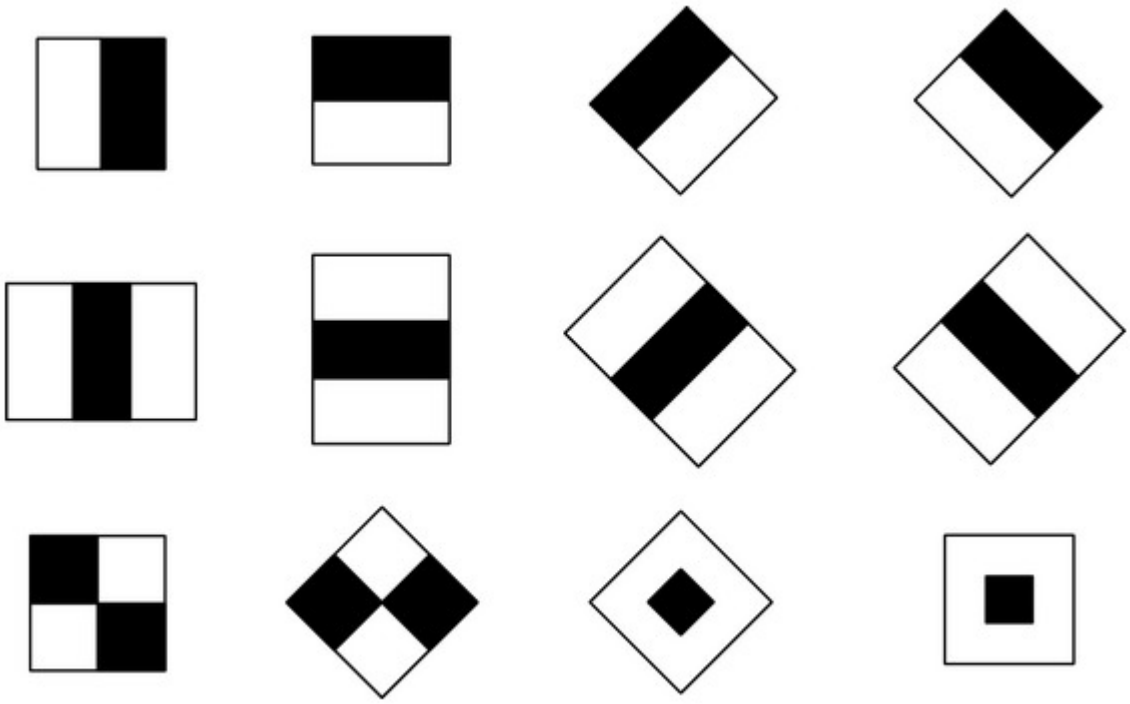
Xiangyi Cheng (xxc283)

Introduction and Background

Intubation is one of the most common procedures performed worldwide in emergency departments which requires high skill. It's performed by inserting a plastic tube into the trachea to maintain an open airway of a patient. This intubation procedure often involves complications due to invisible glottis where is the tube must be inserted. Therefore, device that can lead automated intubation is needed urgently to bring breakthrough in many emergency situations. In my research, IntuBot, a prototype of robotic intubation device, was developed to fill in this need. Currently, the hardware of IntuBot was fully assembled while the algorithm is still at the starting point. As an automated intubation device, IntuBot should be able to accurately detect the vocal cords which are the entrance of the airway. Plastic tube will be pushed into the patients' airway by motors based on this detection.

To detect the vocal cords with high accuracy, a large and reliable database contained vocal cords will be needed. Although there exists amount of vocal cords images online, self-built database is still needed since at least thousands of images are missing. The self-built database will be established by modifying then adding to the existing images. After obtaining the database, object detection using Haar feature-based cascade classifier in OpenCV will be applied.

