Processing WWMR Pictures

The pictures are all rotated and are jpgs. They will be converted into pngs and placed in more helpful folders for our purposes. The rotation is sorted out by "imposing" the metadata onto the image. The metadata contains the orientation, but is not loaded or imposed by default.

In [11]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image, ImageOps
import os
import face_recognition
import shutil
```

In [ ]:
```
# show image from dataset
example_unrotated_fp = r'./test_imgs/test2.jpg'
img = mpimg.imread(example_unrotated_fp)
imgplot = plt.imshow(img)
```

The rotation messes with the face detect algorithm, and would not work if mixed with pictures from different datasets.

Below is an example of what we will do for all the images.

In [ ]:
```
# rotate image and save it
fp = r'./test_imgs/test2.jpg'
out_fp = r'./test_imgs/test2_rot.png'

img = Image.open(fp)
img = ImageOps.exif_transpose(img)

#img.show()  # opens the image in the system default image viewer
img.save(out_fp)
```

In [ ]:
```
# display saved image
img = mpimg.imread(out_fp)
imgplot = plt.imshow(img)
```

In [2]:
```
# subjects labeled 1-16
subject_max_num = 16

correct_mask_folder_names = [
    'Mask Or Respirator Correctly Worn',
    'Mask Or Respirator On The Tip Of The Nose'  # the picture for these just look the same as the correct ones
]

incorrect_mask_folder_names = [
    'Mask Or Respirator Not Worn',
    'Mask Or Respirator Under The Nose',
    'Mask Or Respirator Under The Chin',
    'Mask Or Respirator On The Forehead',
    'Mask Or Respirator Hanging From An Ear',
    'Mask Folded Above The Chin'
]

# not needed for the file extraction planned
"""
mask_types = [
    'Disposable Respirator With Valve',
    'Disposable Respirator Without Valve',
    'Non-Medical Mask',
    'Surgical Mask'
]
"""
print()  # gets rid of printing block comment
```

In [ ]:
```
def extract_correct_from_pose_dirs(pose_dir, correct):
    # enter mask type directories
    for mt_root, mt_subdirectories, mt_files in os.walk(pose_dir):
        for mt_subdirectory in mt_subdirectories:
            mt_dir = os.path.join(pose_dir, mt_subdirectory)

            for pic_root, pic_subdirectories, pic_files in os.walk(mt_dir):
                for pic in pic_files:
                    pic_path = os.path.join(mt_dir, pic)

                    img = Image.open(pic_path)
                    img = ImageOps.exif_transpose(img)
                    #img = img.rotate(90, expand=True)
                    #img.show()  # opens the image in the system default image viewer
                    if correct:
                        out_fp = out_correct_root + str(counter) + '.png'
                    else:
                        out_fp = out_incorrect_root + str(counter) + '.png'

                    img.save(out_fp)
```

In [3]:
```
data_root_directory = r'D:\data\face_mask\WAYS TO WEAR A MASK OR A RESPIRATOR\WWMR-DB - Part 1\\'
counter = 0
out_correct_root =  r'D:\data\face_mask\test\correct\\'
out_incorrect_root =  r'D:\data\face_mask\test\incorrect\\'


# get subject folder names
subject_folder_names = []
for i in range(1, subject_max_num+1):
    subject_folder_names.append('Subject {}'.format(i))

for root, subdirectories, files in os.walk(data_root_directory):
    # filter paths that actually have files, not just directories
    if len(files) > 0:

        # verify path
        is_correct = sum([x in root for x in correct_mask_folder_names]) > 0
        is_incorrect = sum([x in root for x in incorrect_mask_folder_names]) > 0
        if (not is_correct and not is_incorrect) or is_correct and is_incorrect:  # pretty sure an xor cleans this up
            print('error, unexpected files in {}'.format(root))
        else:
            if is_correct:
                for file in files:
                    pic_fp = os.path.join(root, file)
                    img = Image.open(pic_fp)
                    img = ImageOps.exif_transpose(img)
                    out_fp = out_correct_root + str(counter) + '.png'
```

```
                img.save(out_fp)
                counter += 1
        else:
            for file in files:
                pic_fp = os.path.join(root, file)
                img = Image.open(pic_fp)
                img = ImageOps.exif_transpose(img)
                out_fp = out_incorrect_root + str(counter) + '.png'
                img.save(out_fp)
                counter += 1
        print('processed: {}'.format(counter))
```

