

**FACE EXPRESSION DETECTION USING OPENCV**

A. Sravya<sup>1</sup>, M. Bhavana<sup>2</sup>, Dr. V. Sireesha<sup>3</sup>

<sup>1</sup>Dept of CSE, Vasavi College Of Engineering, Hyderabad, Telangana, India

<sup>2</sup>Dept of CSE, Vasavi College Of Engineering, Hyderabad, Telangana, India,

<sup>1</sup>Dept of CSE, Vasavi College Of Engineering, Hyderabad, Telangana, India,

**Abstract**— Computer vision is a field which teaches computers to gather information they see. It is a way computers learn from their environment and meet our demands. In the past few years face recognition has become popular and is being extensively used in various computer vision fields. This has motivated researches and neuroscientists to dig more on expression detection since it has many applications in automatic access control system. In this work, we detect user’s emotions using his facial expressions. The expressions are predicted from the pre-existing image available in the memory and also real time. In order to build a face expression detection system, we scan the training dataset and compare it with the test data which helps the classifier to predict the expression. This implementation is done using Haar-cascade algorithm in order to detect the face. Due to its efficiency, Haar-like rectangle features play an important role in face detection. Also, an algorithm known as fisher face is used to detect the expression from the given dataset loaded by the user. These algorithms are provided by OpenCV which is an open source computer vision library with various kinds of algorithms built in. It is applicable in various computer vision projects. This work has been implemented using Python Idle 2.7, Open Source Computer Vision Library (OpenCV) and NumPy.

**Keywords**— Haar cascade, Fisherface, OpenCV, Numpy, Adaboost.

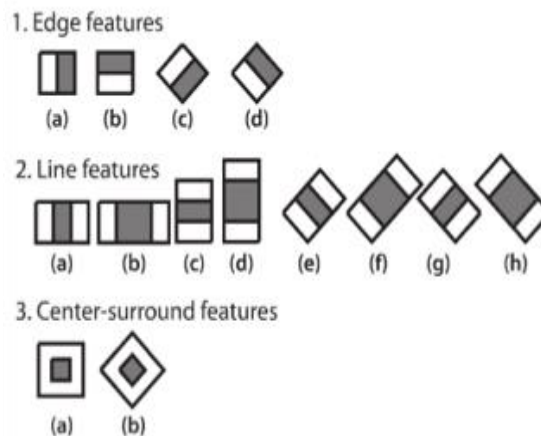
**I. INTRODUCTION**

Image processing is extensively applied in order to get an enhanced image and to extract some useful information out of it. An efficient way of converting an image into a digital form subsequently performing various operations on it is image processing. It is similar to digital signal processing where a 2D image is taken as an input, in which the pixel values range from 0 to 255.

**II. ALGORITHM SPECIFIC TO FACE EXPRESSION**

**A. Haar Cascade Algorithm**

Haar cascade is based on Haar Wavelet[8] technique to analyze pixels in the image and compute them into squares by function. A training dataset is used which is a machine learning method of training the classifier. This process involves integral images which tells us the faces detected. Haar Cascade[1] is based on **Adaboost** learning algorithm[8]. It selects small number of features from a large number and gives us an efficient classifier.



**Fig.1 Feature Extraction**

**Fisher Face Algorithm**

One way to represent the input data is by finding a subspace which represents most of the data variance. This can be obtained with the use of Principal Components Analysis (PCA). When applied to face images, PCA yields a set of eigenfaces[2]. Subspace representation is a set of face images, that results in basis vectors defining the space known as Fisherfaces.

