

Face data type and format from different face detectors

Shengping Jiang

This notebook explains face rectangle data type and from from dlib hog detector, dlib cnn detector, OpenCV Haar Cascade detector, face_recognition.face_locations and mtcnn.detect_faces()

```
In [1]: 1 # Several face detection methods. Running on face1
2 import numpy as np
3 import os
4 import face_recognition as frg
5 from sklearn.neighbors import KNeighborsClassifier
6 import re
7 import math
8 import mtcnn
9 import matplotlib.pyplot as plt
```

```
In [2]: 1 from PIL import Image, ImageDraw, ImageFont
2 from IPython.display import display
3
4 def show_face(img_path, location):
5     #location is a rectangle in order [left,top,right,bottom]
6     pil_image = Image.open(img_path).convert("RGB")
7     [left,top,right,bottom] = location
8
9     draw = ImageDraw.Draw(pil_image)
10    draw.rectangle(((right, top), (left, bottom)), outline=(0, 0, 255))
11    display(pil_image)
```

Face type and format from dlib.get_frontal_face_detector()

The dlib.get_frontal_face_detector() returns a face detector with HOG model

Type of faces from HOG detector: <class '_dlib_pybind11.rectangles'>.

This is a dlib data type. A two faces example: rectangles[[$(79, 130) (154, 204)$], [$(247, 92) (354, 199)$]]. The rectangles looks a python list

Type of face is: <class '_dlib_pybind11.rectangle'>. It is not a python list or tuple. The format is [(left,top) (right, bottom)]. For example: [$(79, 130) (154, 204)$]

Each number in the rectangle can get in below:

```
left = face.left()  
top = face.top()  
right = face.right()  
bottom = face.bottom()
```

Face type and format from `dlib.cnn_face_detection_model_v1(dlib_cnn_model_path)`

Type of faces from cnn detector: <class '_dlib_pybind11.mmod_rectangles'>.

This is a dlib data type. A two faces example: mmod_rectangles[[$(260, 97) (342, 179)$], [$(70, 122) (151, 204)$]].

Type of face is: <class '_dlib_pybind11.mmod_rectangle'>. It is not a python list or tuple but a <`_dlib_pybind11.mmod_rectangle` object>.

Type of face.rect: <class '_dlib_pybind11.rectangle'>

The face rectangle is represented as face.rect. The format is [(left,top) (right, bottom)]. For example: [(79, 130) (154, 204)]

Each number in the rectangle can be obtained as below:

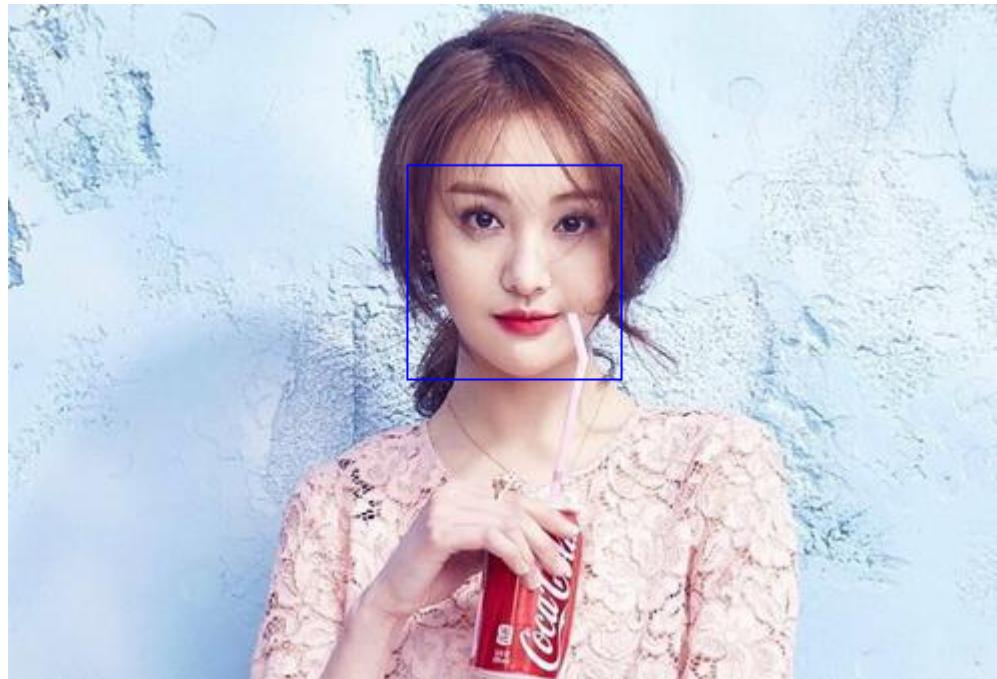
```
left = face.rect.left()  
top = face.rect.top()  
right = face.rect.right()  
bottom = face.rect.bottom()
```

