People Counter Using Multiple Object Tracker

Alexis Bruneau, Ryo Teraoka
AER1515 Perception for Robotics
University of Toronto
Toronto, Canada
alexis.bruneau@mail.utoronto.ca, ryo.teraoka@mail.utoronto.ca

***Abstract*—To avoid any incidents caused by overcrowding, it is necessary to limit the number of people in small places, such as stores and restaurants. Thus, we propose an algorithm to track and count people using YOLOv7. Our method performs multiple object tracking to count the flow of people going in and out of an indoor area in real-time. The algorithm consists of three parts in the following order. First, using YOLOv7 to detect multiple people as bounding boxes. Second, using the bounding boxes information between two frames to track people. Finally, setting up two different areas around the entrance of the store to count the total number of people. We trained our model with the training set from MOT15 to yield the best MOTA and MOTP scores. As a result, our model achieved a MOTA score of 94.3% and a MOTP score of 39.1%.**

I. INTRODUCTION

A. Motivation / Applications

Covid19 restrictions are lifted in many regions recently. Thus, more and more people are starting to go out to different places all at once. Consequently, various places have become more crowded than expected. A serious accident in Itaewon, South Korea, caused by overcrowding in a confined space resulted in multiple deaths. To avoid this kind of incident, it is necessary to count people in small places, such as stores and restaurants.

As mentioned, keeping a count of people going in and out of a region is important. However, keeping an accurate count in a bidirectional manner has been a challenge. One of the first methods implemented was using a break beam sensor. However, this method is unidirectional and cannot distinguish between traffic going in or out. A method that would allow bidirectional traffic count would be to track people going from one area to another. Using camera vision to perform object detection tracking would allow us to do this.

Object detection has always been a challenge in computer vision. However, in recent years, it has become a lot easier to detect objects quickly and accurately using computer vision. YOLOv7 employs CNN to perform real-time object detection and has proven to detect humans accurately [1].

We implemented and developed an algorithm that uses YOLOv7 detection coordinates to track and count people. With this approach, it would be possible to understand the flow of people going in and out of a store. Our method would perform multiple object tracking to count the flow of people going in and out of an indoor area.

For many indoor settings, the count of people must not exceed the max occupancy for security reasons. Our algorithm would keep track of the total amount of people currently in a store and would warn the user if the count approaches the max occupancy. Our approach would be easy for the users to set up as the only modification required would be changing the region of interest for the camera. Some examples where our method could be used would be at the entrance of bars, clubs, or indoor concerts.

B. Related Work

Various research has been proposed to achieve better results in object tracking and counting. Sergiu and Adrian achieved bidirectional counting by performing the following three main steps: image enhancement, blob detection, and blob tracking [2]. Their blob detection was used to recognize when it was a human. They returned a bounding box with width, height, and (X, Y) coordinates for each blob detected. To track the blobs, they compared the current frame to the previous one and used the closest distance to track them. They processed the blobs by either adding, merging, or deleting them from a list. To track and count people they used the following steps. They divided their video in half with a horizontal line. Looking at the (X, Y) coordinates they assign a region to the blob. If that region changes between 2 frames for the same blob, the count is adjusted [2].

Sergio Et Al used three different deep-learning object detectors (EspiNet, Faster-RCNN, and YOLOv3) to count people going in and out of the metropolitan train. Their best results were obtained by using Faster-RCNN and YOLOv3. In their study, the YOLOv3 model was trained on the COCO dataset. Their approach for Multiple Object Tracking was like the approach of [2]. They compare the state of detection at time t versus t + 1 (frames), but they used the Kalman Filter Tracking as their point tracker with a distance measure using Euclidean distance. An issue they found was that most images in COCO's dataset are captured on semi-frontal views of people standing or walking. Using the pre-trained weights from YOLOv3 produced poor results for them. They found that the change in view angle had adverse effects. After using their images and annotating them, they achieved great results using YOLOv3 [3]. Using those techniques, they achieved a MOTA score of 80%.

It should be noted that state-of-the-art repository exists to perform multiple object tracking such as FairMot [4], ByteTrack [5], Ab3dmot [6], and more. However, this was not the scope of our project. In this paper, we analyzed how our

results would compare when using a similar tracking algorithm as the papers mentioned but using pre-trained weights from YOLOv7.

