

2018 Spring BU EC500J1 Project Report

Project Name: ISeeU

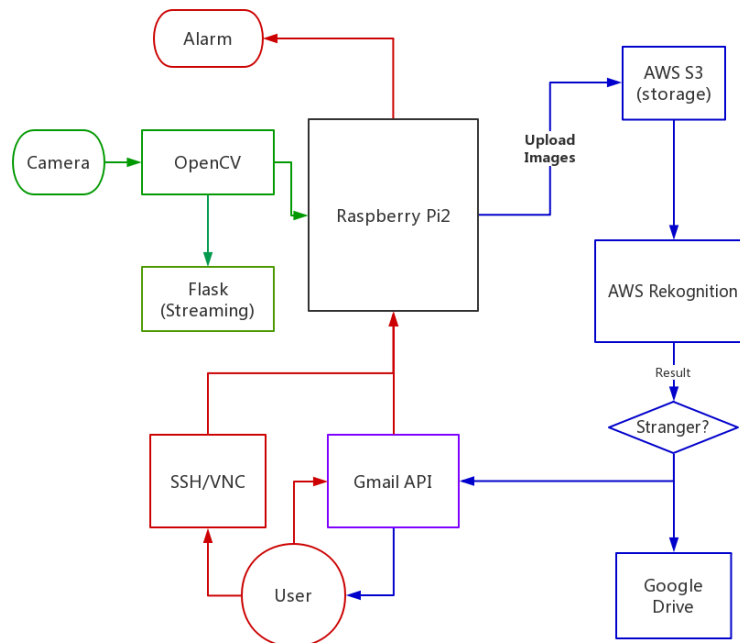
Team members: Sihan Wang, Ganquan Wen, Fengjun Li

Github Link: <https://github.com/shwang95/Intelligence-Surveillance-System>

Introduction

This is a front door security camera built on Raspberry Pi2, ran by python2.7. The camera can detect a face and follow it. It will also take a picture of the detected face and send it to AWS Rekognition to compare with the pictures of people already saved by the user. If the picture of the detected person does not match any one of the pre-saved pictures, it will send an alert with the picture to the user through Gmail. In this situation, users have 2 options, one is replying the email to set off the alert, the other one is picking up the phone, using SSH or VNC application to connect to Raspberry Pi and take charge of it as well.

High level diagram:



Face detection:

The program running on Raspberry Pi2 will detect the face from the camera. Specifically, the face recognition will be implemented with OpenCV, using haar cascades.

AWS Rekognition:

AWS Rekognition provides face comparison function in our project. Specifically, we will use boto in our project. Boto is the AWS SDK for python, and the version we will use is boto3. Boto allowed us to send two pictures to the AWS server and get a JSON response based on the Rekognition service.

Alert (through Gmail API):

It will be triggered if the result from AWS Rekognition does not match the requirement. It will send the picture took by the camera and a code according to now time (day/hour/min/sec) to the user through email. Also, it will upload the time to the Google drive sheet as log. This module requires Gmail API and Google Drive API.

