PyCon Asia-Pacific Singapore

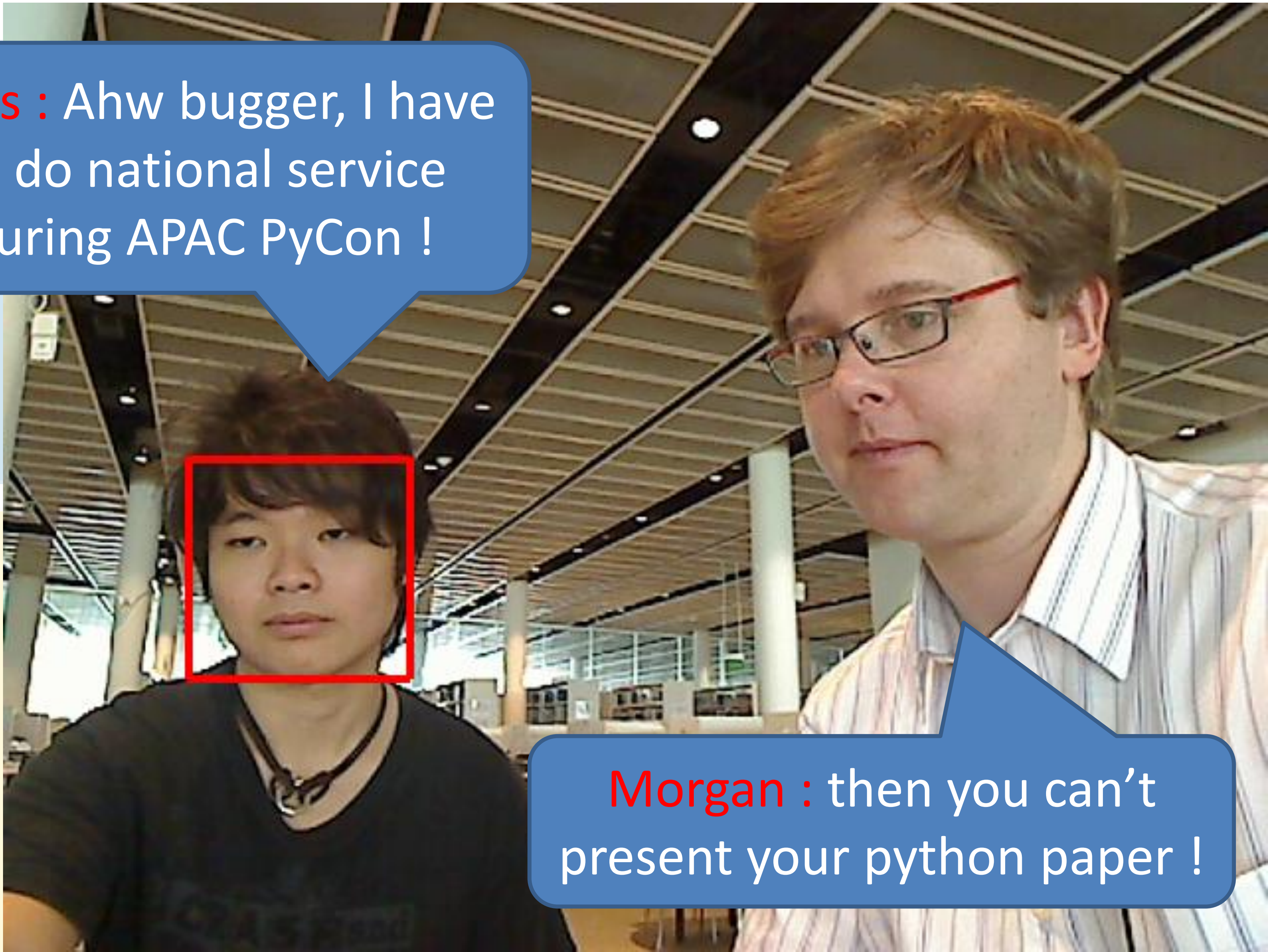
Face Detection Using OpenCV with Python

Linus Neo and Morgan Heijdemann

11.10am-11.50am 10/6/2010 APAC PyCON @ RM 2.2

OpenCV

introduction


Linus : Ahw bugger, I have to do national service during APAC PyCon !
Morgan : then you can't present your python paper !