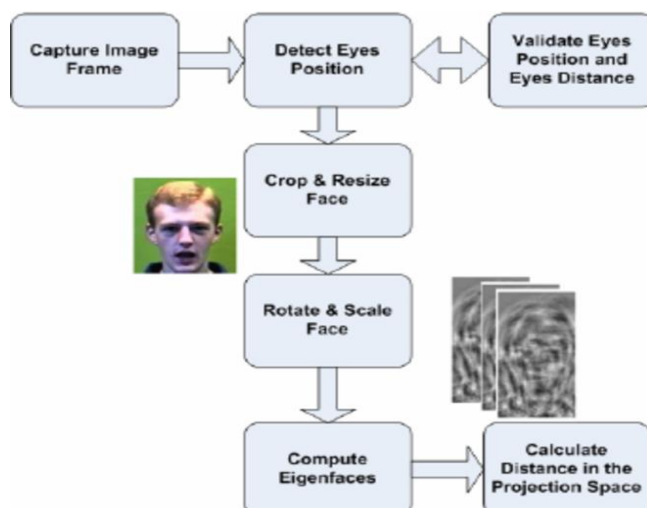


Fig.2 Workflow diagram of face expression

## 2.1 Introduction to OpenCV & Numpy

**OpenCV:** OpenCV is an Open Computer Vision Library[3]. It is portable on platforms like digital signal processors. It is a tool which reduces number of lines, errors, and also avoid memory leakage. OpenCV is a trainer and detector. It contains pre defined classifier for face, eyes, smile, etc., Various features like GPU acceleration[4] have been added to the existing libraries which enhances the tool's features.

**Numpy:** NumPy is a library for Python programming language. It provides large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It is an open-source software that has an array object of N-dimension. It is used to integrate C/C++ and Fortran code for solving linear algebraic expressions, and Fourier transforms[7].

**Example:** >>> import numpy as np  
 >>> a = np.arange(15).reshape(3, 5)  
 >>> aarray([[ 0, 1, 2, 3, 4], [ 5, 6, 7, 8, 9], [10, 11, 12, 13, 14]])  
 >>> a.shape  
 (3, 5)  
 >>> a.ndim  
 2  
 >>> a.dtype.name  
 'int64'

**The types of expressions detected are:**

- A. Neutral
- B. Happy
- C. Anger
- D. Disgust
- E. Surprise
- F. Fear
- G. Sad
- H. Contempt

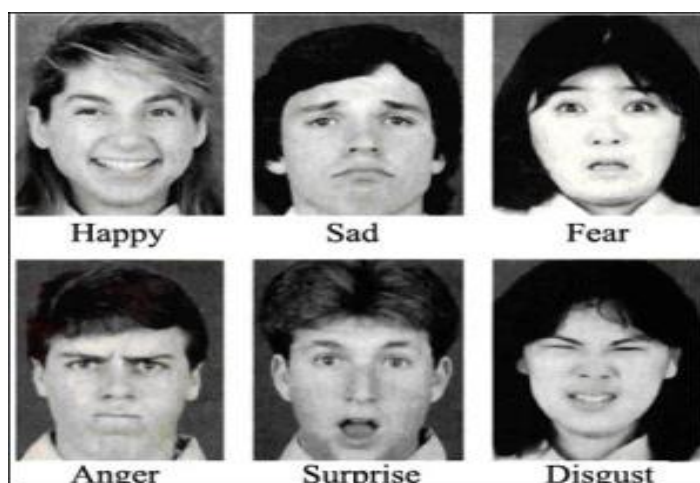


Fig.3 Cohn Kanade dataset

### III. STEPS IMPLEMENTED TO PERFORM EXPRESSION DETECTION USING OPENCV

1. After installing Open CV and Python Idle 2.7, we must create the dataset. Here, we create a dataset by analysing group of images so that our result is accurate and there is enough data to extract sufficient information.
2. The database is then classified into two different directories. First directory contains the images and the second directory contains information about different types of expressions.
3. Different types of classes can be used in OpenCV for emotion recognition, but we will be mainly using Fisher Face. `FisherfaceRecognizer= createFisherFaceRecognizer()`.
4. Now, we extract faces using OenCV which provides classifiers that are responsible for detecting faces. `face_detector= cv2.CascadeClassifier("haarcascade_frontalface_default.xml")`
5. The dataset is split into Training set and Classification set. The training set is used to teach the type of emotions by extracting information from a number of images and the classification set is used to estimate the classifier performance. The images should be of same size for better results.
6. Each image's subject is analyzed, converted to grayscale, cropped and saved to a directory. Conversion of Color Image to Gray Scale:

There are basically two methods to convert a color image to a gray scale image [5]:

#### A. Average Method

In Average method, mean of RGB(Red, Blue, Green)colors in a present color image is calculated . We get,  $Grayscale = (R+G+B)/3$ ;

Sometimes instead of grayscale image we get black image. This is due to the converted image we get ie.,33% each of Red, Blue & Green. Therefore, to solve this problem we use the second method called Weighted Method or Luminosity Method.

#### B. Weighted or Luminosity Method

To limit the problem in Average Method, we use Luminosity method. In this method, we decrement the presence of Red Color and increment the color of Green Color and the blue color's percentage is in between these two colors.

