Human Counter

A. Introduction

The COVID 19 pandemic has resulted in drastic changes to our day to day lives. Public safety has become an important task for many local government and health officials who keep track of people's activity. Understanding an area's max capacity and how the space is utilized is an important part in ensuring public safety standards are held to a high standard. It can also allow institutions to keep track of the current occupancy of their spaces. Such institutions include gyms, public libraries, sporting events, restaurants, and is even utilized in O'neill Library. This serves as the main motivation behind our project.

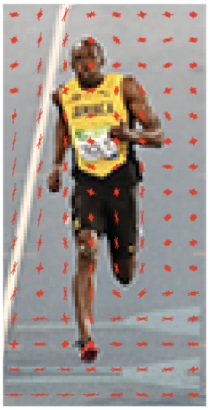
We aimed to design and train a model that can observe a space and keep track of its capacity as well as how the space is utilized. This project is significant because of its potential use in our current day to day activities. It has broad applications beyond a simple human count since it can be extended to monitor human patterns as well. For example, taking the instance of a library, by detecting and tracking people in the area, we could outline "hotspots" of human activity (i.e. places of high contact). We could observe what tables are most used by students and thus need to be cleaned more frequently. We could observe what times in the day are students visiting more often and thus requiring the extra staff. This project could even extend to law enforcement. In a given suspected area, we could potentially detect, count, and track humans and see which house is visited in an abnormal frequency indicating potential drug activity. This could also work in reverse. In a given area, we could observe which house is visited less frequently and would thus hide a potential fugitive (assuming one exists in this case).

The pandemic is not over, and we are currently seeing rises in numbers with incoming variants. Many state law officials are again enforcing new rules and mandates. Capacity limits are being set, and our project can help relieve some of the stress surrounding managing new rules and regulations. Our project aims to detect, count, and track people in a given area to better understand how the space is utilized and how capacity changes over time.

B. Related Work

B.1. HOG with OpenCV

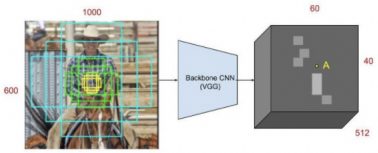
This project uses the Histograms of Oriented Gradients (HOG) algorithm. HOG is a 'feature descriptor', which takes an image and simplifies it by assigning 'features' to important information within the image, and gets rid of unnecessary information. The process begins by calculating the gradient, and from that getting the magnitude and direction. A Histogram of gradients is calculated in an 8x8 cell, and then normalized into a block of 16x16 cells. This all builds a Histogram of Oriented Gradients feature vector (image below).



This concentrates more on the borders and the shape of the images to extract its features individually from the background. Overall, it's less efficient compared to other approaches such as Faster RCNN and takes longer to train.

B.2. Tensorflow and Faster RCNN

This project uses transfer learning techniques to detect people from a surveillance camera. They used Tensorflow's Object Detection API - which provides pre-trained models with 90 detection classes, the class that was used for this project being 'person'. The architecture was modeled after the Faster R-CNN algorithm.There are four main steps, beginning with a Region Proposal Network (RPN) to generate bounding boxes. A single point is chosen as an 'anchor' (image below), of which many different shaped and sized bounding boxes are created. The next stage is feature generation, which is where object features are detected from those bounding boxes. Those features are used to classify the object and then in the final stage the boxes are made more precise. Although the model is efficient and quick,



when paired with the pre-trained models, it failed to reach expectations. The issue is within using pre-trained models and it is suggested to go a different direction.

C. Method

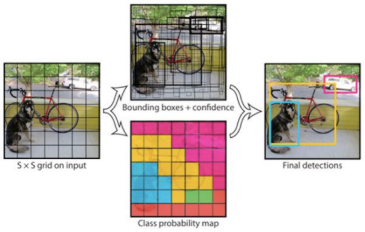
C.1. HOG + Linear SVM

We worked on the Histogram of Oriented Gradient (HOG) feature descriptor based linear SVM to create the human counter. For this we initially worked on images, and then implemented the algorithm for video. We had two different approaches for the video. One which was faster and less accurate and another which was more accurate, but computationally more expensive. The driving force behind the difference between these two are the hyperparameters that we adjusted for more accuracy. The main hyperparameters that we played with include winStride, padding, and scale within the detectMultiScale() function from the OpenCV people detector library. (Code in github).