introduction

PyCon Asia-Pacific Singapore

introduction

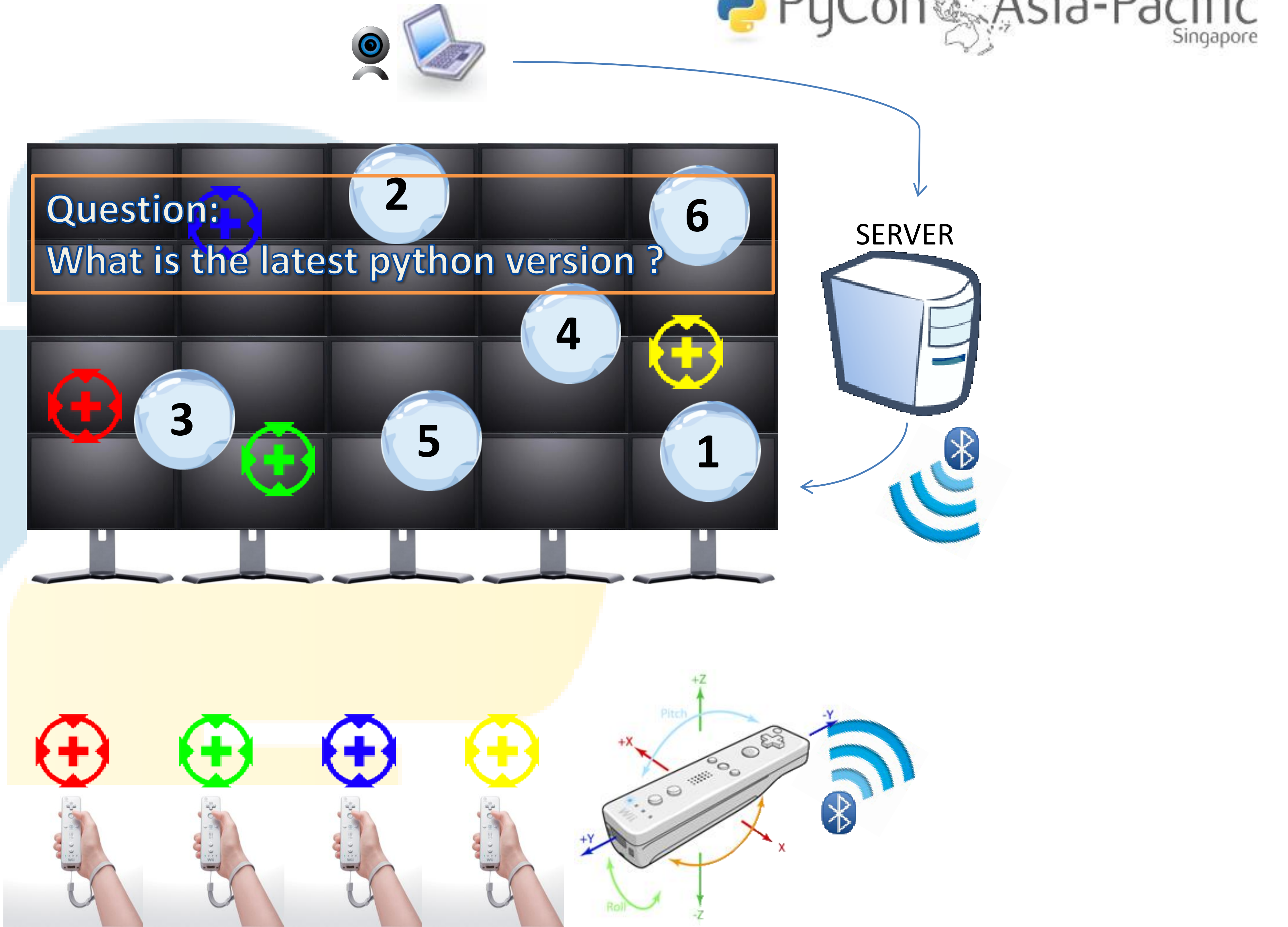
OpenCV (**Open** Source **C**omputer **V**ision) is a library containing real time computer vision functions.