Cascading is a case of ensemble learning using all information collected from the output of a classifier as additional information for the next cassifier in the cascade. Cascading Classifier are trained by importing amount of positive images which contains the target objects in different views and negative images without that objects in the same size into the computer. After it is trained, the classifier can be used to detect that object from a region of an image. Haar feature-based cascade classifier was first come up by Viola, Paul and Michael Jones in 2001. A simple rectangular Haar-like feature can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image, as shown below. The obvious advantage of a haar-like feature over other features is the fast computation speed. Therefore, haar-like features are proper to be used in real-time detection.



After cascade classifier is trained, the classifier will be implemented to detect the vocal cords on a series of images for testing. The results will be evaluated.

Image Pre-processing and Data Preparation

Building a dataset with pre-processed images is critical before classifier training. There are 665 positive images recording vocal cords without pre-processing which were collected from internet. 4090 negative images were downloaded from http://www.image.net (http://www.image.net). The positive images need to be seperated into training images and testing images. Training images will be imported into system to train a classifier while other images are for testing if the detection is successful with the certain classifier.

At the very begining, develop the code below to randomly select images as either training images or testing images. The selected images will be stored into two different folders.

In [18]:
```
import os
import cv2
import random
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mping

def randomly_select_test_image():
    image_list=os.listdir('vc_training/raw_images/
                          Closed_vocal_cord_Expansion')
    #image_list=os.listdir('vc_training/raw_images/Open_vocal_cord')
    select_num=74
    select_image=random.sample(image_list,select_num)
    test_image_num=27
    if not os.path.exists('test'):
        os.makedirs('test')
    if not os.path.exists('test_resized'):
        os.makedirs('test_resized')
    for i in range (0,select_num):
        try:
            #pre_name= 'vc_training/raw_images/Open_vocal_cord/'+
            str(select_image[i])
            pre_name='vc_training/raw_images/
                          Closed_vocal_cord_Expansion/'+
            str(select_image[i])
            new_name='vc_training/test_pos_img/'+
                          str(test_image_num)+'.jpg'

            os.rename(pre_name,new_name)
            #raw_test_image=cv2.imread('test/'+
                          #str(test_image_num)+'.jpg')
            #resized_image=cv2.resize(raw_test_image,(50,50))
            #cv2.imwrite('test1/'+str(test_image_num)+
                          #'.jpg',raw_test_image)
            test_image_num+=1
        except Exception as e:
            print str(e)

randomly_select_test_image()
```

Besides classification, we have to crop the raw images into the images only with region of interest (ROI). And the size of no matter positive images or negative images has to remain the same as well as the color is better to be gray due to color chanel is useless in training with haar-like features.

The two functions below shows how to resize the positive and negative images into the same size and modify the color.

In [20]:
```
def convert_pos():
    pic_num=1
    for img in os.listdir('vc_training/raw_images/pos_target'):
        try:
            img=cv2.imread('vc_training/raw_images/
                           pos_target/'+str(img),
                           cv2.IMREAD_GRAYSCALE)
            resized_img=cv2.resize(img,(50,50))
            cv2.imwrite('vc_training/pos5050/'+str(pic_num)+'.jpg',
                        resized_img)
            pic_num+=1
        except Exception as e:
            print str(e)


def convert_neg():
    pic_num=1173
    for img in os.listdir('vc_training/neg'):
        try:
            img=cv2.imread('vc_training/neg/'+str(img),
                           cv2.IMREAD_GRAYSCALE)
            resized_img=cv2.resize(img,(50,50))
            cv2.imwrite('vc_training/neg/'+str(pic_num)+
                        '.jpg',resized_img)
            pic_num+=1
        except Exception as e:
            print str(e)

convert_pos()
convert_neg()
```

After the pre-processing, the images are ready to be used as training or testing. The images below shows the change of one positive image from original image to pre-proccesed image.

Classifier Training with Haar-like Features

To train a classifier, we have to tell the computer where the negative images and positive images are as well as where the target region are in the positive images. The file related to the location of negative images is called bg.txt. The file info.data records the location and the coordinates of the

ROI in the positive images. The resized images are in 50 ∗ 50 so that the ROI is full of each image.

