Hello everyone,

Yuntao and I wanted to share with you some facial analysis techniques we have learnt and invented for our Hackday project, and how to apply the techniques to solve business problems.

This presentation is not intended to improve our programming skills, so we will not be live coding. Rather, this presentation intends to introduce facial analysis related concepts, framework and algorithms, which hopefully will be helpful if you wanted to work on facial related applications someday.

There have been a lot of advances in facial analysis framework recently, and it can be overwhelming looking into this area, Yuntao and I got have researched and identified some industry mature packages, some of them you maybe familiar with. We will cover those packages, and more importantly is how to create your own application and algorithms on top of those packages.s

We will give a quick demo of the application, and then share the techniques to implement facial landmark detection, phone call detection and how to incorporate all the video streaming, analysis, prediction components into a Python user interface.

Without further due, let’s watch SMARTS in real time.

SMARTS is application that integrates machine learning and facial analytics monitoring and alerting driver abnormal driving in real time.

Smarts works by tracking a driver’s face and learns their standard posture, head position, facial expression and eye blinking rate. This allows SMARTS to alert the driver when they are engaging distracted driving behaviors.

Let’s watch that in action. As you can see, my partner is engaging in a series of distracted driving behaviors such as talking on the phone, not looking ahead and drowsy driving, ( ) and SMARTS is doing this in real time and can operate on any one’s face.

We recognize privacy is important to our clients. That is why SMARTS is built to run locally in car, so facial images stay with the user while relaying the driver’s behaviors to the insurer and the policy holder, producing a report like that seen on the screen. SMARTS also generate a report ….

SMARTS also has applications that go beyond safe driving. For example:

SMARTS can identify driving under the influence, detect and mitigate road rage, track occupants ensuring a small child is never again left in a hot car. Intelligently target airbag deployment, and deter theft by preventing unapproved drivers from operating the vehicle. ( )

**Slide 1:**

Facial landmarks are used to localize and represent salient regions of the face, such as: Eyes,Eyebrows,Nose,Mouth, Jawline

Facial landmarks have been successfully applied to face alignment, head pose estimation, face swapping, blink detection and much more. we’ll be focusing on the **basics of facial landmarks**, including:

Exactly *what* facial landmarks are and *how* they work.

How to detect and extract facial landmarks from an image using dlib, OpenCV, and Python.

Slide 2:

There are two steps in generating facial landmarks.

Step 1 is, and we know there are at least 3 algorithms/ packages to implement it. Step 2 is detecting facial structures on the detected face ROI, and we have found out two packages to do this.

Next, we will go over those algorithms quickly.

Slide 3.

1st method is Haar cascades face detection, and this algorithm has an implementation in OpenCV. This is probably the most well-known method for some of you, so I am not going to bore you too much on

this.

The right screen shot is an example of Haar cascade training in early stage, each black and white patch represents a fewature that the algorithm hunts for in the image. We can think of each patch as a convoluntional kernel, and each patch feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle. There will be lot of such patches generated, opencv documentation says even a 24x24 window results over 160000 features, how to select best features out of 160,000 of them? It is achieved by Adaboost. Basically you group features into a weak classifier and boosting those classifiers stage by stage.

We embedded a link where you download a variety of haar classifiers, and also a tutorial about train your own haar classifer.

Any one recognizes the lady in the example picture?

*Lena test image:*

Engineers, researchers, and students who are familiar with image processing or compression has most likely used the picture of "Lenna" or "Lena" in their experiments or project assignments, as the Lenna picture is one of the most widely used standard test images. Today, the use of Lenna image has been recognized as one of the most important events in the history of electronic imaging

Slide 4

HoG stands for Histogram of Gradient, also a relatively well known algorithm detecting objects, and you can probably implement your own HoG using scipy, numpy; the right part shows an example of detecting vehicles using HoG, and you can find the source code in the Github clicking the source hyper link.

How HoG works? The two animations I found online kinda visualizes the process. we’ll start by making our image black and white, Then we’ll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it:Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker. If you repeat that process for **every single pixel** in the image, you end up with every pixel being replaced by an arrow. These arrows are called *gradients* and they show the flow from light to dark across the entire image.

Slide 5

The 3 way to detect face is of course deep learning!

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's [open source licensing](http://dlib.net/license.html) allows you to use it in any application, free of charge.

Slide 6

After we localized faces, we can start extract facial landmarks.

There are two packages we found can do this. Dlib and openface;

We have dlib in hack day and openface at life selfie, they both work well and almost the same. For example, The facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 point mappings were obtained by training a shape predictor on the labeled.On the right side, we can visualize what each of these 68 coordinates map to. we can see that facial regions can be accessed via simple Python indexing (assuming zero-indexing with Python since the image above is one-indexed):

The mouth can be accessed through points [48, 68].

The right eyebrow through points [17, 22].

The left eyebrow through points [22, 27].

The right eye using [36, 42].

The left eye with [42, 48].

The nose using [27, 35].

And the jaw via [0, 17].

It’s important to note that other flavors of facial landmark detectors exist, including the 194 point model that can be trained on the [HELEN dataset](http://www.ifp.illinois.edu/~vuongle2/helen/).

So up to this point, we have covered the technological framework foundation of our project,

Let me give a short pause and throw a brain teaser to our audience, how would you write an algorithm to detect , we have created our algorithm for hack project, but we are curious to hear any other idea and potential solution that you may come up with as well.

In the remaining of this blog post I’ll demonstrate how to detect these facial landmarks in images.

Future blog posts in this series will use these facial landmarks to extract *specific* regions of the face, apply face alignment, and even build a blink detection system.

Slide 7 overview

Slide 8

We are going to take the head movement as an example.

We identify jaw, nose, eye brows, right and left side of face.

Then we compute the distance between nose center to right side face as x1, to left side face as x2, to jaw y2, to forehead as y1.

In real life, those distances remain consistent most of the time, unless somebody punched at my face that my nose moved to the side.

But camera takes the view into a 2D space, so I am moving my head from left to right, x1 will increase and x2 will decrease, because webcam sees more of the left side of face.

So is the same if I node or life up, y1 and y2 will change. So we measure the x and y movement ratio at thi

Slide 9

Our body is moving constantly, so the ratio actually changes all the time, an abnormal driving behavior should be counted only when the ratio change is significant to a baseline

We take the average of  first 5 frames ratio measurment as the baseline, and now we manualy decide upper and lower limit. –machine learning in the future

Slide 10

Next is the monitoring piece, we measure those ratios at each frame, if the ratio is beyond the thresholds, then we start count, if the count is above a threshold, indicating the hehavior is long enough , trigger an alert

Demo

Slide 11 : similar logic for other behaviors

Phone usage detection