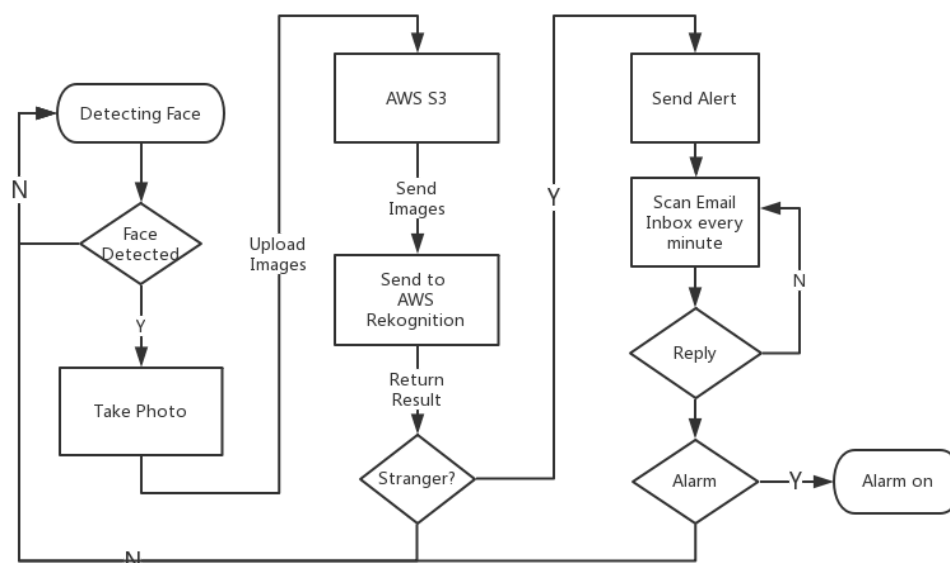
Remotely control and trigger Alarm:

This part is for user to remotely control Raspberry Pi and set off the alarm frighten the stranger. We implemented two methods to do this.

First method is to receive command through Gmail. After the alert email is sent, the program will keep checking the Gmail Inbox every minute in the next hour. The user can reply with the code attached in the email to set off the alarm. This part requires Gmail API.

Second method is using SSH or VNC application on the iphone to take over the Raspberry Pi. User can click screen and control everything by using their fingers.

Flow Diagram:



Failures

Gmail Push Notification:

At first, we thought that push notification is a better way to implement the command receiving function. Because the program keeps sending request to check inbox is not so efficient if the user does not reply the email in short time.

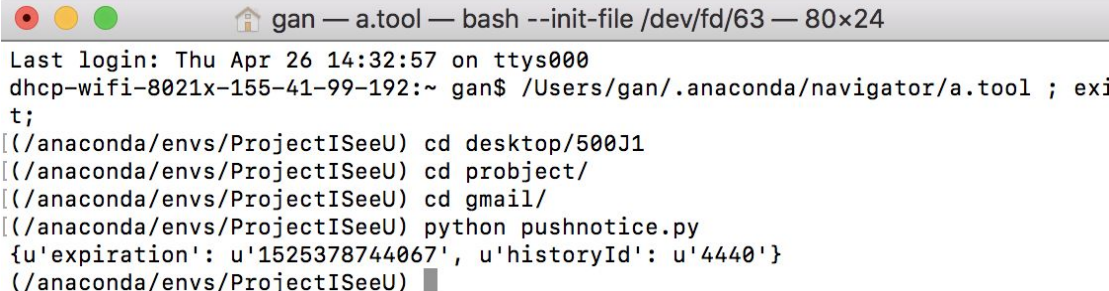
But, push notification is more intractable than sending and receiving email, so we did not finish it.

Besides Gmail API, it also requires Google Cloud SDK to implement this function. We managed to send watch requests. But failed to receive notification. It should be caused by failed to install the Gcloud SDK correctly. I could not use gcloud command to pull the notification. Also, the official Gmail API website does not provide an example python code to implement this function. There are not much I can find on other websites for us to figure out how this works.

If the watch() request is successful you will receive a response like:

```
{
  historyId: 1234567890
  expiration: 1431990098200
}
```

Here is the screenshot of sending the watch request successfully.

A screenshot of a terminal window. The title bar shows 'gan — a.tool — bash --init-file /dev/fd/63 — 80x24'. The terminal content shows the following commands and output:

```
Last login: Thu Apr 26 14:32:57 on ttys000
dhcp-wifi-8021x-155-41-99-192:~ gan$ /Users/gan/.anaconda/navigator/a.tool ; exit;
[/anaconda/envs/ProjectISeeU] cd desktop/500J1
[/anaconda/envs/ProjectISeeU] cd probject/
[/anaconda/envs/ProjectISeeU] cd gmail/
[/anaconda/envs/ProjectISeeU] python pushnotice.py
{u'expiration': u'1525378744067', u'historyId': u'4440'}
[/anaconda/envs/ProjectISeeU]
```

Even though this part is not implemented successfully, after finish the Gmail API and understood the push notification better, I think they are a little bit similar for python program. Because it also needs the program to pull the subscription periodic, just like getting threads from inbox. But pulling notification should spend less time than getting threads.