In [22]:
```
def create_neg_description_files_modified():
    for img in os.listdir('vc_training/neg'):
        line= 'vc_training/neg/'+img+'\n'
        open('vc_training/bg.txt','a').write(line)

def create_pos_description_files():
    for img in os.listdir('vc_training/pos5050'):
        line = img + ' 1 0 0 50 50\n'
        #print line
        open('vc_training/info.dat','a').write(line)



create_neg_description_files_modified()
create_pos_description_files()
```

A part of the content of the opened bg.txt which writes the location of the negative images is shown as below.

```
vc_training/neg/697.jpg
vc_training/neg/64.jpg
vc_training/neg/454.jpg
vc_training/neg/709.jpg
vc_training/neg/644.jpg
vc_training/neg/25.jpg
vc_training/neg/498.jpg
vc_training/neg/789.jpg
vc_training/neg/813.jpg
vc_training/neg/506.jpg
vc_training/neg/1125.jpg
vc_training/neg/319.jpg
vc_training/neg/223.jpg
vc_training/neg/841.jpg
vc_training/neg/685.jpg
vc_training/neg/1102.jpg
vc_training/neg/139.jpg
vc_training/neg/355.jpg
vc_training/neg/1071.jpg
vc_training/neg/681.jpg
vc_training/neg/417.jpg
```

A part of the content of the opened info.dat recording the location and the target region in the positive images is shown as below.

```
9.jpg 1 0 0 50 50
88.jpg 1 0 0 50 50
360.jpg 1 0 0 50 50
6.jpg 1 0 0 50 50
387.jpg 1 0 0 50 50
122.jpg 1 0 0 50 50
409.jpg 1 0 0 50 50
219.jpg 1 0 0 50 50
464.jpg 1 0 0 50 50
476.jpg 1 0 0 50 50
293.jpg 1 0 0 50 50
479.jpg 1 0 0 50 50
529.jpg 1 0 0 50 50
140.jpg 1 0 0 50 50
401.jpg 1 0 0 50 50
222.jpg 1 0 0 50 50
```

As mentioned above, we have 665 positive images but 4090 negative images. Abviously, the amount of the positive images are not enough which will cause low accuracy of the detection. Therefore, it is time to expand our dataset with the existing images. A function called opencv_createsamples in opencv is able to solve the problem by merging one negative image and one positive image into one new image with ROI. As the command shown below, pos/655.jpg points out which positive image would be merged. bg.txt tells where to find the negative images. info/info.lst means the folder which will store the new created images and the file recording the ROI location. The very last 200 reflects how many new images we want to create.

```
opencv_createsamples -img pos/655.jpg -bg bg.txt -info
info5050/info.lst -pngoutput info -maxxangle 0.5 -maxyangle -0.5 -
maxzangle 0.5 -num 200
```

The created images will be like the below image. We can use the command to create more images to be as our part of our dataset.

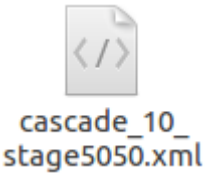
Create a vector file to hightlight all the positive images information.

```
opencv_createsamples -info info/info.lst -num 1000 -w 50 -h 50 -vec
positives.vec
```

After obtaining all the files, we are able to start the training process. The folder data recieves all the trained outcome files. The stage number is set to 10 so that we can have a relatively high accuracy file.

```
opencv_traincascade -data data -vec positives.vec -bg bg.txt -numPos
600 -numNeg 300 -numStages 10 -w 50 -h 50
```

The trained file at the 10 stage is



Detection Testing

Detection can be done with the trained .xml file. The code below shows how to detect one image with the trained file.