It is used for facial tracking and for users to take picture of their faces to view for real time (gaming) purposes.

Early this year version 2 became stable and version 2.1 came out April 2010.

This talk is based on OpenCV_1.1pre1a and python-2.5.4 and PIL 1.1.7 and psyco.
You can download the ppt, installables and demo code at http://noxqs.com/files/opencv.zip

introduction


Question:
What is the latest python version ?
2
6
4
3
5
1
SERVER
+Z
Pitch
+X
-Y
+Y
X
Roll
-Z

the imports

Import the necessary library from openCV and also from openCV’s gui

```
# import the necessary libraries for OpenCV
from opencv import cv
from opencv import highgui
from opencv.cv import *
from opencv.highgui import *
```

OpenCV’s HighGui

OpenCV consisted of it’s own graphical user interface for use in full-scale applications.

It can be used within functionally rich UI frameworks (such as Qt, WinForms or Cocoa) or without any UI at all.

Sometimes there is a need to try some functionality quickly and visualize the results.

OpenCV

It provides easy interface to :

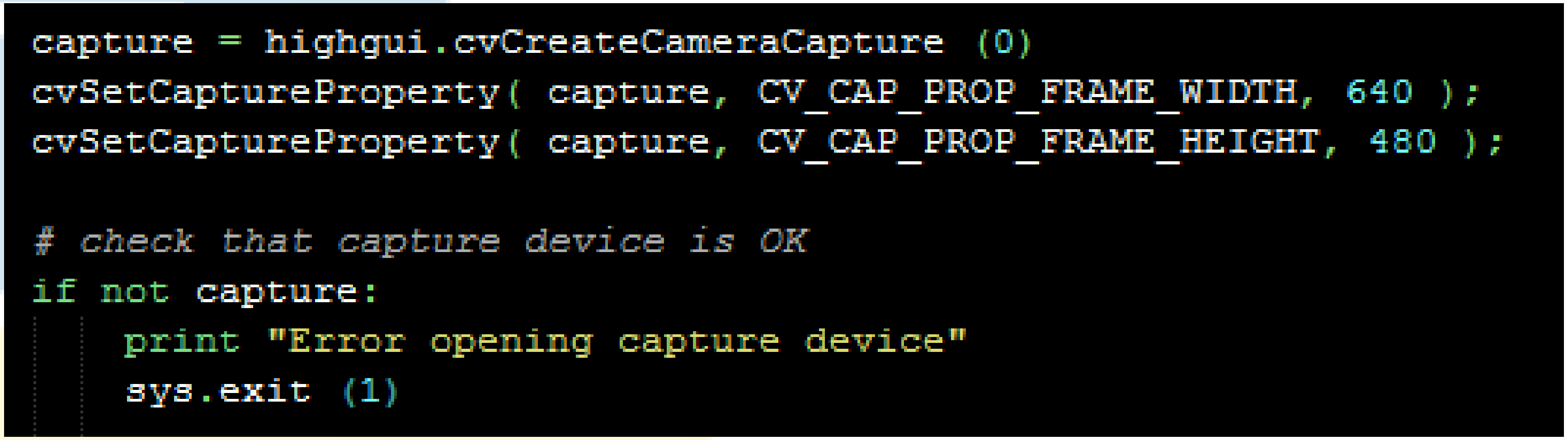
- create and manipulate windows that can display images and "remember" their content (no need to handle repaint events from OS.
- add track bars to the windows, handle simple mouse events as well as keyboard commands
- read and write images to/from disk or memory.
- read video from camera or file and write video to a file.

introduction

In our case, the highgui is to provide user input to take a picture and sending it to the syncing it with other clients via SVN in the back-end. It also allow users to view their facial features being recognized and showing video frames captured by the camera.