II. Methodology

A. Overview

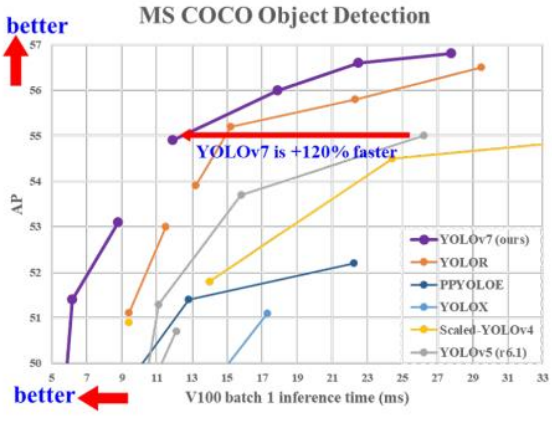
This project aims to achieve real-time counting of people entering and exiting a specific area by object detection and multiple object tracking algorithms using YOLOv7. To accomplish this goal, we assume that a camera is installed on the ceiling of a store so that it can capture its entrance area. The camera is placed in the ceiling corner and is looking vertically down at an angle.

The algorithm for this project consists of three parts: first, using YOLOv7 to detect multiple people as bounding boxes; second, using the center of the bounding box to track people; and third, setting up two different areas around the entrance of the store to count the number of people.

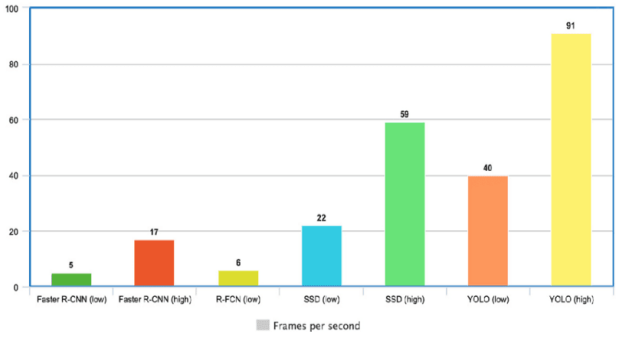
We tuned three parameters to obtain the best metric score on a test dataset. The parameters we tuned are the confidence threshold, IoU, and distance threshold. The metrics we chose to determine the best parameters are the following MOTP (Multiple Object Track Precision) and MOTA (Multiple Object Track Accuracy) which are common metrics used for Muli-Object Tracking [7]. To Train our model, we utilize MOTChallenge dataset [8].

B. Detection – YOLOv7

YOLOv7 is a state-of-the-art real-time object detection used in many applications such as autonomous driving, robotics, and medical image analysis. Its accuracy and speed have proven to be up to 120% quicker and more accurate than other methods in **Error! Reference source not found.** The application requires detecting and tracking people in real-time. Therefore, its detection speed is important to accurately count the number of people.



(a) Comparison between different version of YOLO [1]



(b) Comparison between different object detectors [9]

Fig. 1. YOLOv7 vs Other Object Detector

YOLOv7 works in the following four steps. First, it performs a residual block by dividing the image into N grids, each of them divided into SxS equidimensional regions. Next, a bounding box regression is performed. Each of the N grids divided during the residual block process is responsible for detecting and localizing the objects contained within it. However, many duplicate predictions occur because multiple grids predict the same object with different bounding boxes. Therefore, as a third step, non-max suppression is required to suppress all bounding boxes with low probability scores. To further increase the probability, Intersection over Union (IoU) can be applied between duplicated bounding boxes. IoU can be computed with the following equation.

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ Union} \quad (1)$$

Finally, based on the probability, the bounding box that achieves the highest IoU is the area selected. Through the above four steps, YOLOv7 outputs the bounding box coordinates "$(x_{left\_bottom}, y_{right\_top}, width, height)$" as the result of object detection.

YOLOv7 provides several pre-trained models as weights for object detection using the COCO's dataset. As shown in Table 1, YOLOv7 provides the fastest speed with an average of 2.8 ms for batch 32. We opted to choose this model for its speed. If the results of this model were low, we would've tested a different model such as YOLOv7-E6E [1].