In [3]:

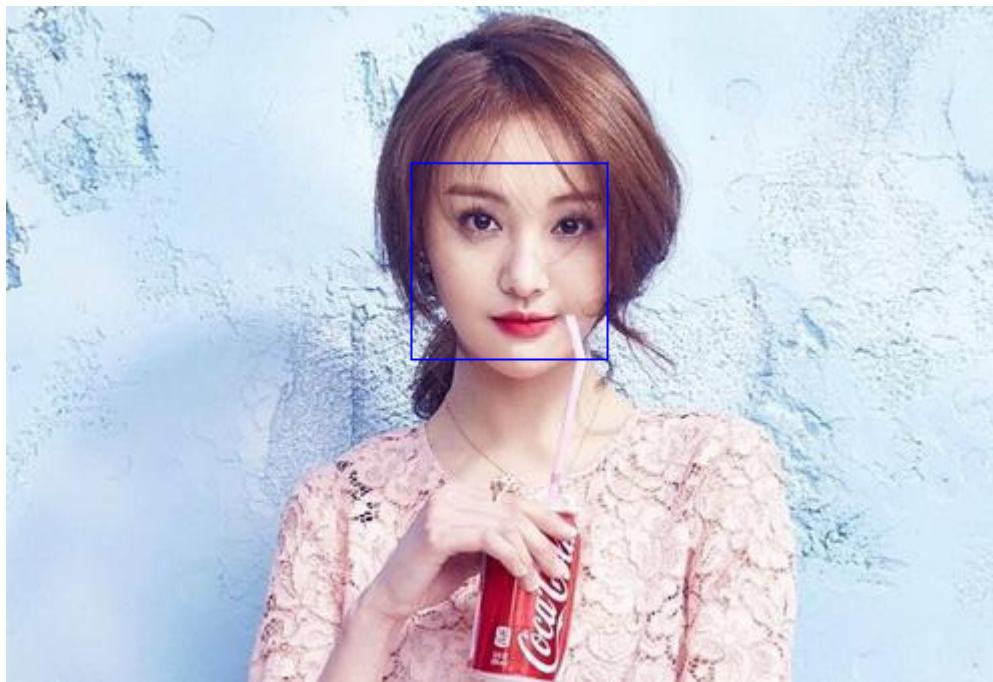
```
1 # dlib_hog_face_detector()---dlib.get_frontal_face_detector()
2
3 import dlib
4 import cv2
5
6 dlib_hog_face_detector = dlib.get_frontal_face_detector()
7 dlib_cnn_model_path = 'models/mmod_human_face_detector.dat'
8 dlib_cnn_face_detector = dlib.cnn_face_detection_model_v1(dlib_cnn_model_path)
9 #img_path = 'notebook_images/kit_with_rose.jpg'
10 img_path = 'notebook_images/015.jpg'
11 img = cv2.imread(img_path)
12
13 #Test hog model
14 # type of face: <class '_dlib_pybind11.rectangle'>
15 #_dlib_pybind11.rectangle format: [(left,top) (right,bottom)]
16 faces = dlib_hog_face_detector(img, 1)
17 print('type of faces(hog)', type(faces))
18 print('faces(hog):', faces)
19
20 #to draw faces on image
21
22 for face in faces:
23     print('type of face(hog):', type(face))
24     print('face(hog):', face)
25     left = face.left()
26     top = face.top()
27     right = face.right()
28     bottom = face.bottom()
29     location = [left,top,right,bottom]
30     #print("type of location:",type(location))
31     #print("location:",location)
32     print("Face found with HOG model:")
33     show_face(img_path, location)
34
35 # test cnn model
36 # type of face: <class '_dlib_pybind11.mmod_rectangle'>
37 #_dlib_pybind11.mmod_rectangle.rect -> [(left,top) (right,bottom)]
38 faces = dlib_cnn_face_detector(img, 1) # result
39 print('type of faces(cnn)', type(faces))
40 print('faces(cnn):', faces)
41 #to draw faces on image
42
```

```
43 for face in faces:  
44     print('type of face.rect(cnn)', type(face.rect))  
45     print('face.rect(cnn):', face.rect)  
46     left = face.rect.left()  
47     top = face.rect.top()  
48     right = face.rect.right()  
49     bottom = face.rect.bottom()  
50     location = [left,top,right,bottom]  
51     #print("type of location:", type(location))  
52     #print("location:", location)  
53     print("Face found with CNN model:")  
54     show_face(img_path, location)
```

```
type of faces(hog) <class '_dlib_pybind11.rectangles'>  
faces(hog): rectangles[[199, 80) (306, 187)]]  
type of face(hog): <class '_dlib_pybind11.rectangle'>  
face(hog): [(199, 80) (306, 187)]  
Face found with HOG model:
```



```
type of faces(cnn) <class '_dlib_pybind11.mmod_rectangles'>
faces(cnn): mmod_rectangles([(202, 78) (300, 176)])
type of face.rect(cnn) <class '_dlib_pybind11.rectangle'>
face.rect(cnn): [(202, 78) (300, 176)]
Face found with CNN model:
```



