avatar_face_recognition

Neelkuamr Mistry
ID 214371306

Contents

Chapter 1

Introduction

This face recognition and mask recognition project is part of a parent project which includes developing a Robotic Avatar. For the Avatar to interact efficiently with its users, it requires various informational input such as who it is interacting with, gender, age, distance/direction of person from robot, emotion, etc. This project code was developed to fulfil couple of those input requirements: who the person is and if they are wearing a mask or not. Mask detection was not a part of scope to start with. However, current COVID-19 pandemic situation fueled the idea of inclusion of mask detection module into the project scope.

Below is a picture of actual Robot which the team is working with.

There are 3 main modules of this projects which are somewhat interdependent.

1. Face Detection: In order to recognize the name of user or if user is wearing a mask or not, we first need to detect faces in the image frames.
2. Face Recognition: Once the faces are detected, we search through our existing face database to see if the face(s) in frame matches with any other face in database. If we find matching face, then we recognize the person. If not, we call the person "Unknown".
3. Mask Detection: In this module, we simply loop through all the faces in frame and identify if user(s) are wearing a mask or not. This module also calculates the percentage confidence Avatar has for each face.

Initial project was solely based on python. As the project evolved, team decided to use ROS (Robot Operating System) environment for this project.

1.1 Basic about ROS

ROS is an open-source, meta-operating system for robot. For our purpose, it provides us the modularity we require and an ability to process, send and receive the various forms of information (i.e., compressed image, string, integer, etc.) between different modules.

ROS has numerous functionalities. However, we mostly worked with nodes, topics and (custom) messages.

- Nodes: Nodes are written in python script and they perform some actions on the input information and produces an output.
- Topics: Topics are, hypothetically, places where input/output is published.
- Messages: ROS Messages are very similar to our day to day cellphone messages. We use them to send/receive the information between nodes and topics. ROS has a library of standard messages (string, integer, etc.) but we can build custom messages if we wish. We are building a custom message for the purpose of this project.

The relation between Nodes and Topics is "Many-to-Many" – which means any number of nodes can publish/subscribe to any number of topics, and vice versa.

Chapter 2

Project Basics

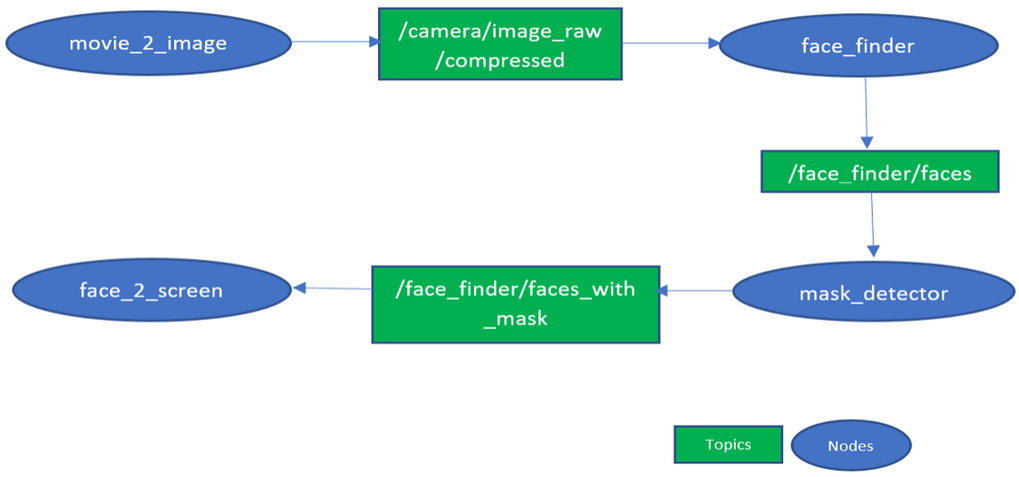
For face detection, we have 2 nodes – one is based on haar cascade and another is based on pre-trained Neural-Net. Why do we have two? Initially, we used haar cascade to detect faces in image frames. (Image frames can be generated from live webcam or pre-recorded video.) However, when mask detection was added to the scope of the project, we noticed that, most of the times, haar cascade fails to detect faces with mask. To address this, we decided to use a pre-trained Neural-Net already available on the internet. This Neural-Net is trained on faces with mask and without mask.