Table 1 Comparison with different pretrained weights

Model	Test Size	AP^{test}	$AP_{50}{}^{test}$	$AP_{75}{}^{test}$	batch 1 fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161 *fps*	2.8 *ms*
YOLOv7-X	640	53.1%	71.2%	57.8%	114 *fps*	4.3 *ms*
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84 *fps*	7.6 *ms*
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56 *fps*	12.3 *ms*
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44 *fps*	15.0 *ms*
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36 *fps*	18.7 *ms*

C. Multiple Object Tracking and Counting Algorithm

1) Tracking

YOLOv7 returns information that is important for the tracking and counting section of the algorithm. We used the information from all the bounding boxes between two frames to track and assign IDs to people. As mentioned, YOLOv7 detects people and returns the following information for each detection (x_{center}, y_{center}, width, and height) of the bounding boxes in pixels.

To track people, we used the center coordinates of bounding boxes and compare the Euclidean distance between the current (t) and previous $(t-1)$ frame.

$$D = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \quad (1)$$

We perform the brute-force search to compare all the detected points of the current frame to the previous one. We assume that the smallest Euclidean between points corresponds to the same person. We created two dictionaries to store the information of ID. One dictionary for the current frame D_t and one for the previous one D_{t-1}. We assign values to the ID in the dictionary such as its center coordinates and current region. Unlike (Mezei, 2010), we used a dictionary instead of updating our values in a list. This allowed us to store multiple values easily to the IDs and is faster to look up items.

There were 4 cases we had to model when updating our dictionary. They are represented in the equations below where $len(D_{t-1})$ is the number of people detected in the previous frame, and $len(D_t)$ represent the number of people detected in the current frame.

$$Case\ 1: len(D_{t-1}) = 0\ \&\ len(D_t) > 0$$

$$Case\ 2: len(D_{t-1}) = len(D_t)$$

$$Case\ 3: len(D_{t-1}) < len(D_t)$$

$$Case\ 4: len(D_{t-1}) > len(D_t)$$

Case 1 represents that no one was in the region of interest, and someone entered it. In this case, D_{t-1} get assigned to be equal to D_t for the next frame.

Case 2 represents that between 2 frames, the same number of people are in that region. In this case, we perform the brute-force search to compare each point of D_t and D_{t-1} to calculate the smallest Euclidean distance. Then, we update D_{t-1} for the next frame. It is to note that in this case, the ID of D_{t-1} stays the same, but we replace its coordinates with the match found of D_t.

Case 3 represents that someone new entered the region of interest. In that case, we compare each point, and all the best matches get updated in D_{t-1}. Then, we verify which ID in D_t was not assigned and add it to D_{t-1}.

Finally, Case 4 represents someone who left the region of interest. In this case, we compute all the best-matched matches. Then, we look at which values of D_{t-1} was not matched and remove them.

2) Counting

In our model, we created two regions that share a border. In each frame, we look at the coordinates x_{center}, y_{center} of the bounding boxes and verify if they lie within one of the regions. If they do, we update the region status of that ID in the dictionary. Between each frame, we compare for each frame if their region status changed. If it did, we update the total count accordingly. It is worth mentioning that our model starts with a count of 0 when it starts. The figure underneath illustrates how are regions are set up.