In [4]:
```
def detect_vocal_cord_from_image():
    vocal_cord_cascade=cv2.CascadeClassifier('vc_training/
                                             cascade_10_stage5050.xml')
    img_path= 'vc_training/test_pos_img/8.jpg'
    img=cv2.imread(img_path)
    gray_img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    vocal_cord=vocal_cord_cascade.detectMultiScale(gray_img,5,60)
    for (x,y,w,h) in vocal_cord:
        print 'X coordinate:', x
        print 'Y coordinate:', y
        print 'Width:', w
        print 'Height:',h

        # the image to draw, starting point,
        # ending point, color, line thickness.
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        font=cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(img,'vocal cord',(x,y-5),font,
                    0.4,(200,255,255),1,cv2.LINE_AA)

    cv2.imwrite('result/vocal_cord_detection.jpg',img)
    img1=mping.imread('result/vocal_cord_detection.jpg')
    plt.imshow(img1)
    plt.show()
    cv2.waitKey(0)
    #cv2.destroyAllWindows()

detect_vocal_cord_from_image()
```

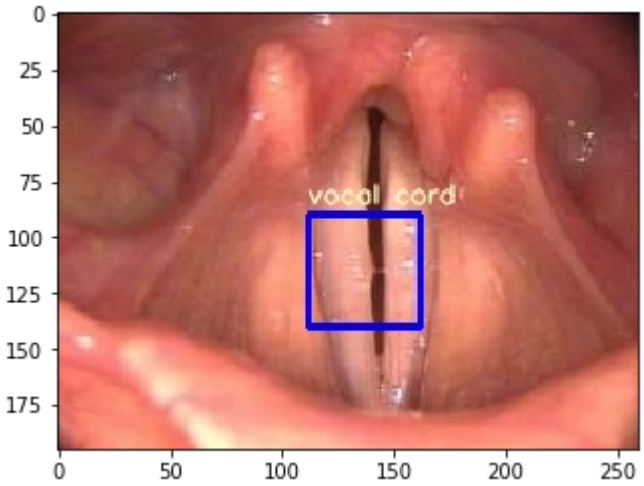
```
X coordinate: 92
Y coordinate: 89
Width: 50
Height: 50
```

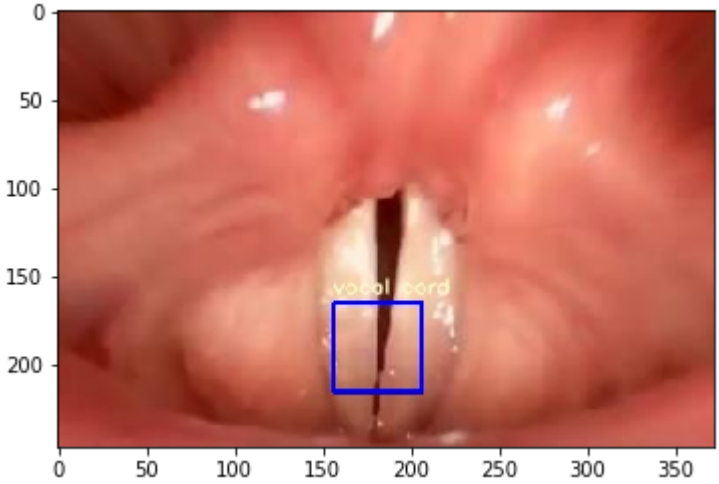
To detect more images, I also developed a code to detect the testing images into the certain folder.