Mask detection introduced a unique challenge for face recognition. The existing methodologies for face recognition are built to work with faces without mask only. As mask covers some of the facial features, it is difficult to correctly (or confidently) identify faces. We could think of one way to possibly overcome this challenge. For each person in the face database, we store face images with mask and without mask. This technique will still not be compensating the loss of facial features incurred by mask covering. But something is better than nothing! By the way, we are using dlib's state-of-art face recognition here.

We mentioned "training data-set/database" couple of times before. What exactly it is? This is a ".pickle" file which includes the facial encoding of all the faces known to avatar. The expectation from Avatar is to build on this database as it gets introduced to more people. Each time a new face is found in the video stream, Avatar compares that face with these existing encoding to see if it knows the person. At this point, you could understand the importance of our training data-set accuracy. We have a training node (rebuild_database.py) dedicated to build/append new data to the training data-set. Ideally, we do not want to miss a single face in any image we use to train our Avatar on. That is why, we used "num_jitters" with our face_recognition in out rebuild_database node (training node). This parameter indicates how many times to re-sample the face when calculating encoding. Higher jitter value is more accurate, but slower. This does not guarantee 100% accuracy, but it does make a difference.

Chapter 3

Ground Level ROS Structure



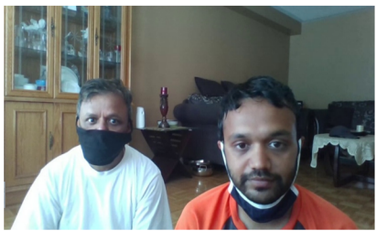
The figure above is a pictorial representation of high-level working structure of this package. Ros-nodes are represented using blue ellipse and ros-topics are represented using green rectangles. Arrows between them represents the message (data) flow. Node movie_2_image takes a video input either from cam_2_image node or movie_2_image node. (node cam_2_image grabs image frames from mounted webcam and node movie_2_image grabs image frames from a movie file.) The figure shows how each node subscribes to a topic to get input, processes the information, and publish it to an appropriate topic. Node face_finder identifies faces in image frames and recognizes them. Node Mask_detector detects if people in the frames are wearing a mask. Finally, node face_2_screen displays the output to the screen.

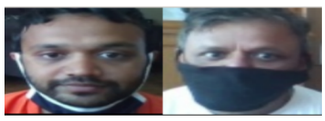
Input: Video containing faces (Live or pre-recorded)

Output:

- Individual face images of all people in frame
- Dimensions and locations of faces in frame
- Json string containing recognized names, mask detection info with percentage confidence

Below is a sample input/output representation.





Json: {'name': 'Neel Mistry', 'maskVal': 'No-Mask 92.10%'},

{'name': 'Nitesh Mistry', 'maskVal': 'Mask 98.30%'}

Locations: [(78, 92, 131, 158), (235, 88, 306, 174)]

Chapter 4

Test Description

To test the accuracy of all the modules we have, we conducted few tests. We asked the project team to make short videos of themselves putting on and taking off face masks. Some of those videos also had an unknown person in the frame. Next, using ffmpeg tool, we saved took frames of each individual's video and saved them on computer as an image file. These images are then divided into three different data-sets.

Training Data-set: We selected around 10% images with different angles of faces, with mask and without mask. This data-set is used to train our Avatar.

Validation Data-set: Another 10% selected images were used for Validation Data-set. Once our Avatar is trained on Training Data-set images, we repeatedly ran face recognition and mask detection on validation data-set images and fine-tuned the hyper-parameters to achieve best accuracy.

Test Data-set: Remaining approx. 80% images are the ones we used to test accuracy of this project. Once we found (possibly) best hyper-parameters for Avatar using validation data-set images, we ran Avatar on test data-set images and recorded the accuracy results for face recognition and mask detection.

We also performed some other tests which are described briefly in results section.

Chapter 5

Results

Below are the results of the tests conducted.