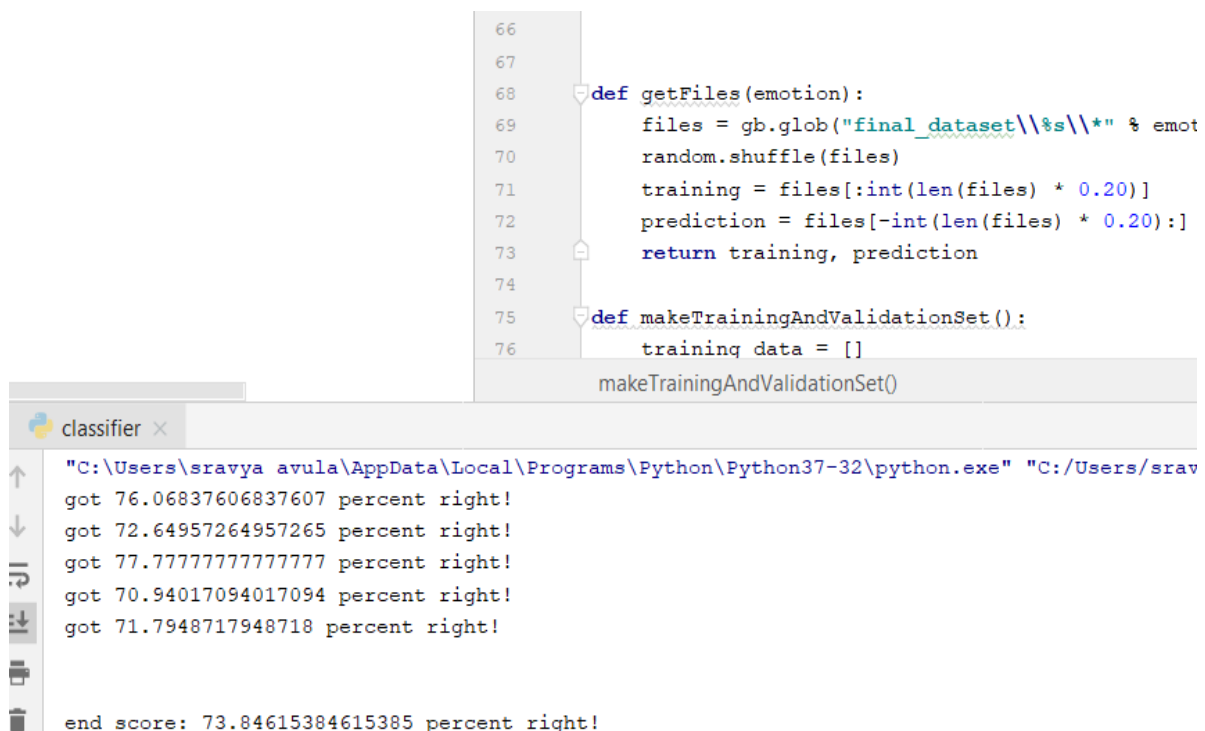
Thus, by the equation we get,  $Grayscale = ((0.3 * R) + (0.59 * G) + (0.11 * B))$ .

We use this because of the wavelength patterns of these colors. Blue has the least wavelength while Red has the maximum wavelength.

7. Finally, we compile training set as 80% of the test data and classify the remaining 20% on the classification set.

### IV. RESULTS

CASE 1: WHEN THE SIZE OF TRAINING DATA SET IS 20 % OF THE FINAL DATA SET WE GET:



```
66
67
68 def getFiles(emotion):
69     files = gb.glob("final_dataset\\%s\\*" % emot
70     random.shuffle(files)
71     training = files[:int(len(files) * 0.20)]
72     prediction = files[-int(len(files) * 0.20):]
73     return training, prediction
74
75 def makeTrainingAndValidationSet():
76     training data = []
    makeTrainingAndValidationSet()
```

```
classifier x
"C:\Users\sravya avula\AppData\Local\Programs\Python\Python37-32\python.exe" "C:/Users/srav
got 76.06837606837607 percent right!
got 72.64957264957265 percent right!
got 77.77777777777777 percent right!
got 70.94017094017094 percent right!
got 71.7948717948718 percent right!

end score: 73.84615384615385 percent right!
```