Init video

Initialize video capturing from camera and try to open the capture device



```
capture = highgui.cvCreateCameraCapture (0)
cvSetCaptureProperty( capture, CV_CAP_PROP_FRAME_WIDTH, 640 );
cvSetCaptureProperty( capture, CV_CAP_PROP_FRAME_HEIGHT, 480 );

# check that capture device is OK
if not capture:
    print "Error opening capture device"
    sys.exit (1)
```

CvCapture cvCreateCameraCapture Function

The function cvCreateCameraCapture allocates and initialized the CvCapture structure for reading a video stream from the camera. Currently two camera interfaces can be used on Windows: Video for Windows (VFW) and Matrox Imaging Library (MIL); and two on Linux: V4L and FireWire (IEEE1394).

```
CvCapture* cvCreateCameraCapture( int index );
```

index
Index of the camera to be used. If there is only one camera or it does not matter what camera to use -1 may be passed.

```
capture = highgui. cvCreateCameraCapture ( 0 )
```

CvCapture cvCreateFileCapture Function

The function cvCreateFileCapture allocates and initialized the CvCapture structure for reading the video stream from the specified file.

CvCapture* cvCreateFileCapture(const char* filename);

filename
Name of the video file.

Filename can also be past through the command line if argument exist on the command line.

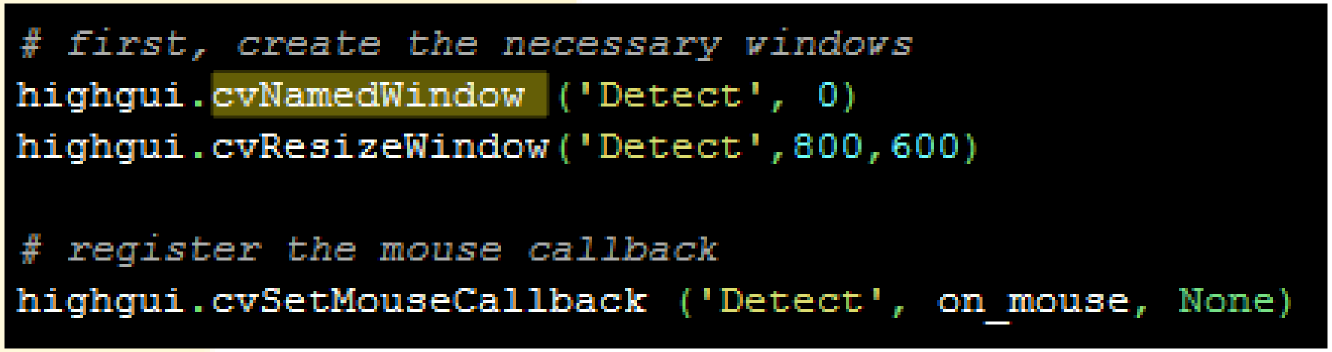
capture = highgui. cvCreateFileCapture (sys.argv[1])

HighGUI cvNamedWindow Function

This function creates a window to show images and trackbars, images can be form of frames captured by the camera. Created windows is referred by their names.

int cvNamedWindow(const char* name, int flags=CV_WINDOW_AUTOSIZE);
name : Name of the window which is used as window identifier and appears in the window caption.
flags : Flags of the window. Currently the only supported flag is CV_WINDOW_AUTOSIZE. If it is set, window size is automatically adjusted to fit the displayed image.

