal network (RPN) of the model. While the bug doesn't affect the output of RPN and the RPN can produce expected output in a forward pass, the bug prevents the model from being trained. The main failure mode of our implementation is that during training of RPN, the GPU memory consumption keeps increasing after each batch, and eventually the program crashes due to out-of-memory error. After researching online, our best guess is that some pytorch tensors that should've been detached from the compute graph of the network weren't detached, and the compute graph keeps accumulating in certain tensors, causing the GPU memory to keep increasing. However, because of the complex tensor manipulations in the model and loss function, we were unable to pinpoint the problem with our limited knowledge of PyTorch. Due to time constraint, we decided to not proceed with this implementation after spending more than 40 hours trying to make it work.

Reference:

https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/
http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html
https://link.springer.com/chapter/10.1007/978-3-540-69139-6_146#:~:text=Contrast%20stretching%20(often%20called%20normalization,the%20image%20type%20concerned%20allows.
https://journals.sagepub.com/doi/full/10.1177/1550147719877612
https://stackoverflow.com/questions/42257173/contrast-stretching-in-python-opencv
https://www.sciencedirect.com/science/article/pii/S1877050915014143
http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html#devkit
http://www.cyto.purdue.edu/cdroms/micro2/content/education/wirth05.pdf
https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/
https://learnopencv.com/otsu-thresholding-with-opencv/
https://www.geeksforgeeks.org/region-and-edge-based-segmentaion/
https://arxiv.org/pdf/1504.08083.pdf
https://arxiv.org/pdf/1506.01497.pdf
https://arxiv.org/pdf/1703.06870.pdf
https://arxiv.org/pdf/1512.03385.pdf
https://arxiv.org/pdf/1804.06215.pdf
https://arxiv.org/pdf/1612.03144.pdf
https://colab.research.google.com/github/tensorflow/tpu/blob/master/models/official/mask_rcnn/mask_rcnn_demo.ipynb