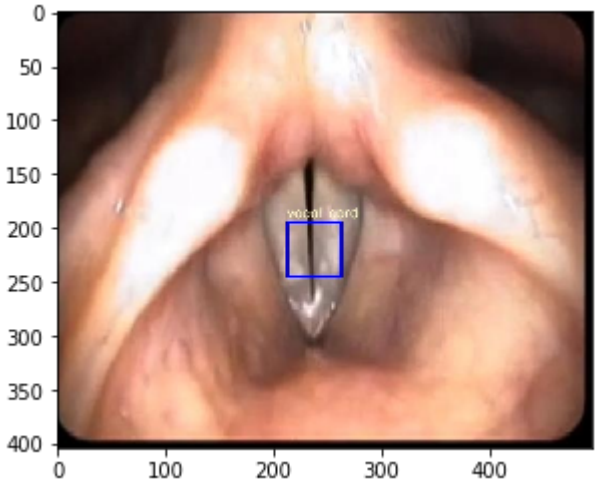
In [15]:
```
def detect_vocal_cord_from_folder():
    vocal_cord_cascade=cv2.CascadeClassifier('vc_training/
                                             cascade_10_stage5050.xml')

    scale_factor=10
    min_neighbors=60

    for img in os.listdir('examples1'):
        img1= cv2.imread('examples1/'+str(img))
        cv2.imshow('1',img1)
        gray_img=cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
        vocal_cord=vocal_cord_cascade.detectMultiScale(img1,
                                        scale_factor,min_neighbors)

        for (x,y,w,h) in vocal_cord:
            cv2.rectangle(img1,(x,y),(x+w,y+h),(255,0,0),2)

            font=cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(img1,'vocal cord',(x,y-5),font,
                        0.4,(200,255,255),1,cv2.LINE_AA)

        cv2.imwrite('result/'+img,img1)

detect_vocal_cord_from_folder()
```

Since it is hard to show the results of those 100 testing images in the notebook, I choose 6 of them to show the results here.

In [44]:
```
def show_images():
    for img_show in os.listdir('result'):
        img_show= mping.imread('result/'+str(img_show))
        #plt.figure(figsize=(10,4))
        #plt.subplot(1,6,i)
        plt.imshow(img_show)
        plt.show()
show_images()
```

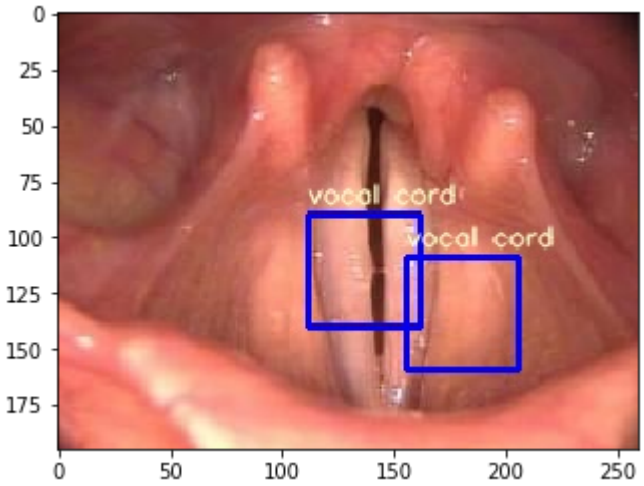


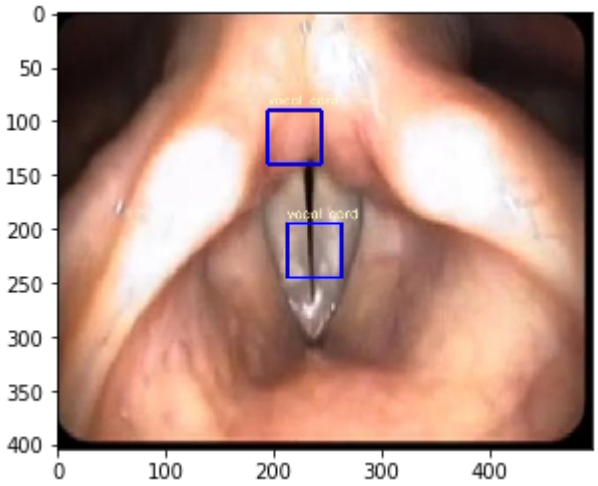




However, in some images, the wrong regions will be labeld as vocal cords as well.