Created windows can be resized using the **cvResizeWindow** function,
void cvResizeWindow(const char* name, int width, int height);
where the parameters are the name of the created window, width and height of the window.



```
# first, create the necessary windows
highgui.cvNamedWindow ('Detect', 0)
highgui.cvResizeWindow('Detect',800,600)

# register the mouse callback
highgui.cvSetMouseCallback ('Detect', on_mouse, None)
```

Main Loop

The main loop is responsible for the application to constantly use the frame taken by the camera and show it on the window. It is also used to handle events for key pressed for taking picture and also leaving the application.

Main Loop

Function consist in a main loop to create a basic window that shows captured frame

```
while 1:
    #capture the current image
    frame = highgui.cvQueryFrame( capture )
    #handle event, wait for user key input
    events = highgui.cvWaitKey( 10 )
    # display the frames to have a visual output
    highgui.cvShowImage( 'Camera', frame )
```

At this point, starting the application will allow user to see the frame being captured and rendering it onto the screen(window).

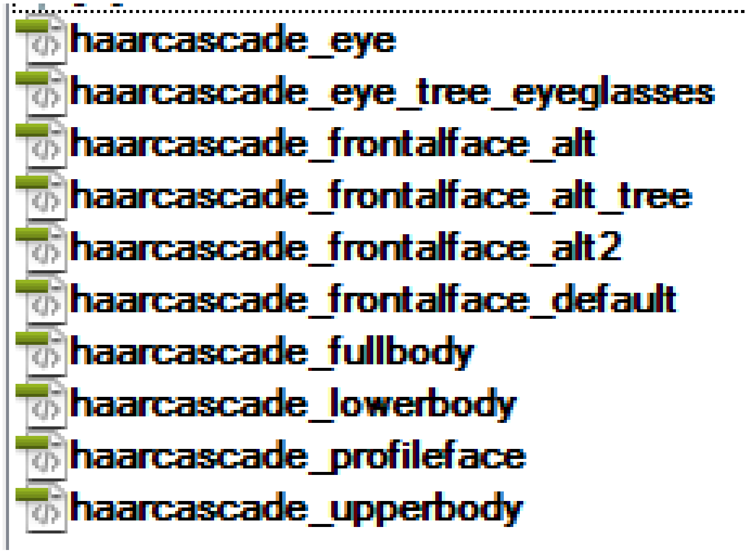
Facial Tracking

In the main loop, there is a function which capture and return frame (cvQueryFrame) and another function which display the frames(cvShowImage). After the camera capture a frame and before displaying it, we can apply some functions on the captured frame such as facial tracking

Facial Tracking

To do this, you need a HAAR cascade(xml files) which are digital image features used in object recognition.

This feature set considers rectangular regions of the image and sums up the pixels in this region. This sum is used to categorize images. HAAR cascade can be created or found in the internet.


haarcascade_eye
haarcascade_eye_tree_eyeglasses
haarcascade_frontalface_alt
haarcascade_frontalface_alt_tree
haarcascade_frontalface_alt2
haarcascade_frontalface_default
haarcascade_fullbody
haarcascade_lowerbody
haarcascade_profileface
haarcascade_upperbody

implementation

```
# allocate temporary images
gray = cvCreateImage( cvSize(img.width,img.height), 8, 1 )
small_img = cvCreateImage(
     cvSize( cvRound ( img.width/self.image_scale ),
     cvRound (img.height/self.image_scale ) ), 8, 1
   )

# convert color input image to grayscale
cvCvtColor( img, gray, CV_BGR2GRAY )

# scale input image for faster processing
cvResize( gray, small_img, CV_INTER_LINEAR )
```

Bounding box

From all the received faces, create two CV points for each bounding box of face.

This will return a cv point object that consist of 2 points for each x and y.

The code below change the cvPoint into string so that the value can be extracted with String.split() function.

The point are extracted so that PIL will know where to crop.

