CSC420 Project Report: human detection

Zizhuang Fan

December 2018

1 Introduction

The project I chose is detecting human in a video and possibly classifying actions and to determine a threat level. The idea of this project is to assist humans in quick responsive decision making in many real-world application. Some of the possible applications include pedestrian traffic awareness and autonomous driving, law enforcement and surveillance alarms, or for my naive purpose, to create an undetectable aimbot for any FPS games without accessing in-game memory.

In my project, I used the HOG (Dalal and Triggs, 2005) descriptors and trained them in a neural network, and used the sliding box technique for detection. My hypothesis is that classifying actions in upright human figures is going to be difficult since that actions are very similar in nature in a monocular image.

2 Data

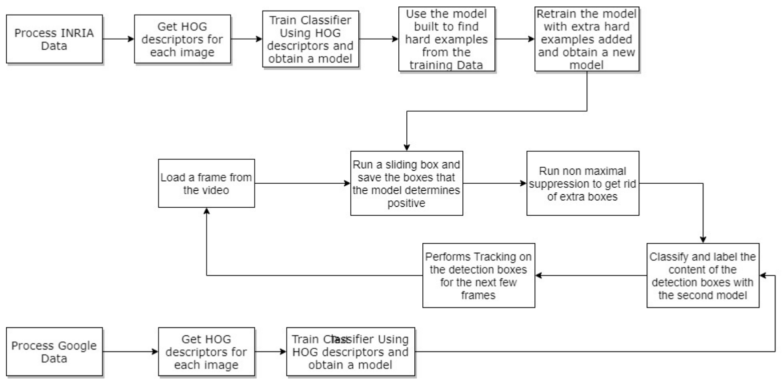
The testing and training data I used for human detection is from INRIA Person Detection dataset. This dataset contains 2416 positive training images and 1126 for testing. 1218 training negative images and 453 for testing.

The testing and training date I used for action classification is hand cropped and labeled from Google images. This data is very small, only more than a hundred for each class.

The demo video used is the Oxford Town center video used for the CVPR 2011 and the BMVC 2009 papers.

3 Methods

3.1 Process Overview


Process INRIA Data
Get HOG descriptors for each image
Train Classifier Using HOG descriptors and obtain a model
Use the model built to find hard examples from the training Data
Retrain the model with extra hard examples added and obtain a new model
Load a frame from the video
Run a sliding box and save the boxes that the model determines positive
Run non maximal suppression to get rid of extra boxes
Performs Tracking on the detection boxes for the next few frames
Classify and label the content of the detection boxes with the second model
Process Google Data
Get HOG descriptors for each image
Train Classifier Using HOG descriptors and obtain a model

The first row of boxes refers to humandetectionModel.py
The second and third row of boxes refers to extractPeopleTracker.py
The third row of boxes refers to poseDetectModel.py

3.2 HOG descriptor

The Hog descriptor first calculates the gradients of the image on both the x and y-direction using a 1d Sobel filter. Then the magnitude and direction of the gradients are calculated from the resulting g_x and g_y. Grouping Pixels in 8x8 cells, the magnitude and directions are used to form a histogram with 9 boxes, each representing the magnitude of gradients for a 20-degree angle interval, for each cell. Cells are normalized in blocks of 2x2 cells to reduce lighting variations, and for each block, a 4x9 descriptor is formed. There are total of 7x15 possible blocks so the total dimension of the hog descriptor for the scope of this project is 4x9x7x15 = 3780. The HOG descriptor used was from OpenCV's implementation since my own implementation would be way too slow without learning CUDA code.

3.3 Human Detection Model

Refer Sections to humanDetectModel.ipynb
The negative images are from the INRIA dataset are uncropped full-size images. For each of the negative images, 10 random windows of 64 x 128 are extracted for both testing and training negative images.(Section 1 and 2)

Once the images are processed, the hog descriptors are computed for each image (Section 3). Then they are put into a neural network for training. The network I used consists of a 128 dimension relu layer and a 2 dimension softmax layer. The loss function used is sparse_categorical_crossentropy. (Section 4)

The training negative images are examined by the resulting model and false positives are marked. I traced back to original INRIA image that the false positives sample are from, and the original image is searched exhaustively using a Gaussian pyramid to sample more false positive hard examples of window size 64x128. (Section 5)

The hard examples are collected and added to the training images, and the model is retrained and saved (Section 6 and 7)

3.4 Pose Detection Model

Refer Sections to PoseDetectModel.ipynb
In this part, a very small data set was hand cropped from Google images to attempt to classify between standing, walking, and running position. The hog descriptor was used and the model was similar to that of the humanDetectionModel.

Note: This section was supposed to be reserved for my partner. With very late notice(a week before the due date), she decided to late withdraw from the course, so this part needed to be significantly reduced. The original plan was to collect data on the angle of joints of different human actions and train a network to recognize the angles, then try to classify human actions from there, similar to the paper listed in the project proposal.

