Individual lab report #5

Nov, 22, 2016

Xiaoyang Liu

Team F Rescue Rangers

Juncheng Zhang (Henry)

Karthik Ramachandran

Sumit Saxena

Individual Progress

- Implement human detection algorithm based on HOG and SVM in Visual Studio 2015 with OpenCV 3.1.0, and developed a method to obtain human locations in pedestrian images.
- Used background subtraction algorithm to detect moving objects in a video clip in VS2015 with OpenCV 3.1.0.
- Tried a few methods to create suitable training set for the classification in MATLAB 2016a.

Implement human detection algorithm in VS2015 with OpenCV 3.1.0

1) **Transfer codes of HOG and SVM algorithm from MATLAB 2016a to VS2015 with OpenCV3.0.1**

Since the last progress review, I have implemented most of the human detection algorithms based on HOG and SVM in MATLAB 2016a. The reason why we decided to transfer codes from MATLAB to Visual Studio is that the codes written in the latter one is much easier to be extended and integrated with other modules. I have done the following things to make it happen

- Installing OpenCV and Configuring it with VS2015
- Transfer codes format from MATLAB to C++
- Utilize relative OpenCV classes and structs under cv and cv::ml namespace

In terms of implementing HOG and SVM algorithms in VS2015, I found two classes that are relevant in OpenCV 3.1.0.

- cv::HOGDescriptor (Struct)
- cv::ml::SVM (Class)

In addition, I found the classifier(getDefaultPeopleDetector) in cv::HOGDescriptor struct, which is mentioned in [1]. The following tests are all based this trained classifier provided by the author of [1].

2) **Developed a method to obtain human locations in pedestrian images**

In terms of our project, the final goal after recognizing humans in frames is to generate positions of the humans. As I mentioned in the previous individual laboratory report, we have already been able to detect humans in each pictures by using the algorithm mentioned in [1]. So, our next step is to develop a method to obtain human locations in the pedestrian testing images.

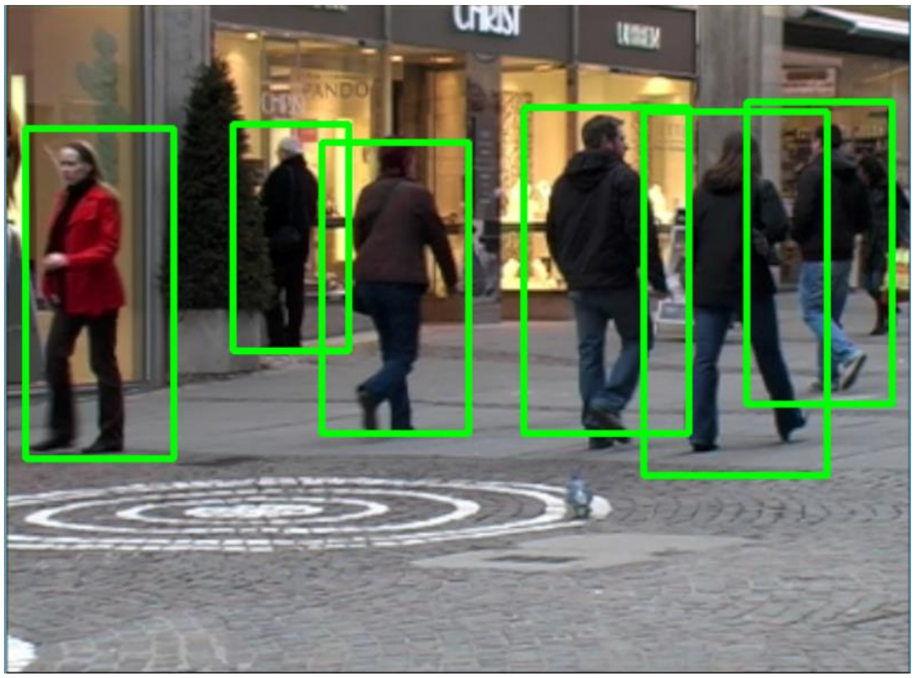
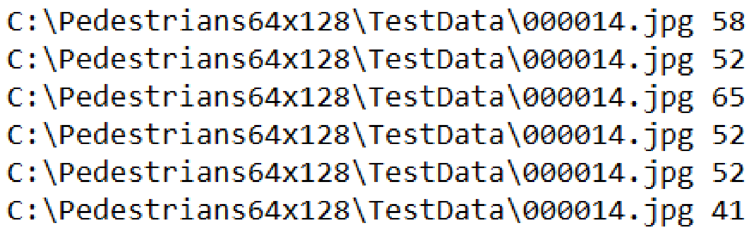