Gmail unstableness in receiving email:

It is a very weird situation. Sometimes, it will be about half an hour before the mailbox of ISeeU received the email. It usually happens if I tested this email function for several times in a short time like ten minutes. I am not sure what causes this. My best guess is that Gmail may limit their users to receive too many mails in a short time.

<input type="checkbox"/>	☆	me, Ganquan (2)	Alert from ISeeUI - 25185602 On Wed, Apr 25, 2018 at 6:56 PM, <wenganq11@gmail.com> wrote: Stranger show up at 2018-04-25 18:56:02 R	7:06 pm
<input type="checkbox"/>	☆	me, Ganquan (2)	Alert from ISeeUI - 25175555 On Wed, Apr 25, 2018 at 5:55 PM, <wenganq11@gmail.com> wrote: Stranger show up at 2018-04-25 17:55:55 R	6:18 pm
<input type="checkbox"/>	☆	Google Cloud Platform	Google Cloud Newsletter, April 2018 20+ security enhancements harden Google Cloud - News, research, and resources highlight how Google	4:52 pm
<input type="checkbox"/>	☆	Ganquan Wen (3)	alarm - 567281	Apr 24

On the top right, you can see the time for the email actual arrival is about 20 minutes late from it is sent (sent at 18:56, but received at 19:06).

Detection of if the camera be moved (the parts that did not be implemented):

We focused on the main functions of our camera, taking pictures and get recognition result. So we did not implement this part because it is not a part of the main functions, even though it is a cool and useful function. We spent most time on taking pictures, getting Rekogniton result, sending alert , getting commands and integrating them all together.

Using Amazon dash button to shutdown/Reboot Raspberry Pi

Sometimes users may hold a party , In this situation, there would be many unknown people come into the house. In order to shutdown the Raspberry Pi to prevent sending email and alarming. We planned to use Amazon dash button as a switch to shutdown Raspberry Pi.

To achieve this function, we need steps below: first we set up the button via amazon and find the dash button hardware address, then create a shutdown shell script include command `$sudo shutdown -h now` .Next we create a node script to run the shell script.Finally install supervisor to run the node script at system start. After these steps, the dash button can be used as a switch button.

This button can really make benefit to users to prevent disturbing while finally we abandoned this function. The reason is that we concern this dash button may be a backdoor for someone to shut the machine forcible. Another reason is that we found an alternative method which is SSH/VNC to realise this function.

Success:

Face Detection:

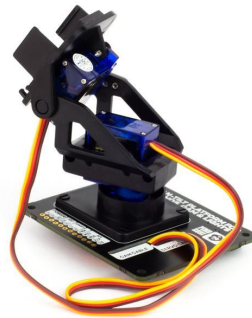
We use OpenCV for initial face detection. The program will process on every frame captured by the camera, and then push the image as jpeg formation to the flask application. For face detection, we use haar cascades. If there is a face detected, the program will save a temporary image to enable AWS Rekognition for face comparison.

AWS Rekognition:

We use AWS Recognition for face comparison, users provide source images i.e., faces that have no need to alert, and the client program will upload the temporary image captured to the AWS S3 storage, then the AWS Rekognition will then response a json file with the comparison result, based on the result, the program will determine whether to send a alert email or do nothing. After all steps, the program will delete the temporary image.

Camera:(ability to move around)

We use a two-servo pan-tilt hat manufactured by Pimoroni Ltd., and it provides a library for control the hat. We calculate the middle point of the face, i.e, the middle point of the diagonal of the face, and compare it with the middle point of the whole picture, then use the processed scale to calculate the exact angle for both pan and tilt servo to move.