3.5 Video Human detection

Refer Sections to extractPeople.ipynb
video is loaded frame by frame. frames are resized for less computation, and a sliding box is done for detection every set interval. Positive boxes are saved and non maximum suppression is applied. Then the detection proposed boxes classified by the pose detection model and are tracked for the next frames in an interval. (Section 6)

3.5.1 Sliding box

For each detection frame, a box of fixed size 64x128 slides through the frame from left to right, top to bottom. Each time the box slides, the coordinates of the box shift by 12 pixels and the content of the box is evaluated by the Human detection model. Boxes are saved if it is marked positive. Once the box reaches the bottom right corner, the frame is scaled down within a Gaussian pyramid, and the box slides from the beginning again, saving positive boxes. With each iteration, bigger objects in the frame are detected. (Section 3)

3.5.2 Non maximum Suppression

From all the proposed boxes, in each iteration, the proposed box with the highest score is selected, removed and saved, and all boxes remaining that has an overlap: $\frac{area(box \cap box_0)}{area(box \cup box_0)} > 0.45$ is removed. Iteration continues until all boxes are removed, and saved boxes are left to be drawn onto the video. (Section 2)

4 Results and Discussion

4.1 Human Detection Model

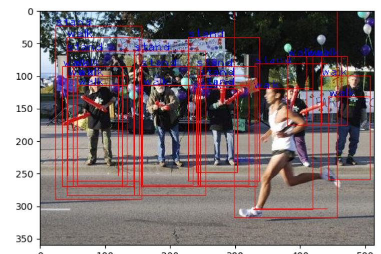
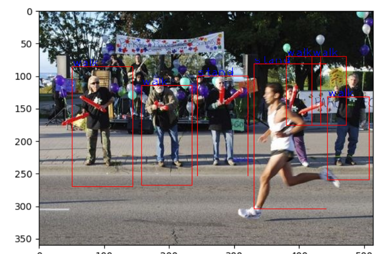
The initial validation accuracy was already quite high up to 98.48%. The Second step that added more hard examples did not make a significance in increasing the validation accuracy. Since the method selects random images from the negative images, I reran the whole training process many times and see an average of <0.03% increase in validation accuracy, and signs of over fitting in some iterations.

4.2 Pose Detection Model

Due to the significant reduction on this part, and very small data size, the validation accuracy was only 43-48%. I am quite satisfied with that since it is higher than a random guess of 33%. Also, the data collected on this part was through a Google image search such as "walking pedestrians". This may produce some very iconic images that are biased, and causes over-fitting to stereotypical postures.

4.3 sliding box and non maximal suppression

extractPeople.py (Section 4 on jupyter) runs a sample image

On the left is a sample image. In the middle is the sample detection when no non maximum suppression is used. On the right is the sample detection when non maximum suppression is applied.

To no surprise, there are still false positives such as the two boxes behind the man on the right. Also, some of the highest scoring boxes are not exactly the best box to fit the human, and the people hiding behind some balloons at the back are not detected.

lowering the detection threshold will allow me to obtain more number of correct detections, but also increase the number of false positives. Having more layers of the image pyramid will allow me to detect more variance in sizes of humans and reducing the shifting of the sliding box could allow more precision, but both will significantly increase run time.

4.4 Detection VS Tracking

The example for this section is shown in the Video presentation
The detection method detects every 5 frames, while the tracking method detects every 25 frames then tracks the detections for the next 24 frames.
The runtime for detection method and tracking method for the first 250 frames when there are not many people are 199, 53 respectively. The runtime for detection method and tracking method for the first 650 frames when more people are present after 250 frames are 509, 193 respectively.

Tracking time increase when there is more instances of people, and detection is less responsive, time using only detection remains rather consistent and responsive. However, detection method takes very long and tracking produces a lot smoother visualization in this case.

5 Main challenges

One challenge for this project was tuning the parameters described in 4.3. Is difficult to find a good threshold that on average, gives the best recall and precision on a different set of images with various sizes and various distribution of the position of the people while maintaining a manageable runtime for a typical student machine.

Another challenge is that the lack of experiences with libraries. A lot of time was spent to figure out what OpenCV libraries are useful. There are lots of tutorials that take a long time to learn, and some of them are not very helpful. Also, coding following intuition from class concepts requires massive parallel processing, the CPU code that I'm experienced to implement is not very runable to application. Learning GPU optimizing code seems like another course on its own.

6 Conclusion and future Work

In this project, I trained and retrained a neural network to detect a human in upright positions. Implemented sliding box and non maximum suppression detection techniques.As well as experiencing detection with tracking in a video setting. The action classification was not implemented as I hoped since my partner left, so that would be done after the due date of this assignment. In addition to that, finding options to detect more than an upright human would be the next step. Learning GPU code, and running my sliding box implementation in parallel threads can make this project very feasible to an aimbot for games in real time since variation on human figures is much less in games than the real world.

7 References

Dalal, N., Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05). doi:10.1109/cvpr.2005.177

Onishi, K., Takiguchi, T., Ariki, Y. (2009). 3D Human Posture Estimation Using HOG Features of Monocular Images. INTECH Open Access Publisher. doi:10.1109/ICPR.2008.4761608