Eye Power

This project contains 5 Python files, and a folder [/eye] that contains the images training and test images.

1. **Utils.py:** Contains functions used by the other 4 Py files. All functions were written as methods within the *Utils* class. The CNN model is also found in this file.

2. **capture_training_images.py:** Launches a window with a yellow dot against a black background. Users should

   a) maximize the window so that it covers the entire screen

   b) fix the position of your right eye relative to the screen

   c) cover the left eye, and focus the right eye on the yellow dot

   d) press 'c' to capture and save JPEGs of the right eye in eye/train

   e) repeat d) until desired number of training files have been saved

3. **train_test_split.py:** Transfers 10% of the images in eye/train to eye/test. Ratio is adjustable.

4. **train_cnn.py:** Trains the CNN using the image files in eye/train and eye/test folders. Saves the model weights in "eye_model.hdf5". Run this file only if you wish to train the model.

5. **demo.py:** Loads the model weights from "eye_model.hdf5", and launches a window with a green dot which you can control with your right eye.

Additional Info:

- The scripts are designed to detect human eyes using Haar Cascades (a function within OpenCV). It’s a bit like a pre-trained object detector for different objects. The one used here is for the right eye. More info: https://towardsdatascience.com/object-detection-with-haar-cascades-in-python-ad9e70ed50aa

Basic OpenCV Commands and APIs

- **Basic Imports**

```
Import numpy as np
Import matplotlib.pyplot as plt
Import cv2
From PIL import Image
```

- **Load and display an image with PIL.** Images imported using PIL need to be converted into numpy arrays first.

```
pic = Image.open(‘../DATA/file.jpg’)
pic_array = np.array(pic)
plt.imshow(pic[:,:,0], cmap=’gray’) #Show image
```

- **Reading an image directly using cv2**

```
pic = cv2.imread(‘../DATA/file.jpg’)
type(pic) → numpy.ndarray
```

- **Images imported using cv2 will be in BGR colour format. We need to convert it into other colour models for proper viewing using matplotlib.**

```
img = cv2.imread(‘data/file.jpg’)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  #RGB
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #Gray scale
```

- **Resize an image (absolute and by ratio)**

```
# Absolute
new_img = cv2.resize(img, (1000,600)) # width, height
plt.imshow(new_img)

# Ratio
w_ratio = 0.3
h_ratio = 0.5
new_img = cv2.resize(img, (0,0), w_ratio, h_ratio)
```

- **Flip an image**

```
new_img = cv2.resize(img, (1000,600)) # width, height
plt.imshow(new_img)

w_ratio = 0.3
h_ratio = 0.5
new_img = cv2.resize(img, (0,0), w_ratio, h_ratio)
```

- **Save an image**

```
cv2.imwrite(‘image_file.jpg’, new_img)
```

Draw on Images

- **Create a blank image**

```
img = np.zeros((512,521,3), dtype=np.int16)
```

- **Draw shapes.** Shapes are immediately rendered onto the image when the function is called.

```
cv2.rectangle(img,
              pt1=(100,200), # top left
              pt2=(300,400),  # bottom right
              color=(255,0,0),
              thickness=5)
cv2.circle(img,
           pt1=(100,200),
           pt2=(300,400),
           color=(255,0,0),
           thickness=-1) #-1 indicates fully shaded circle
cv2.line(img,
         pt1=(100,200), # start
         pt2=(300,400), # end
         color=(255,0,0),
         thickness=5)

plt.imshow(img)
```

- **Rendering text**

```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,
            text=’Hello my text’,
            fontFace=font,
            fontScale=1, # font size
            color=(255,255,255),
            thickness=4,
            org=(100,200),
            lineType=cv2.LINE_AA)

plt.imshow(img)
```

- **Polygon Drawing**. Need to define the vertices of the polygon first.

<u>Working with Videos</u>

- **Loading and editing a video.** The *VideoCapture* function is not designed for people to view videos. It is designed to loop over the images for editing and calculation. To view the video normally, we have to add a time delay equivalent to 1/fps.

```
import cv2
import time

fps = 20

cap = cv2.VideoCapture(‘video.mp4’)

if cap.isOpened() == False:
      #Check if initialised
      print(“File not found or wrong CODEC”)

while cap.isOpened():
      # ret is will be True if frame is read correctly.
      # ret will be False if it comes to the end of the video.
      ret, frame = cap.read()

      if ret==True:

            # Optional for viewing purposes
            time.sleep(1/fps)

            cv2.imshow(‘display_window’, frame)

            if cv2.waitKey(10) & 0xFF==ord(‘q’)
                  break

      else:
            break

cap.release()
cv2.destoryAllWindows()
```

- **Draw static shapes on videos.** Shapes are immediately rendered onto the image when the function is called.

```
cap = cv2.VideoCapture(0)  # Uses webcam

w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

# draw a rect starting at w/2 and h/2, of dim 10 by 15
x=w/2
y=h/2
x_dim= 10
y_dim= 15

while True:
      ret, frame = cap.read()

      # Draw the rectangle here
      cv2.rectangle(frame,
                    pt1=(x,y),
                    pt2=(x+x_dim, y+y_dim),
                    thickness=5,
                    color=(225,225,255))

      cv2.imshow(‘display_window’, frame)

      if cv2.waitKey(10) & 0xFF == 27:
            break

cap.release()
cv2.destroyAllWindows()
```