5.1 Face-Recognition [1]

Individual videos people where they are taking off/putting on mask after every few seconds

Confusion Matrix - Face Recognition

True Values		Predicted Values									
		Abdulla	Enas	Deeksha	Jenkin	Nav	Neel	Nilam	Mihir	Unknown	No-faces
n = 100 Frames Per Video	Abdulla	56	0	5	0	1	3	0	0	9	26
	Enas	0	51	4	1	0	2	6	1	13	22
	Deeksha	3	2	43	1	3	4	2	2	16	24
	Jenkin	0	1	0	60	2	5	0	1	12	19
	Nav	3	1	0	1	39	13	0	1	24	18
	Neel	0	0	1	1	4	63	2	3	17	9
	Nilam	4	2	7	0	0	1	50	4	12	20
	Mihir	1	0	2	1	0	1	0	66	14	15
	Unknown	-	-	-	-	-	-	-	-	-	-
	No-faces	-	-	-	-	-	-	-	-	-	-

Individual videos of Neel and Nabh never wearing mask

Confusion Matrix - Face Recognition

True Values	Abdulla	Enas	Helio	Jenkin	Nabh	Neel	Nitesh	Shahid	Tim	Unknown
Abdulla										
Enas										
Helio										
Jenkin										
Nabh	2	0	1	3	24	15	1	1	0	3
Neel	5	0	0	2	6	30	2	0	0	5
Nitesh										
Shahid										
Tim										
Unknown										

n = 50 Frames

Predicted Values

Individual videos of Neel and Nabh always wearing mask

Confusion Matrix - Face Recognition

True Values	Abdulla	Enas	Helio	Jenkin	Nabh	Neel	Nitesh	Shahid	Tim	Unknown
Abdulla										
Enas										
Helio										
Jenkin										
Nabh	7	2	0	4	15	11	4	2	0	5
Neel	8	0	0	3	17	14	2	0	0	6
Nitesh										
Shahid										
Tim										
Unknown										

n = 50 Frames

Predicted Values

5.2 Face-Detection

Face detection in empty video

# of frames	Faces Detected	No faces detected
100 (5th frame out of 500 total)	2	98

Face detection in video with 1 person - always wearing mask

# of frames	Faces Detected	No faces detected
50 (10th frame out of 500 total)	42	8

Face detection in video with 1 person - never wearing mask

# of frames	Faces Detected	No faces detected
50 (10th frame out of 500 total)	49	1

Face detection in video with 7 faces - None wearing mask

Detected # of faces	7	6	5	4	3	2	1	0
In # of frames (Total = 50)	23	11	12	3	1	0	0	0

5.3 Mask-Detection

Mask Detection in video with 1 person - wearing mask all time

# of frames	Mask detected	Mask not detected
50 (10th frame out of 500 total)	49	1

Mask detection in video with 1 person - never wearing mask

# of frames	Mask detected	Mask not detected
50 (10th frame out of 500 total)	1	49

Chapter 6

Accuracy Calculations

-Face Recognition (combination of mask on/off) = (56+51+43+60+39+63+50+66)/800 = 53.5 %

-Face Recognition (Mask off) = (24+30)/100 = 54 %

-Face Recognition (Mask on) = (15+14)/100 = 29 %

-Face Detection (mask on) = 84 %

-Face Detection (mask off) = 98 %

-Mask Detection = 98 %

Chapter 7

Analysis & Conclusion

From the results above, we can conclude that our mask detection works with approximately 98% accuracy, which is impressive! The confidence percentage with each detection varies though, which is obvious, and we care less about that.

Face detection accuracy has quite a bit difference when we deal with masked vs non-masked faces(around 14%).

Face recognition with combination of mask on/off has an accuracy of approximately 53.5% percentage. While testing, we realized that most mis-identifications took place when the person was having mask on. The training data-set had 94 images in total and it detected faces in 78 of them. 13 of 16 images, where it could not detect face, were with mask on. So, it is comparatively more accurate working with faces without mask. Accuracy is affected by how many training images we had for each person with mask and without mask. Generally, the accuracy percentage improves as we provide more images of each person in the data-set. Another factors affecting accuracy are face positions in images, image size, face angles, lighting, etc. Our future steps include working on most of these factors to achieve desired accuracy.

Chapter 8

Improvement Opportunities

Current face-detection and recognition accuracy can be improved by performing some data augmentation task such as image resizing, image warping, face rotation in images, and other pre-processing. The training database size also impacts the accuracy.

Even after all, if we still struggle to achieve face-recognition accuracy with mask on, then Robot can be programmed to ask the masked users to take off the mask momentarily so that it can recognize him/her.

[2]

Bibliography

[1] URL: http://books.google.com/books?id=W-xMPgAACAAJ.

[2] Adrian Rosebrock. URL: https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/.