CASE 2 : WHEN THE SIZE OF TRAINING DATA SET IS 40 % OF THE FINAL DATA SET WE GET:

```
70     random.shuffle(files)
71     training = files[:int(len(files) * 0.40)]
72     prediction = files[-int(len(files) * 0.40):]
73     return training, prediction
74
75 def makeTrainingAndValidationSet():
76     training_data = []
    getFiles()
classifier x
"C:\Users\sravya avula\AppData\Local\Programs\Python\Python37-32\python.exe" "C:/Users/srav
got 71.84873949579831 percent right!
got 81.09243697478992 percent right!
got 81.5126050420168 percent right!
got 81.5126050420168 percent right!
got 83.19327731092437 percent right!

end score: 79.83193277310923 percent right!

Process finished with exit code 0
```

CASE 3 : WHEN THE SIZE OF TRAINING DATA SET IS 60 % OF THE FINAL DATA SET WE GET:

```
hk.jpg
hk2.jpg
img_seq.py
video_capture.py
ernal Libraries
atches and Consoles
72     training = files[:int(len(files) * 0.60)] #
73     prediction = files[-int(len(files) * 0.40):]
74     return training, prediction
75
76
77 def makeTrainingAndValidationSet():
78     training_data = []
79     training_labels = []
80     prediction_data = []
81     prediction_labels = []
classifier x
D:\forPro\MajorProject-1\ExpressionRecogniser\venv\Scripts\python.exe D:/forPro/MajorProjec
got 79.83193277310924 percent right!
got 85.71428571428571 percent right!
got 78.57142857142857 percent right!
got 81.9327731092437 percent right!
got 78.57142857142857 percent right!

end score: 80.92436974789916 percent right!
```

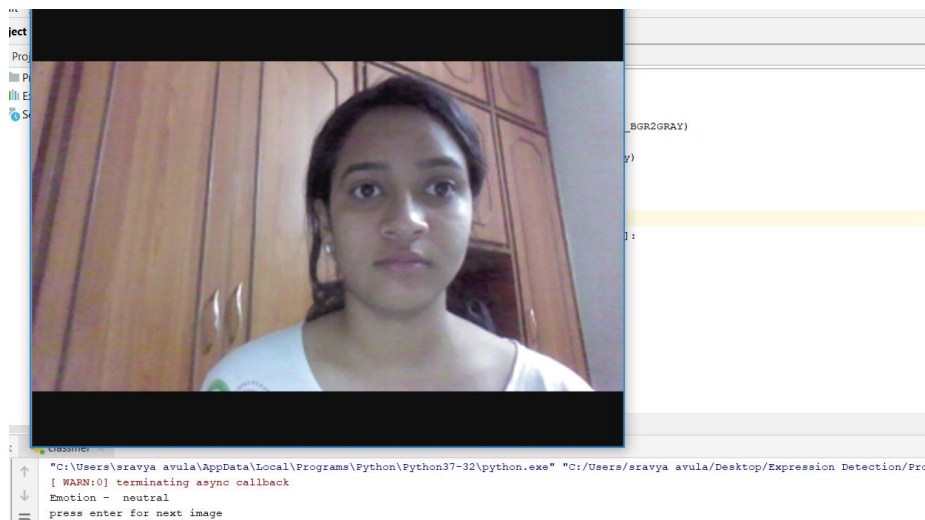
CASE 4 : WHEN THE SIZE OF TRAINING DATA SET IS 80 % OF THE FINAL DATA SET WE GET:

```
hk.jpg 72 training = files[:int(len(files) * 0.80)] #
hk2.jpg 73 prediction = files[-int(len(files) * 0.20):]
img_seq.py 74 return training, prediction
video_capture.py 75
ernal Libraries 76
atches and Consoles 77
78 def makeTrainingAndValidationSet():
79     training_data = []
80     training_labels = []
81     prediction_data = []
82     prediction_labels = []
83
84     getFiles()
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

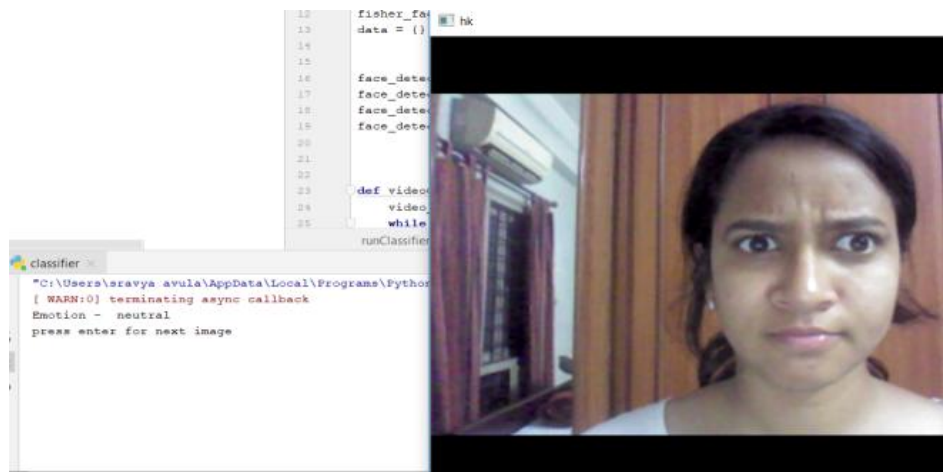
```
classifier x
D:\forPro\MajorProject-1\ExpressionRecogniser\venv\Scripts\python.exe D:/forPro/MajorProjec
got 82.90598290598291 percent right!
got 82.90598290598291 percent right!
got 82.90598290598291 percent right!
got 83.76068376068376 percent right!
got 82.90598290598291 percent right!