Figure.1 Generating a multiscale rectangle for every human candidate



```
C:\Pedestrians64x128\TestData\000014.jpg 58
C:\Pedestrians64x128\TestData\000014.jpg 52
C:\Pedestrians64x128\TestData\000014.jpg 65
C:\Pedestrians64x128\TestData\000014.jpg 52
C:\Pedestrians64x128\TestData\000014.jpg 52
C:\Pedestrians64x128\TestData\000014.jpg 41
```

Figure.2 The central position of each rectangle (the last column)

Figure.1 and figure.2 illustrate the general idea of locating humans in one input frame. First of all, I generate a multiscale rectangle for every human candidate detected in the picture by using detectMultiScale function in Struct HOGDescriptor. It starts with scanning the image with a small window whose size is set by window stride, and processes it with the trained classifier and eventually labels those who contain humans. The final shape of those green rectangles in figure.1 is determined by two things. The first one is the integration of all the small windows with humans, and the second one is dominant rectangles which turn out to be larger than those who intersects with them. Second, we can obtain the top-left and bottom-right index of those rectangles, and the index of central location will be the average of these two variants.

Create negative training set for the classification

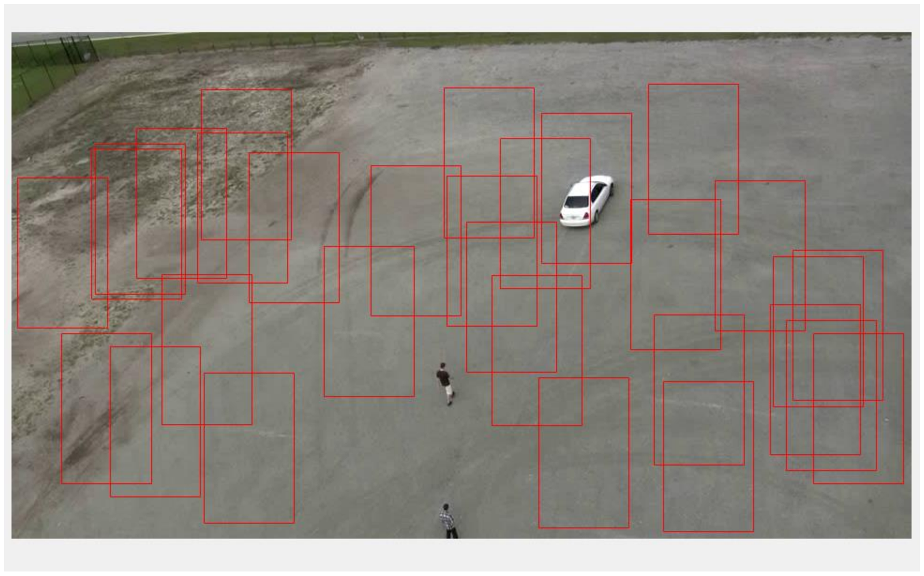


Figure 3. Capture background images randomly from a given picture

Figure 4. Cropped background image in a randomly chosen rectangle

As I mentioned in the previous ILR report, we have figured out a way to create our own positive training set, which is consist of human images taken from drones. Therefore, this time I continue creating our own dataset by generating a negative training set with the help of MATLAB.

The idea of implementing this is very easy. The first thing to do is to randomly choose points within the width and height of the given image. Next, expand the point into a designated-size rectangle, which must be the same as that of positive training images. Finally, eliminate all the captured images which contains humans that might have bad influences on training the classifier.