```
processed: 3
processed: 6
processed: 9
processed: 12
processed: 15
processed: 18
processed: 23
processed: 28
processed: 33
processed: 38
processed: 41
processed: 44
processed: 47
processed: 50
processed: 53
processed: 56
processed: 59
processed: 62
processed: 65
processed: 68
processed: 71
processed: 74
processed: 77
processed: 80
processed: 83
processed: 86
processed: 89
processed: 92
processed: 95
processed: 100
processed: 103
processed: 106
processed: 109
processed: 112
processed: 115
processed: 116
processed: 117
processed: 118
processed: 119
processed: 120
processed: 121
processed: 122
processed: 124
processed: 126
processed: 129
processed: 131
processed: 133
processed: 135
processed: 137
processed: 139
processed: 142
processed: 145
processed: 150
processed: 153
processed: 156
processed: 159
processed: 162
processed: 165
processed: 168
processed: 171
```

```
processed: 176
processed: 179
processed: 182
processed: 185
processed: 188
processed: 191
processed: 194
processed: 197
processed: 200
processed: 205
processed: 210
processed: 216
processed: 219
processed: 222
processed: 225
processed: 228
processed: 231
processed: 234
processed: 237
processed: 240
processed: 243
processed: 246
processed: 250
processed: 253
processed: 256
processed: 259
processed: 262
processed: 265
processed: 268
processed: 271
processed: 274
processed: 277
processed: 282
processed: 287
processed: 290
processed: 293
processed: 296
processed: 299
processed: 302
processed: 305
processed: 308
processed: 311
processed: 314
processed: 317
processed: 320
processed: 325
processed: 328
processed: 331
processed: 334
processed: 337
processed: 340
processed: 343
processed: 346
processed: 349
processed: 352
processed: 355
processed: 358
processed: 363
processed: 368
processed: 373
```

```
processed: 378
processed: 381
processed: 384
processed: 387
processed: 390
processed: 393
processed: 396
processed: 399
processed: 402
processed: 405
processed: 408
processed: 411
processed: 414
processed: 417
processed: 420
processed: 423
processed: 426
processed: 429
processed: 431
processed: 434
processed: 439
processed: 442
processed: 445
processed: 448
processed: 451
processed: 454
processed: 457
processed: 460
processed: 463
processed: 468
processed: 473
processed: 476
processed: 479
processed: 482
processed: 485
processed: 488
processed: 491
processed: 494
processed: 497
processed: 500
processed: 501
processed: 502
processed: 503
processed: 504
processed: 505
processed: 506
processed: 507
processed: 508
processed: 511
processed: 514
processed: 519
processed: 522
processed: 525
processed: 528
processed: 531
processed: 534
processed: 537
processed: 540
processed: 545
processed: 548
```

```
processed: 551

processed: 554

processed: 557

processed: 560
```

Now, since we need to draw bounding boxes, we need to see how many of these faces can be recognized by off the shelf face detection.

In [12]:
```
correct_root =  r'D:\data\face_mask\test\correct\\'
incorrect_root =  r'D:\data\face_mask\test\incorrect\\'

out_detectedface_correct_root =  r'D:\data\face_mask\test\detected correct\\'
out_detectedface_incorrect_root =  r'D:\data\face_mask\test\detected incorrect\\'


for root, subdirectories, files in os.walk(correct_root):
    for f in files:

        og_img_path = os.path.join(root, f)
        img = face_recognition.load_image_file(og_img_path)

        face_locs = face_recognition.face_locations(img)

        # if a face is detected
        if len(face_locs) > 0:
            # then put it in the detected folder
            out_fp = os.path.join(out_detectedface_correct_root, f)
            shutil.copyfile(og_img_path, out_fp)
```

In [13]:
```
for root, subdirectories, files in os.walk(incorrect_root):
    for f in files:

        og_img_path = os.path.join(root, f)
        img = face_recognition.load_image_file(og_img_path)

        face_locs = face_recognition.face_locations(img)

        # if a face is detected
        if len(face_locs) > 0:
            # then put it in the detected folder
            out_fp = os.path.join(out_detectedface_incorrect_root, f)
            shutil.copyfile(og_img_path, out_fp)
```

In [ ]: