FaceMaskDetection_Preprocessing_WWMR

August 12, 2022

[1]:
```
import os
import shutil
sorted(os.listdir())
```

[1]:
```
['.DS_Store',
 '.ipynb_checkpoints',
 'CS6220 - Final Project - Face Mask Detection.pdf',
 'Dataset_853',
 'Face Mask Dataset',
 'MaskClassification.ipynb',
 'Not_Worn',
 'README.txt',
 'Untitled.ipynb',
 'WWMR-DB - Labels',
 'WWMR-DB - Part 1',
 'WWMR-DB - Part 2',
 'Worn_Correctly',
 'Worn_Incorrectly',
 'backup',
 'danya_headshot.jpg',
 'darknet',
 'darknet53.conv.74',
 'haarcascade_frontalface_default.xml',
 'haarcascade_profileface.xml',
 'obj',
 'profile.jpeg',
 'profile.png',
 'results']
```

[9]:
```
from PIL import Image
```

[32]:
```
#Create New Folders if they dont exist
if not os.path.isdir('Worn_Correctly'):
    os.makedirs("Worn_Correctly")

if not os.path.isdir('Not_Worn'):
    os.makedirs("Not_Worn")
```

[33]:
```
#Make functions for copying images into a single folder
def copyImages(start, destination, category, png = True):
    for folder in os.listdir(start):
        subdir = os.path.join(start, folder)
        if os.path.isdir(subdir):

            subdir2 = os.path.join(subdir, category)
            if os.path.isdir(subdir2):

                for folder2 in os.listdir(subdir2):
                    subdir3 = os.path.join(subdir2, folder2)
                    if os.path.isdir(subdir3):
                        for file in os.listdir(subdir3):
                            image = os.path.join(subdir3, file)

                            if png:
                                im1 = Image.open(image)
                                im1.save(os.path.join(destination, file.split(".
 ↪")[0] +'.png'))
                            else:
                                shutil.copy2(image, destination)


def copyImages2(start, destination, category, png = True):
    for folder in os.listdir(start):
        subdir = os.path.join(start, folder)
        if os.path.isdir(subdir):

            subdir2 = os.path.join(subdir, category)
            if os.path.isdir(subdir2):

                for file in os.listdir(subdir2):
                    image = os.path.join(subdir2, file)

                    if png:
                        im1 = Image.open(image)
                        im1.save(os.path.join(destination, file.split(".")[0]␣
 ↪+'.png'))
                    else:
                        shutil.copy2(image, destination)
```

0.0.1 Copy Images to Binary Classification Folders

[34]:
```
start = os.getcwd() + "/WWMR-DB - Part 1"
destination = os.getcwd() + "/Worn_Correctly"
category = "Mask Or Respirator Correctly Worn"
copyImages(start, destination, category)
```

```
len(os.listdir(destination))
```

[34]: 70

[35]:
```
start = os.getcwd() + "/WWMR-DB - Part 2"
destination = os.getcwd() + "/Worn_Correctly"
category = "Mask Or Respirator Correctly Worn"
copyImages(start, destination, category)
len(os.listdir(destination))
```

[35]: 152

[36]:
```
start = os.getcwd() + "/WWMR-DB - Part 1"
destination = os.getcwd() + "/Not_Worn"
category = "Mask Or Respirator Not Worn"
copyImages2(start, destination, category)
print(len(os.listdir(destination)))

start = os.getcwd() + "/WWMR-DB - Part 2"
destination = os.getcwd() + "/Not_Worn"
category = "Mask Or Respirator Not Worn"
copyImages2(start, destination, category)
print(len(os.listdir(destination)))
```

```
46
113
```

[37]:
```
start = os.getcwd() + "/WWMR-DB - Part 1"
destination = os.getcwd() + "/Worn_Incorrectly"
category = "Mask Or Respirator Hanging From An Ear"
copyImages(start, destination, category)
print(len(os.listdir(destination)))

start = os.getcwd() + "/WWMR-DB - Part 2"
destination = os.getcwd() + "/Worn_Incorrectly"
category = "Mask Or Respirator Hanging From An Ear"
copyImages(start, destination, category)
print(len(os.listdir(destination)))
```

```
114
241
```

[38]:
```
start = os.getcwd() + "/WWMR-DB - Part 1"
destination = os.getcwd() + "/Worn_Incorrectly"
category = "Mask Or Respirator On The Forehead"
copyImages(start, destination, category)
print(len(os.listdir(destination)))
```

```
start = os.getcwd() + "/WWMR-DB - Part 2"
destination = os.getcwd() + "/Worn_Incorrectly"
category = "Mask Or Respirator On The Forehead"
copyImages(start, destination, category)
print(len(os.listdir(destination)))
```

```
311
393
```

[39]:
```
start = os.getcwd() + "/WWMR-DB - Part 1"
destination = os.getcwd() + "/Worn_Incorrectly"
category = "Mask Or Respirator Under The Chin"
copyImages(start, destination, category)
print(len(os.listdir(destination)))

start = os.getcwd() + "/WWMR-DB - Part 2"
destination = os.getcwd() + "/Worn_Incorrectly"
category = "Mask Or Respirator Under The Chin"
copyImages(start, destination, category)
print(len(os.listdir(destination)))
```

```
463
545
```

[40]:
```
start = os.getcwd() + "/WWMR-DB - Part 1"
destination = os.getcwd() + "/Worn_Incorrectly"
category = "Mask Or Respirator Under The Nose"
copyImages(start, destination, category)
print(len(os.listdir(destination)))

start = os.getcwd() + "/WWMR-DB - Part 2"
destination = os.getcwd() + "/Worn_Incorrectly"
category = "Mask Or Respirator Under The Nose"
copyImages(start, destination, category)
print(len(os.listdir(destination)))
```

```
615
697
```

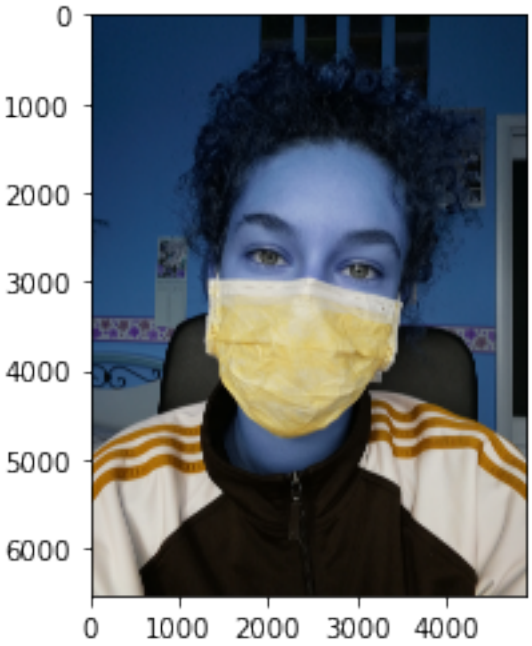
0.0.2 Read Images to create list

[3]:
```
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[10]:
```
folder = os.getcwd() + "/Worn_Correctly"
wc_list = [cv2.imread(os.path.join(folder, image)) for image in os.
 ↪listdir(folder)]
```

[11]:
```
folder = os.getcwd() + "/Not_Worn"
nw_list = [cv2.imread(os.path.join(folder, image)) for image in os.
 ↪listdir(folder)]
```

[109]:
```
plt.imshow(wc_list[0])
plt.show()
```



0.0.3 How many unique image dimensions are there?

[91]:
```
df = pd.DataFrame([[image.shape[0],image.shape[1]] for image in wc_list])
df = pd.concat([df, pd.DataFrame([[image.shape[0],image.shape[1]] for image in␣
 ↪nw_list])])
df.columns = ["R", "C"]
df
```

[91]:

	R	C
0	6528	4896
1	5184	3880
2	3264	2448
3	5184	3880
4	5184	3880

```
..    …   …
108  6528  4896
109  3264  2448
110  4032  1860
111  3264  2448
112  3264  2448

[265 rows x 2 columns]
```

[105]:
```
df1 = df.groupby(['R','C']).size().reset_index().rename(columns={0:'count'})
df1.sort_values(by = ["count"], ascending=False).reset_index(drop = True)
```

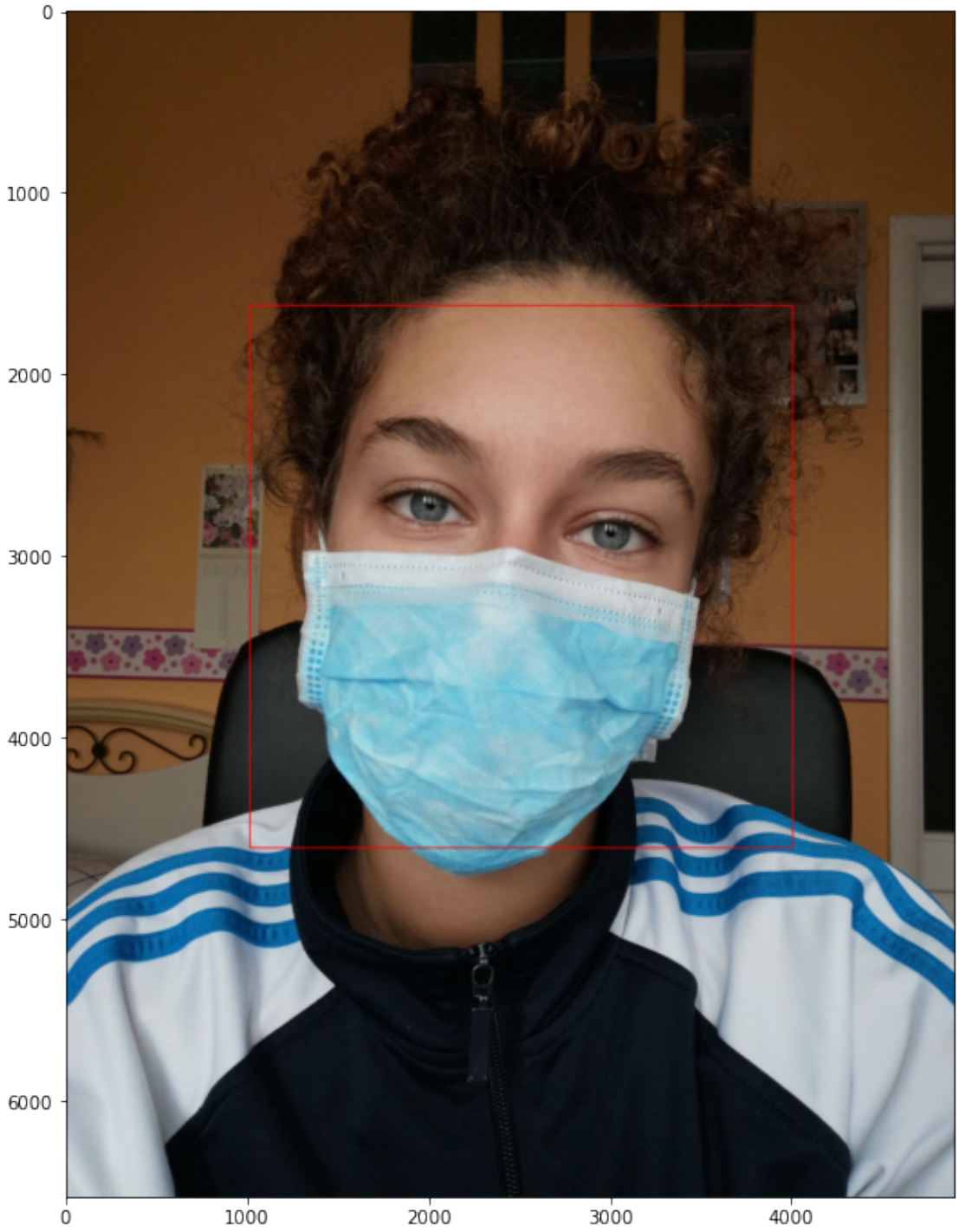
[105]:

	R	C	count
0	3264	2448	50
1	5184	3880	30
2	4608	3456	26
3	4624	3472	20
4	1280	720	19
5	3456	5184	18
6	6528	4896	12
7	3088	2320	10
8	1920	1080	9
9	4032	1860	9
10	1280	960	8
11	4224	3136	6
12	3264	1588	6
13	640	480	6
14	3000	3000	6
15	2944	2208	6
16	3264	1472	4
17	2576	1932	2
18	2320	3088	2
19	3968	2976	2
20	2032	1080	2
21	4032	3024	2
22	4128	3096	2
23	4160	3120	2
24	1584	1176	2
25	1280	958	2
26	1600	1200	1
27	1600	900	1

0.1 Face Detection

[79]:
```
haar_cascade_face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

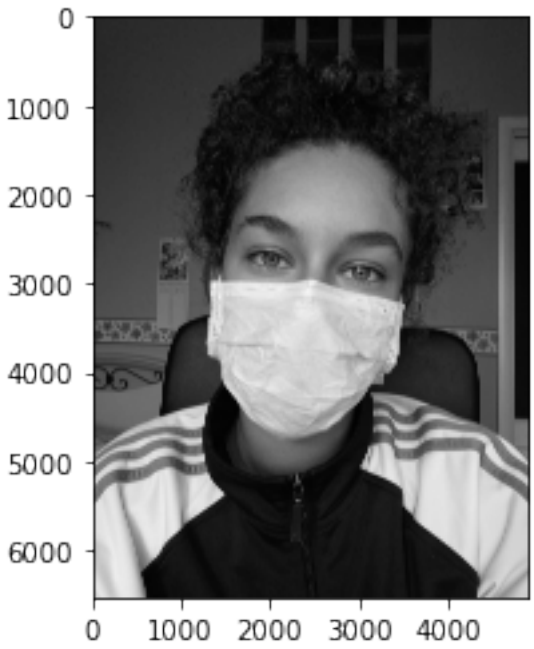
[46]:
```
faces = haar_cascade_face.detectMultiScale(wc_list[0],scaleFactor=1.1,␣
 ↪minNeighbors=4) #returns a list of (x,y,w,h) tuples

out_img = cv2.cvtColor(wc_list[0], cv2.COLOR_RGB2BGR) #colored output image

#plotting
for (x,y,w,h) in faces:
    cv2.rectangle(out_img,(x,y),(x+w,y+h),(255, 0, 0),5)
plt.figure(figsize=(12,12))
plt.imshow(out_img)
plt.show()
```



[47]:
```
faces
```

[47]:
```
array([[1012, 1622, 2985, 2985]], dtype=int32)
```

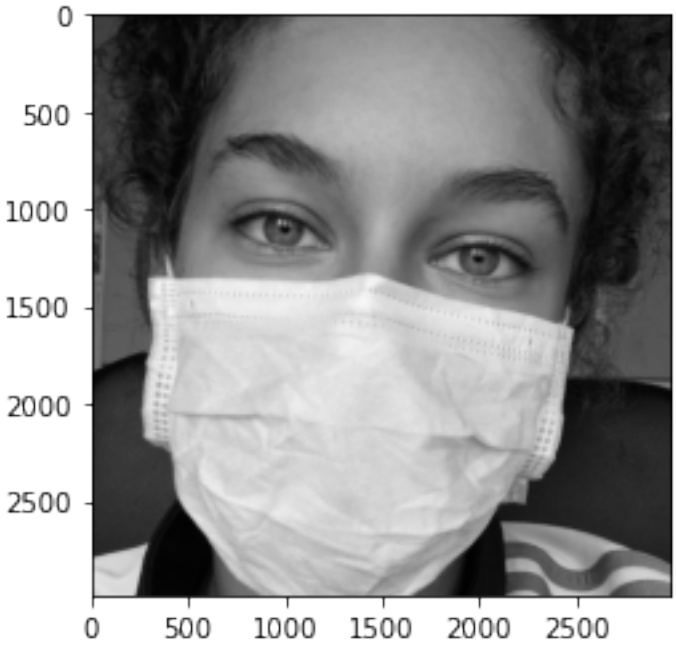
[33]:
```
img = cv2.cvtColor(wc_list[0], cv2.COLOR_RGB2GRAY) #colored output image
plt.imshow(img, cmap='gray')
plt.show()
```



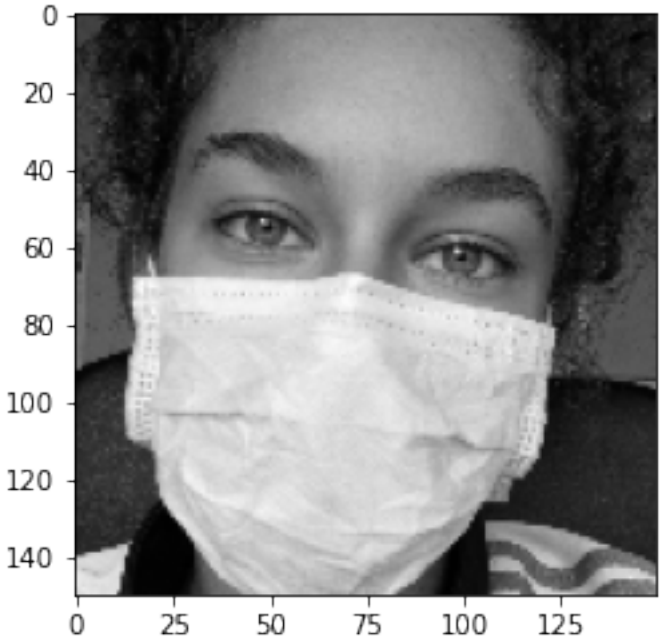
[29]:
```
img.shape
```

[29]:
```
(6528, 4896)
```

[34]:
```
img2 = img[y:y+h, x:x+w]
plt.imshow(img2, cmap='gray')
plt.show()
```



[35]:
```
img3 = cv2.resize(img2, (150, 150))
plt.imshow(img3, cmap='gray')
plt.show()
```