The exact angle calculate method for pan and tilt is: $\text{angle_to_move} = (\text{mid_of_face(px)} - \text{mid_of_image(px)}) / \text{half_of_image(px)} * 5(^{\circ})$. And since the servo cannot move over 180° or 0° , there will be a limit on the angle.

Streaming

For streaming, we simply use Flask as our web frame and use a background program to push the processed frames continuously to the web page. The background program will generate a binary stringstream from the frame picture encoded in jpeg format and then send to Flask application, then the application will put the binary stringstream into a `` tag, with boundary as frame type, this allows the page treat these continuous jpeg image stream as mjpeg video.

MIME:

It is used to create email package. It can include multiple parts like the sender, receivers, subject, text, and attachment like image or video. It can send email by python just like using the email app or webpage. You can attach anything you want if it does not exceed the limitation of the email service you are using.

Gmail API:

We used to login Gmail by stmp, but it is not safe for it needs username and password in plain text. Now we use Gmail API to authenticate. We got the secret key of a Gmail account and used it to create a credential json file. The program only need the credential so the secret key does not need to be sent away. Also, if we want to change the permission of the program on Gmail, we can delete the old credential json file and change the 'scopes' on alert.authenticate to get a new credential for different permissions. ('<https://mail.google.com>' is for all permission. See more details on <https://developers.google.com/gmail/api/auth/scopes>)

Send Log to Google Drive:

This part is implemented successfully and smooth, because it is very similar to Lab2.

SSH and VNC:

With the aid of SSH and VNC, we achieved the function that users can access and control the raspberry pi remotely. Users can access Raspberry Pi 's command line by using SSH and access full desktop environment by VNC. To achieve this function, we need to make Raspberry Pi as a host and then input related information such as the IP address and corresponding port into SSH or VNC to access Raspberry Pi.

Moreover, what if user's phone and Raspberry Pi is not under the same local network environment? In other word, users may work outside under WAN network while the Raspberry Pi works in the home under LAN network. To solve this problem, it is necessary to use port forward to pass connections from outside network onto the IP of Raspberry Pi on port 22 for SSH or port 5900 for VNC. For convenience, we set Raspberry Pi a reserved IP address from DHCP so that user can access Raspberry Pi by only one click.

Efficiency:

The haar cascades face detection will cost ~0.02s per frame, and the time consumption of AWS Rekognition will be ~3s, including upload images to AWS S3 storage and get response. Push the frame to the web will cause a ~2s delay due to network delay. It will ~15s to send the alert email with the picture. It scan for email reply every 30s.

Required Libraries:

Camera and AWS Rekognition:

Pre-installed libraries: opencv

Need to be installed libraries :

Flask: \$ sudo pip install flask

boto3: \$ sudo pip install boto3

awscli: \$ sudo pip install awscli (configuration needed)

pantiltthat: \$ sudo pip install pantiltthat

Alert sending and email retrieving (including Gmail API):

Pre-installed libraries: httplib2, base64, os, email.mime, apiclient, time

Need to be installed libraries:

oauth2client: \$ sudo pip install --upgrade google-api-python-client

OpenSSL: \$ brew install openssl

gsread: \$ sudo pip install gsread

Reference:

OpenCV (2.4.9.1): <https://docs.opencv.org/2.4.9/>

Flask: <http://flask.pocoo.org/>

awscli: <https://aws.amazon.com/cli>

boto3: <https://boto3.readthedocs.io/en/latest/>

Pantiltthat: <https://github.com/pimoroni/pantilt-hat>

Gmail API: <https://developers.google.com/gmail/api/quickstart/python>

MIME: <https://docs.python.org/2/library/email-examples.html>

Video Demo:

<https://www.youtube.com/watch?v=EuehalPf9R8>