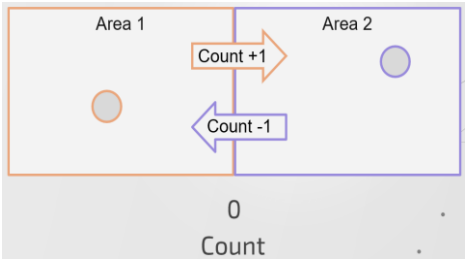


Fig. 2. Boundary Setup Count 0

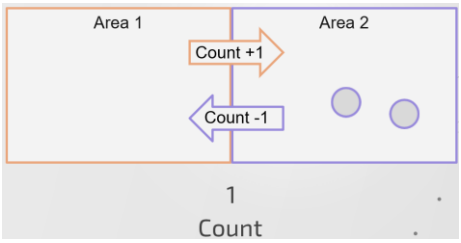


Fig. 3. Boundary Setup Count 1

3) Tuning

We chose the following three parameters to obtain the best results. The first parameter is the object confidence threshold of YOLOv7. This threshold sets the minimum confidence threshold of the model for detected objects. A low threshold can remove false object detections due to low box confidence. However, a high threshold can miss the detection of people that had average scores. The second parameter is the IoU threshold. This removes all bounding boxes with probability scores below the threshold. The third threshold is a distance threshold for properly tracking people and counting people entering and leaving a particular area; YOLOv7 achieves this by using bounding box coordinates as the output of object detection. However, this output contains noise due to human shadows and environmental brightness. As a result, the person's position can jump far away and is not considered to be the correct person's motion. Therefore, if the Euclidean distance between the person in the previous frame D_{t-1} and the current frame D_t is greater than the distance threshold, the person is not considered the same person, and its coordinates are not updated.

Table 2 Mapping of ID

	Ground Truth					Detected Output					
Frame	ID	Bottom_left	Top_right	Width	Height	ID	Bottom_left	Top_right	Width	Height	**Assigned ID**
8	9	481	163	31	75	person0	587	230.7	23	85	**19**
8	15	301	210	32	87	person1	292	213.6	40	86	**15**
8	19	572	241	40	84	person2	483	172.9	30	78	**9**
9	9	478	165	31	75	person0	571	232.9	34	87	**19**
9	15	307	208	31	87	person1	302	213.6	30	88	**15**
9	19	562	240	40	84	person2	478	172.9	29	77	**9**
10	9	474	166	31	75	person0	556	232.1	48	85	**19**
10	15	314	207	31	86	person1	309	211.4	27	87	**15**
10	19	552	240	39	84	person2	472	172.9	32	77	**9**
10	NA	NA	NA	NA	NA	person4	750	294.3	18	32	**Overestimate**

D. Evaluation Metrics

Three errors were considered for tracking: False-Negative,False-Positive, and ID Switch. False-Negative consists of detecting an object does not present in the ground truth. FalsePositive consists of detecting an object that is not present in the ground truth. Finally, ID Switch consists of having a switch between the correspondence of Ground Truth and the detection output between two frames. Using those errors, two tracking metrics are calculated. Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) are computed to validate our model.

MOTA measures the overall accuracy of both tracking and detection with the equation in the following. However, this metric does not include localization error, and detection performance outweighs its association performance. MOTA's possible score lies in the following range $[-\infty, 1)$ where 1 represents a perfect score, and any lower scores are worse. MOTA can be calculated using the following equation 3.

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDS_t)}{\sum_t GT_t} \quad (3)$$

Where FN_t is False negative, FP_t is False Positive, IDS_t is Identity switch, and GT_t is Ground truth object count. Identity switch is the identity differences between the same person in the previous flame and the current flame. Ground truth object count means the number of detected objects in the ground truth. MOTP measures the accuracy of localization of detection boxes. It is the average dissimilarity between all true positives and their corresponding ground truth. This metric quantifies mostly the localization accuracy of the detector but does not provide much information about the performance of the tracker. MOTP is calculated with the following equation 4.

$$MOTP = \frac{\sum_{i,t} d_t}{\sum_i c_t} \quad (4)$$

Where d_t is the distance between the centroid of objects in the ground truth and the detection output. c_t is the total matches made between the ground truth and the detection output [7]. The distance d_t can be expressed as the following expression:

$$d_t = 1 - IoU \quad (5)$$

MOTP score can range between (0,1) but usually ranges between $50 - 100\%$ where a lower score is better.

E. Training / Testing

To tune our model to have the best accuracy when facing unseen images, we need to train it using a dataset that contains ground truth. The dataset from MOTChallenge was used. More precisely, we used PETS09 [10]. Among the 11 training sets in MOT15, PETS09-S2L1 was selected since its camera angles are close to our assumed situation. A frame of our dataset can be seen in the figure underneath Fig. 4. The ground truth contains the following information that is pertinent to our calculations:

Frame, ID, x_{bottom_left}, y_{top_right}, width, height

However, the IDs assigned in our model do not match the IDs assigned in their ground truth. Therefore, MOTA and MOTP cannot be calculated without further steps. We had to map our ID to the ones from the ground truth.