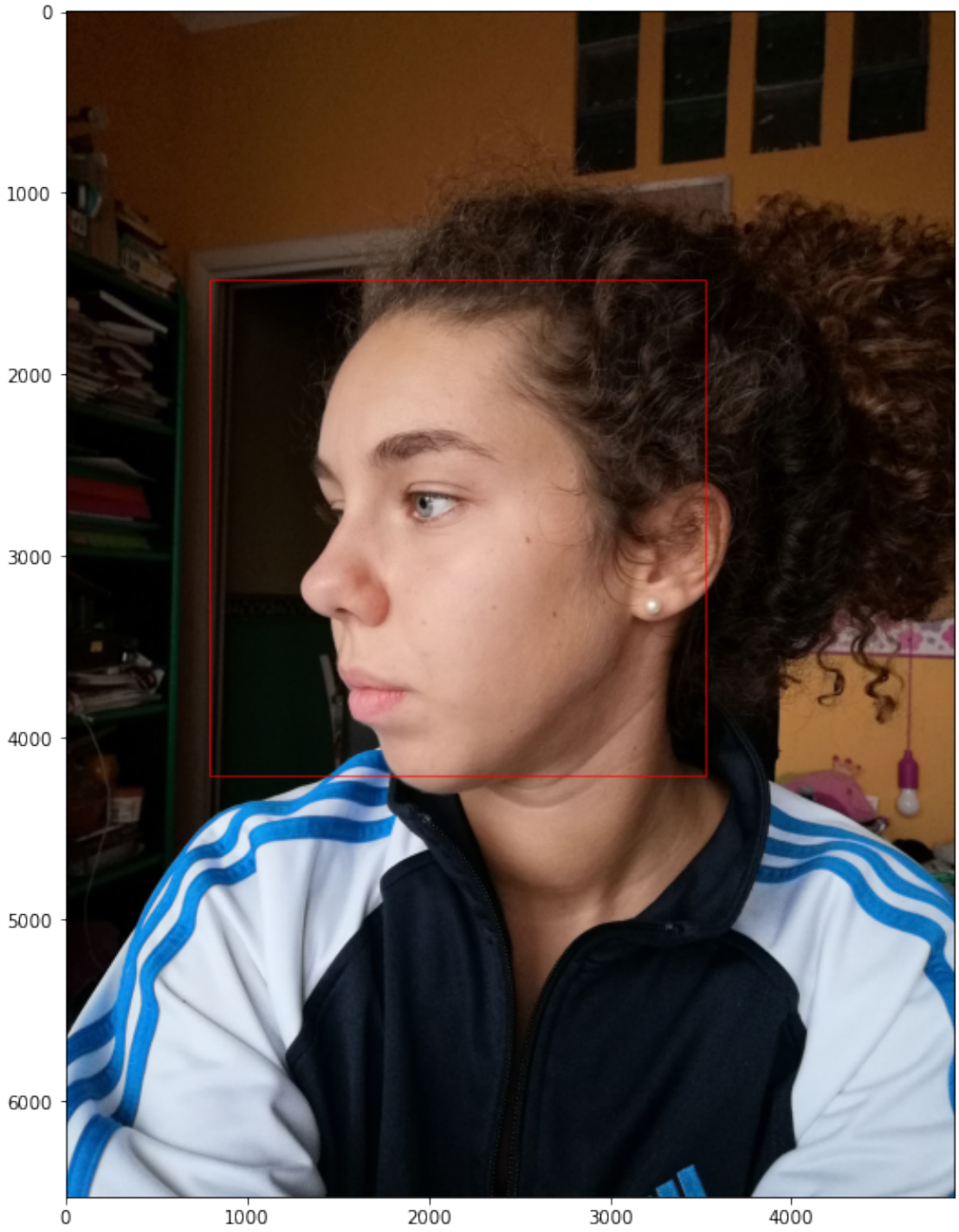


[61]:
```
img = cv2.imread("profile.jpeg")
plt.imshow(img)
plt.show()
```

[ ]:
```
haar_cascade_profile = cv2.CascadeClassifier('haarcascade_profileface.xml')
```

[75]:
```
plt.imshow(nw_list[2])
plt.show()
```

[72]:
```
profile = cv2.flip(nw_list[2], 1)
faces = haar_cascade_profile.detectMultiScale(profile,scaleFactor=1.1,␣
 ↪minNeighbors=4)

out_img = cv2.cvtColor(profile, cv2.COLOR_RGB2BGR) #colored output image

#plotting
for (x,y,w,h) in faces:
    cv2.rectangle(out_img,(x,y),(x+w,y+h),(255, 0, 0),5)
plt.figure(figsize=(12,12))
plt.imshow(out_img)
plt.show()
```

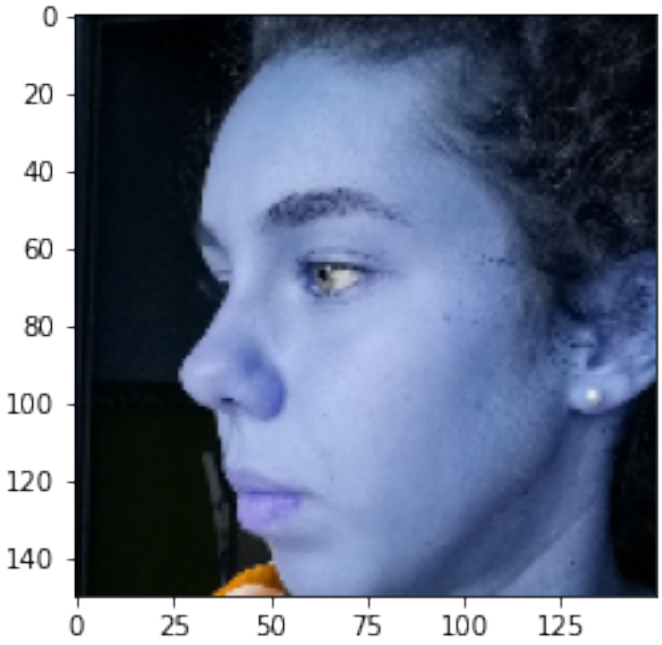


[83]:
```
def cropFace(cascade, image):

    for (x,y,w,h) in cascade:
```

```
        img = image[y:y+h, x:x+w, :]
        img = cv2.resize(img, (150, 150))
    return img


def isolateFace(image, plot = False):

    image_flipped = cv2.flip(image, 1)

    face = haar_cascade_face.detectMultiScale(image, scaleFactor=1.1,␣
 ↪minNeighbors=4)

    if len(face) == 1:

        return cropFace(face, image)

    else:

        profile = haar_cascade_profile.detectMultiScale(image, scaleFactor=1.1,␣
 ↪minNeighbors=4)
        image_flipped = cv2.flip(image, 1)
        profile_flipped = haar_cascade_profile.detectMultiScale(image_flipped,␣
 ↪scaleFactor=1.1, minNeighbors=4)

        if len(profile) == 1:

            return cropFace(profile, image)

        elif len(profile_flipped) == 1:

            return cropFace(profile_flipped, image_flipped)
```

[86]:
```
a = isolateFace(nw_list[2])
```

[87]:
```
plt.imshow(a, cmap='gray')
plt.show()
```


0
20
40
60
80
100
120
140
0
25
50
75
100
125