Just some notes on SVM: There are seven stages to this model and implementation of the algorithm. It begins with extraction of the HOG features from the data set using samples. You train the positive and negative samples to create the model. These trained models are then used to generate detectors, which are used to identify the 'false positive' tests, or where the model detected a human that was not actually there. You again extract the HOG features from these samples, further training the model. THe object is then identified and the detection area, or bounding box, is optimized.

C.2. YOLO

YOLO - standing for You Only Look Once, is a more efficient and faster algorithm used for object detection. Unlike HOG, which repurposes classifiers to detect things, YOLO makes use of an end-to-end neural network that predicts bounding boxes and class all at the same time. It essentially divides an image into N number of S x S grids. Each cell of the grid predicts bounding boxes and confidence score. This method does require more computational power, and has issues with detecting smaller objects compared to other methods.



C.3. Moving Object Detection using Frame Differencing

We also implemented a moving object detection algorithm using frame differencing with OpenCV. This surprisingly yielded very good results. As the name suggests, we can detect any moving object in a video using this technique. Basically, the main motive here is to do motion detection in videos. Moving object detection has a range of use cases ranging from surveillance to security. So, finding a technique that can be easily used in low computation devices is crucial.
Advantages
-The first one is low computation power. As we do not use any neural network or deep learning technique, it is not computationally demanding.
-We can even run it on a CPU. Even a moderately powerful CPU will suffice for employing this detection technique for moving objects.
Disadvantages:
-First of all, we can only detect moving objects. If our goal is that, then it is all fine. But we will not be able to detect static objects using this technique. This also means that we cannot use this technique on images but only on videos.
-It cannot be actually completely real-time as we have to wait at least for a certain number of frames to get the background model. We also have to get a certain number of frames for differencing and summing and then only we can start detection.
-Using this with static cameras works pretty well. But with moving cameras it will not work at all as the objects will be just everywhere. Therefore, it is best suited for surveillance tasks where the camera is stationary.
-The background model and the moving objects should be pretty distinguishable. The lighting should also be good. Else, we may face issues like double detections for a single object, like in video 2.
-And objects close to each other will be detected as a single object, which is a big issue.

D. Experiments

We tested our various different algorithms on both images and video (highlighted in our github). We found that our initial results from just running the HOG algorithm did an adequate job on images, but really struggled with people at further distances and with groups of people. It also struggled with video. From there we saw great improvement from pairing this with linear SVM based model. Our use of hyper parameters further yielded better results. Next we implemented the Moving Object Detection using Frame Differencing, which yielded better results than both of these. Lastly, we explored YOLO, a deep-learning based model, which gave us some issues unfortunately.

E. Conclusions

Through our initial results we found that our original model making use of HOG was having accuracy issues and yielding very poor results most of the time. Bounding boxes were being created around non-human objects, and missing instances where there were people. When used on video surveillance we found that when there weren't too many people on screen it was able to accurately detect and track people walking in the foreground, but when two people were close side by side, or a large group was walking together, bounding boxes fidgeted and could not accurately detect the number of people. To address these issues as we mentioned we played the hyperparameters, which yielded better results, but at a cost of speed. The moving object detection using frame differencing and summing technique proved to also be quite effective.

F. Contributions

Contributions:
Abrar Jalal - HOG + linear SVM, moving object detection algorithm using frame differencing
Matt Abbene - YOLO
Sean Hurley - HOG, Introduction, Related Works, Method, Conclusion
Github link - https://github.com/mattabb/BC-CSCI-3343

G. References

https://github.com/intel-iot-devkit/people-counter-opencv
https://github.com/darpan-jain/crowd-counting-using-tensorflow
https://pjreddie.com/darknet/yolo/
https://debuggercafe.com/moving-object-detection-using-frame-differencing-with-opencv/
https://debuggercafe.com/object-detection-using-pytorch-yolov3/
https://debuggercafe.com/opencv-hog-hyperparameter-tuning-for-accurate-and-fast-person-detection/