Greedy matching based on the bounding boxes IoU was applied to assign the IDs correctly. For each frame, we loop through each detected bounding box one by one. Each detected bounding box is compared to each ground truth of the same frame, and the one with the highest IoU is considered the same person. When a ground truth ID is assigned, it cannot be assigned again within the same frame. It is to note that this method does not consider all possibilities. Hence, this might result in finding a local minimum only. The following steps were followed to map the IDs.

Fig. 4. Ground Truth Video

Table 3 Evaluation Metrics Score with Different Hyperparameter

Hyperparameters		Evaluation Metrics				
Confidence	IoU	MOTA Score	MOTP Score	False-negative	False-positive	ID Switch
0.15	0.35	92.9%	39.7%	42	287	1
0.25	0.35	94.0%	39.2%	50	227	4
0.35	0.35	94.3%	39.2%	58	202	3
0.15	0.45	92.6%	39.9%	39	302	1
0.25	0.45	93.8%	39.2%	47	237	4
0.35	0.45	94.2%	39.2%	54	211	3
0.15	0.55	92.3%	40.0%	37	316	4
0.25	0.55	93.7%	39.2%	44	245	4
0.35	0.55	94.3%	39.1%	51	212	3

Table 2 illustrates the mapping of our detection from frame 8 to 10. When our model detects a False Positive, we assigned their ID as overestimate.

Once the IDs are assigned correctly, MOTA and MOTP can be calculated. From there, we can tune the chosen hyperparameters. Finally, for the best results on the training set, we can apply the results on the test video.

III. Results

A. Training

After training, we were able to achieve good MOTA and MOTP scores. The best results obtained were a MOTA score of 94.3% and a MOTP score of 39.1% using an IoU threshold of 0.35 and a Confidence threshold of 0.55. The average MOTA score across our test was 93.6% with a standard deviation of 0.72%. The average score for MOTP was 39.4% with a standard deviation of 0.331%. We used three values for the IoU and Confidence threshold. We used 0.15, 0.25, and 0.35 for the Confidence threshold and 0.35, 0.45, and 0.55 for the IoU threshold. The distance threshold was kept to 0 for those steps. All the results obtained can be seen in Table 3.

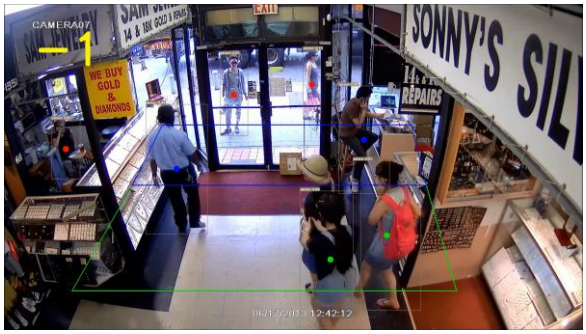
From **Table 3** our results, we can see that YOLOv7 was able to detect people correctly with pretty good accuracy. Unlike the project from [5]. YOLOv7 did not require us to label our own training set. Using a simple tracking approach, we obtained a MOTA score of 94.3% compared to the 80% score that [5] achieved. It is to note that our setting was less challenging to detect people correctly. Among the MOT15 challenge dataset, it is the one with the lowest density. It consists of 8 people walking in and out of the video, walking in unusual patterns with limited occlusion Fig. 4.**Error! Reference source not found.**

After we found the best combination of IoU and Confidence threshold, we tested different distance thresholds to see how it would impact MOTA and MOTP. We tested four different values, 0,100,250,500. Both metrics stayed identical across each distance threshold used. With the best parameter found, the detection of the bounding boxes is stable enough that the distance threshold does not impact the score. It is to note that the distance threshold could still be used as a safeguard for scenarios where the detection of boxes moves more in environments that are harder to detect.