In [45]:
```
def show_images():
    for img_show in os.listdir('result1'):
        img_show= mping.imread('result1/'+str(img_show))
        #plt.figure(figsize=(10,4))
        #plt.subplot(1,6,i)
        plt.imshow(img_show)
        plt.show()
show_images()
```

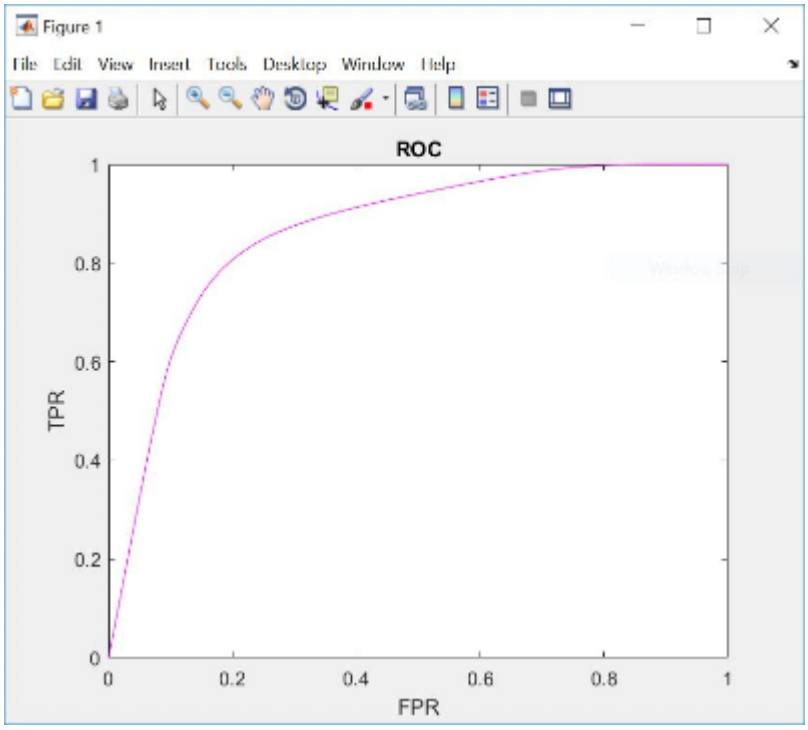




The accuracy of the detection will be analyzied in the latter part of the notebook.

ROC Curve and Accuracy

The ROC curve is drawn in MATLAB by plotting on the true positive rate (TPR) and the false positive rate (FPR) on the stage 10 at the various threshold settings, as shown below.



With the changing of the threshold, the TPR and FPR are changing as shown in the ROC curve. As we know, the ideal situation is that the point is at the left upper corner of the ROC curve so that TPR is 1 and FPR is 0 which means the accuracy is 100%. In reality, however, it is not practical to have the ideal situation. What we are looking for is the point near left upper corner. In this way, a proper threshould 60 is set and based on this threshold, we can see that the accuracy of the training is accepable that TPR is above 85% when FPR is at 22% or so. This data is obtained by testing the images. But if we detect the vocal cords in a video with a strict threshold, the performance will be better due to amount of frames.

Conclussion and Discussion

From this final project, vocal cords detection using haar-like features done in openCV is achieved. The accuracy is above 85%. Besides the outcome from 10-stage, I also tested the files trained from 6-stage, 8-stage and 12-stage on the testing images. And I realized that if the stage is low, the computer will regconize many regions even not real vocal cords as the target object. But in another direction, the situation that there is no region identified as the target even with the vocal cords will happen if the stage is set too high. Therefore, a proper stage is important to the detection. Moreover, the threshold should be chosen reasonably once the stage is set. A good threshold is able to bring us excellent accuracy and nice performance, as demenstrated in the ROC curve above.

The advantage of the cascade training is that the trained file can be imported into some other processors with low storage and computaional capacity. In this way, some devices without connecting to a powerful computer are also able to do the detection.

Reference

https://en.wikipedia.org/wiki/Cascading_classifiers (https://en.wikipedia.org/wiki/Cascading_classifiers)

Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

https://en.wikipedia.org/wiki/Receiver_operating_characteristic (https://en.wikipedia.org/wiki/Receiver_operating_characteristic)