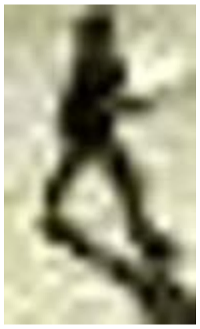


Figure 5. Person in a cropped aerial image

By combining the positive and the negative aerial dataset together, I've created a reasonable-size training set with 900 negative images and 308 positive images. Although there are still many things we can do to improve the representativeness of this training set, we have already got an accuracy of 61.25% when examine our algorithm on a test set with 5 negative test images and 16 positive images.

Table 1 Confusion Matrix

Train| Test	**Positive**	**Negative**
Positive	4	2
Negative	6	9

Used background subtraction algorithm to detect moving objects in a video clip in VS2015 with OpenCV 3.1.0

Since we have already figured out ways of getting enough suitable training set, the next big issue for our CV job is to find a way to capture test images from a video clip or a set of images. It is very significant because we cannot detect accurately if humans are too small to be seen in a very large picture. Therefore, we narrow down our detecting range. Basically, we can first find the moving object in the scene which might be a possible human candidate, and then crop the moving object sending it to the trained classifier. Here I implement a background subtraction method in order to achieve the goal.

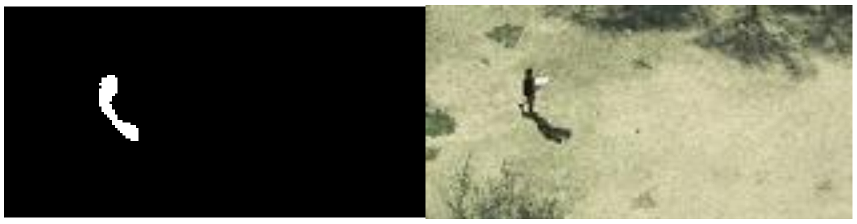


Figure 6. Segmentation mask(left) and the raw input video frame(right)

Above is the result of segmenting moving object (the person who is carrying package) and static background from a given video clip. The algorithm I used to implement the background subtraction strategy is called ViBe[2], a widely used background subtraction method in computer vision field.

Challenge

- Finding which is the most representative training set is a tough task. At first, after generating positive training set, I randomly choose some other aerial images without humans to be the negative training set. However, the accuracy turned out to be very low. Later I realized that those negative training set I select covers too many other irrelevant features, which might largely disturb the training process of the classifier. That's the main reason I decided to create our own negative training set as well.
- In terms of background subtraction, ViBe[2]is not a perfect choice in our case because it will segment human shadows as well as the body. However, we are not expecting any shadows in our test set for the shadows will affect the classification process if there are no train images that have the "shadow features". Besides, ViBe[2] is quite sensitive to vibration of the camera which somehow can make the algorithm segment the background into foreground. But in real aerial video case, we cannot ensure that the videos will have a static background.

Team Work

In this week, Juncheng(Henry) devoted most of his time implementing another way which is based on blob and edge detection to find human candidates. The result turns out that this method can detect most of humans in one frame, so we might use it as a final solution to capture potential human candidates and recognize the cropped images by using SVM and HOG algorithm. Karthik worked on changes to the waypoint navigation app for giving providing precise GPS location as input (in addition to map) and changes to detect altitude and location of the drone using RGB camera and image by exploring AprilTags. Sumit mainly focused on the drop mechanism.

Future Work

The next checkpoint is FVE. So, in the coming week it's very urgent for us to integrate all the things together and test on large enough aerial dataset to verify the efficiency of our algorithm.

Since we have got a camera from our sponsor, we can collect data used for the FVE. So in the coming week, me and Juncheng will run the integrated code on a self-collected dataset, and modify our algorithms in order to detect humans in at least 60% of the given images.

Reference

[1] Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, pages 886 – 893 IEEE, 2005

[2] Barnich, Olivier; Van Droogenbroeck, Marc (2009). "ViBe: A powerful random technique to estimate the background in video sequences": 945–948. doi:10.1109/ICASSP.2009.4959741