Face data type and format from OpenCV cv2.CascadeClassifier

type of faces(openCV): <class 'numpy.ndarray'>
faces format(openCV): [[75 117 87 87] [244 89 105 105]]
type of face(openCV): <class 'numpy.ndarray'>
face format(openCV): [left top width height]. for example: [75 117 87 87]
Get each number of a face rectangle as below:
left, top, width, height = face

In [4]:

```
1 # Haar Cascade
2 # https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c
3
4 import cv2
5 classifier = cv2.CascadeClassifier('models/haarcascade_frontalface_alt2.xml')
6 #img_path = 'notebook_images/kid_with_rose.jpg'
7 img_path = 'notebook_images/001.jpg'
8 img = cv2.imread(img_path)
9 faces = classifier.detectMultiScale(img)# result
10 print('type of faces(openCV):', type(faces))
11 print('faces format(openCV):', faces)
12
13 # type of face(openCV): <class 'numpy.ndarray'>
14 # face format(openCV): [x y w h]
15 #to draw faces on image
16 for face in faces:
17     print('type of face(openCV):', type(face))
18     print('face format(openCV):', face)
19     x, y, w, h = face
20     location = [x, y, x+w,y+h]
21     show_face(img_path, location)
```

```
type of faces(openCV): <class 'numpy.ndarray'>
faces format(openCV): [[89 24 84 84]]
type of face(openCV): <class 'numpy.ndarray'>
face format(openCV): [89 24 84 84]
```



Face data type and format from `face_recognition.face_locations` are as below:

Faces (if more than one face found) is a list of tuples. Each tuple has four numbers which is a face rectangle. Example: [(130, 154, 204, 79), (92, 354, 199, 247)] (two face rectangles).

Each face rectangle is in format: (top, right, bottom, left). For example: (130, 154, 204, 79)

When use `ace_recognition`, Face data type and format are same either use hog model or cnn model

In [8]:

```
1 # face_recognition
2 # Face rectangles detected by HOG model
3 import face_recognition as frg
4
5 img_path = 'notebook_images/kit_with_rose.jpg'
6 img = cv2.imread(img_path)
7
8 #Use face_recognition.face_locations(image, model='hog')
9 #format: A list of tuples face locations (top, right, bottom, left)
10 faces = frg.face_locations(img, model='hog')
11 print('type of faces:', type(faces))
12 print('faces:', faces)
13
14 for face in faces:
15     print('type of face(hog):', type(face))
16     print('face(hog):', face)
17     # you can use [top, right, bottom, left] = face, (top, right, bottom, left) = face
18     # and below method to get each number
19     top, right, bottom, left = face
20     print('top, right, bottom, left:', top, right, bottom, left)
21     # change to location format of show_face()
22     location = [left, top, right, bottom]
23     print("Face found with HOG model:")
24     show_face(img_path, location)
25
26
27
28
```

```
type of faces: <class 'list'>
faces: [(130, 154, 204, 79), (92, 354, 199, 247)]
type of face(hog): <class 'tuple'>
face(hog): (130, 154, 204, 79)
top, right, bottom, left: 130 154 204 79
Face found with HOG model:
```



```
type of face(hog): <class 'tuple'>
face(hog): (92, 354, 199, 247)
top, right, bottom, left: 92 354 199 247
Face found with HOG model:
```