Finding the boundingbox

After that draw a rectangle on the frame to show the location of the face from the two cv Points given.

```
if faces:
   for face_rect in faces:

# the input to cvHaarDetectObjects was resized, so scale
# the bounding box of each face and convert it to two
# CvPoints
pt1=cvPoint(int(face_rect.x*self.image_scale),
                int(face_rect.y*self.image_scale))

pt2=cvPoint(int((face_rect.x+face_rect.width)*
   self.image_scale),
   int((face_rect.y+face_rect.height)*self.image_scale))
```

Drawing box

```
#extract each point of the box using split function
ptt1 = str(pt1).split(' ')
ptt2 = str(pt2).split(' ')
p1 = ptt1[0].split('[')
p2 = ptt1[1].split(']')
p3 = ptt2[0].split('[')
p4 = ptt2[1].split(']')

#draw a rectangle for the 2 cvPoint
cvRectangle( img, pt1, pt2, CV_RGB(255,0,0), 3, 8, 0 )
```

OpenCV with PIL(Python Image Library)

In the digibanner, when the user input the key to take a picture, a new picture will be created showing image of a cropped face using facial tracking mapped onto a bubble. The way digibanner is done is it made use of PIL(Python Image Library) to crop, paste and save.

introduction

When the input key is pressed, the highGUI will save a .png image of the last frame taken from the camera.

After saving, PIL will reopen the picture and crop the points given from the facial tracking.

Using PIL to open the picture of the bubble and resizing the cropped face to match the bubble. Map the face to the bubble and save it to a bubble folder, the name of the created picture will be “bubbles%s.png” where %s = total number of pictures already created inside the folder.

introduction

```
# Crop face from picture and map it onto a bubble:
# save the last frame when user input the key to take a
# picture
highgui.cvSaveImage( "lastFrame.png", frame )

#open the last frame using PIL library
captured_face =  Image.open( "lastFrame.png" )

#crop the face using the points from the facial tracking
# and pass it into a variable
captured_face=captured_face.crop( ( pt1, pt2, pt3, pt4 ) )
```

introduction

```
# resize the cropped face, map it onto a bubble picture and finally save it into
# the bubble folder
final_face = captured_face.resize( ( 160, 160 ) )
bubble_img = Image.open( "bubble.png" )
bubble_img.paste( final_face, ( 45, 45 ) )
bubble_img.save( "bubbles/bubbles%s.png"%(
                                len( os.listdir( os.getcwd()+"/bubbles" ) )
                        ) )
```

PIL Image crop function

This function returns a rectangular region from the current image.

im.crop(box)

box

The box is a 4-tuple defining the left, upper, right, and lower pixel coordinate.

im

This represent the image to crop.

PIL Image paste Function

This function pastes another image into this image.
im.paste(image, box)

image
This the image to paste on im. In digibanner, it represents the cropped face.

box
The box is a 4-tuple defining the left, upper, right, and lower pixel coordinate. It represents the location to paste on the image

im
This represent the image to crop.

PIL Image save Function

This function pastes another image into this image.
im.save(name)

name
This represent the directory and name of the image to save.

im
This represent the image to crop.

SVN statement on command line

Sync all the folders with all the clients in digibanner using SVN so that all clients will have the image to show when the bubble goes to the screen.

An example of this SVN command:

os.system("d:\\digibanner\\tools\\svn\\svn.exe add *.png") os.system("d:\\digibanner\\tools\\svn\\svn.exe commit *.png –m added_face")

References

OpenCV wiki

http://opencv.willowgarage.com

OPenCV Python Interface

http://opencv.willowgarage.com/documentation/python/index.html

Python Image Library wiki

http://www.pythonware.com/library/pil/handbook/index.htm

QUESTION ?

THANK YOU !

For more info contact

Neo Yong Sheng Linus
Gratuated Republic Polytechnic student
ys.linus.neo@gmail.com

Demonstration next !