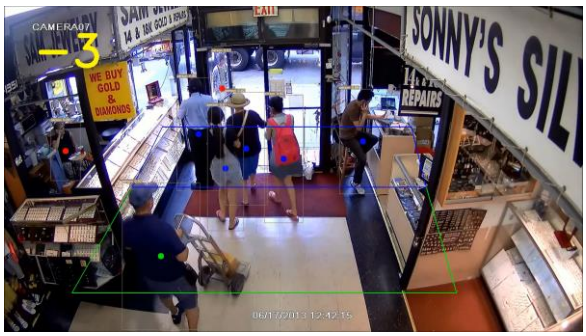
B. Testing

Once we found the best results, we used those hyperparameters on our test video. The test video uses a CCTV camera pointing downward at an entrance of a store [11]. Originally the region's shared border was at the entrance door. Visually, the results were not great. It missed quite a few people going in. We observed that the door caused a lot of occlusion instances and decreased the accuracy of our model. We determined that the counting accuracy of our model is subject to occlusion instances. Hence, we moved both regions to one side of the entrance door, and it removed some occlusion instances.

If an occlusion instance occurs, that situation falls in case scenario four followed by three discussed in the Multiple Object Tracking and Counting Algorithm. This can cause the detected person to be considered a new person, and the count does not get updated accordingly. This can be seen in Fig. 5. Test Results (a) and (b). One person was occluded, and the counter only counted two of the three people.



(a) Video Count = -1



(b) Video Count = -3 (Fail)

(c) Video Counter = -6

(d) Video Counter = -5

Fig. 5. Test Results

C. *Improvements*

From Table 3 we can observe that the number of False-Positive dominates the number of False-Negative and IDs Switch. This may indicate that the IoU and Confidence level could be increased to have a better balance between False-Positive, False-Negative, and IDs which could achieve a better score. The effect of a higher IoU or Confidence threshold can be investigated in future work.

We also mentioned that occlusion decreased our counting accuracy. The following approach could be considered. Instead of removing the detected points right away when someone leaves the region of interest, we could keep the point for a set number of frames before removing them. This will allow the ID to rematch if the person gets back in view.

Finally, as a next step, training could be performed on different dataset. In which different setting are tested such as density of people in the video, the luminosity of the video, or the angle at which it is taken. Doing so might result in the train model to perform better to unseen data.

IV. Conclusion

In conclusion, this report presents the algorithm that tracks multiple people and counts people entering and exiting a specific area. We utilized YOLOv7 as a state-of-the-art real-time object detector to detect people in real time. We then tracked people using the bounding box information from YOLOv7. By assigning the same ID to the person with the smallest centroid movement Euclidean distance of the bounding box between the previous frame and the current frame, we were able to track the person appropriately. Next, we set up two area to count people based on movement between these area. To accomplish higher accuracy, we tuned three parameters according to MOTP and MOTA. The parameters that gave the highest scores was a confidence threshold of 0.45 and a IoU threshold of 0.25. The distance threshold did not influence the Our model achieved MOTA score of 94.3%, and MOTP score of 39.1%. This result is more accurate than the results of MOTA 80% performed by Velastin et al [3].

V. Bibliography

[1] Wang1, C.-Y. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object.

[2] Mezei, S. &. (2010). A computer vision approach to object tracking and counting. *Studia Universitatis Babes-Bolyai, Informatica*.

[3] Velastin, S. A. (2020). Detecting, tracking and counting people getting on/off a metropolitan train using a standard video camera.

[4] Zhang, Y. W. (2021). Fairmot: On the fairness of detection and re-identification in multiple object tracking.

[5] Zhang, Y. S. (2022). Bytetrack: Multi-object tracking by associating every detection box.

[6] Weng, X. W. (2020). Ab3dmot: A baseline for 3d multi-object tracking and new evaluation metrics.

[7] Milan, A. L.-T. (2016). A benchmark for multi-object tracking.

[8] *Multiple Object Tracking Benchmark*. (n.d.). Retrieved from https://motchallenge.net/

[9] S A Sanchez, H. J. (2019). A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework .

[10] *Multiple Object Tracking Benchmark*. (n.d.). Retrieved from PETS09-S2L1: https://motchallenge.net/vis/PETS09-S2L1

[11] *HD SDI CCTV 1080P Security Camera Front Door Entrance Demo*. (2013). Retrieved from https://youtu.be/-IvBKBx0UBo