In [9]:

```
1 # face_recognition
2 # Face rectangles detected by CNN model
3 import face_recognition as frg
4
5 img_path = 'notebook_images/kit_with_rose.jpg'
6 img = cv2.imread(img_path)
7
8 #Use face_recognition.face_locations(image, model='cnn')
9 #format: A list of tuples face locations (top, right, bottom, left)
10 faces = frg.face_locations(img, model='cnn')
11 print('type of faces:', type(faces))
12 print('faces:', faces)
13
14 for face in faces:
15     print('type of face(cnn):', type(face))
16     print('face(cnn):', face)
17     # you can use [top, right, bottom, left] = face, (top, right, bottom, left) = face
18     # and below method to get each number
19     top, right, bottom, left = face
20     print('top, right, bottom, left:', top, right, bottom, left)
21     # change to location format of show_face()
22     location = [left, top, right, bottom]
23     print("Face found with CNN model:")
24     show_face(img_path, location)
25
```

```
type of faces: <class 'list'>
faces: [(97, 342, 179, 260), (122, 151, 204, 70)]
type of face(cnn): <class 'tuple'>
face(cnn): (97, 342, 179, 260)
top, right, bottom, left: 97 342 179 260
Face found with CNN model:
```



type of face(cnn): <class 'tuple'>

```
face(cnn): (122, 151, 204, 70)
top, right, bottom, left: 122 151 204 70
Face found with CNN model:
```



Face data type and format from mtcnn.detect_faces(img)

1 The type of faces is a list. But each element (face) of the list is a dict(字典). The format of faces is [...],{...},...{...}]

The face rectangle is the value of key 'box' of a dict. It is face['box'] while face is an element of faces

The face['box'] is a list of four numbers. It is in format [left, top, width, height]. For example: face['box']: [78, 112, 74, 95]

In [11]:

```
1 # mtcnn
2 import mtcnn
3
4 img_path = 'notebook_images/biden2.jpg'
5 #img_path = 'notebook_images/kit_with_rose.jpg'
6 img = cv2.imread(img_path)
7
8 detector = mtcnn.MTCNN()
9 # detect faces in the image
10 faces = detector.detect_faces(img)
11 print('type of faces:', type(faces))
12 print('faces:', faces)
13 for face in faces:
14     # get coordinates
15     print('Type of face(mtcnn):', type(face))
16     print('format of face(mtcnn):', face)
17     print("type of face['box']: ", type(face['box']))
18     print("face['box']: ", face['box'])
19     x, y, w, h = face['box']
20     print('x, y, w, h: ', x, y, w, h)
21     #change to location of show_face()--[left, top, right, bottom]
22     location = [x, y, x+w, y+h]
23     print('locations:', location)
24     show_face(img_path, location)
25
26
27
```

```
type of faces: <class 'list'>
faces: [{`box`: [432, 177, 449, 581], `confidence`: 0.9973528385162354, `keypoints`: {`left_eye`: (560, 400), `right_eye`: (755, 410), `nose`: (652, 526), `mouth_left`: (538, 593), `mouth_right`: (763, 604)}}, ...
Type of face(mtcnn): <class 'dict'>
format of face(mtcnn): {`box`: [432, 177, 449, 581], `confidence`: 0.9973528385162354, `keypoints`: {`left_eye`: (560, 400), `right_eye`: (755, 410), `nose`: (652, 526), `mouth_left`: (538, 593), `mouth_right`: (763, 604)}}
type of face['box']: <class 'list'>
face['box']: [432, 177, 449, 581]
x, y, w, h: 432 177 449 581
locations: [432, 177, 881, 758]
```







In []:

1

In []:

1