end score: 83.07692307692308 percent right!
```

REAL TIME EXPRESSION 1: NEUTRAL

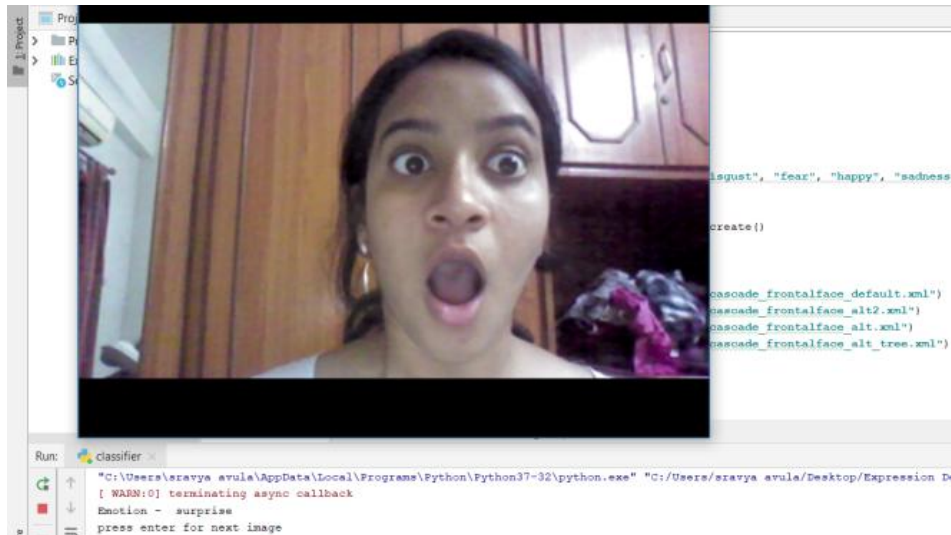


REAL TIME EXPRESSION 2: ANGER





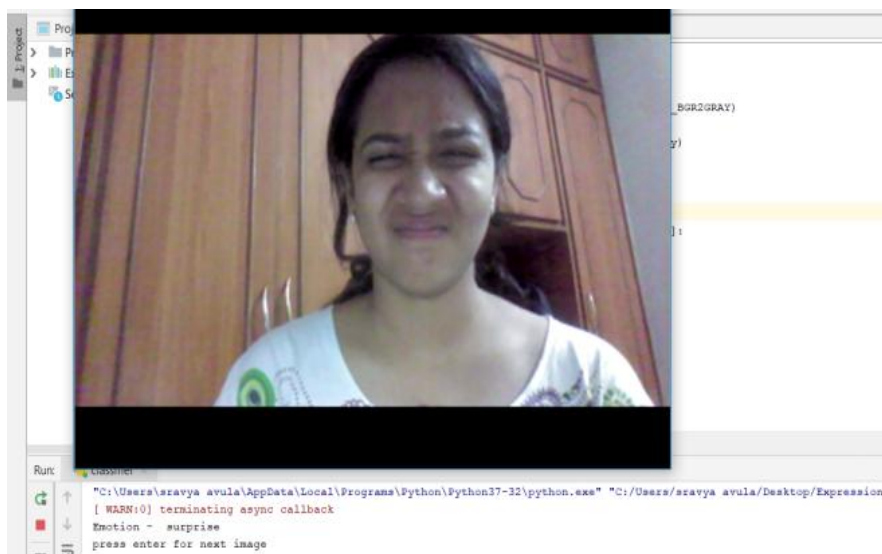
REAL TIME EXPRESSION 3: SURPRISE



REAL TIME EXPRESSION 4: CONTEMPT



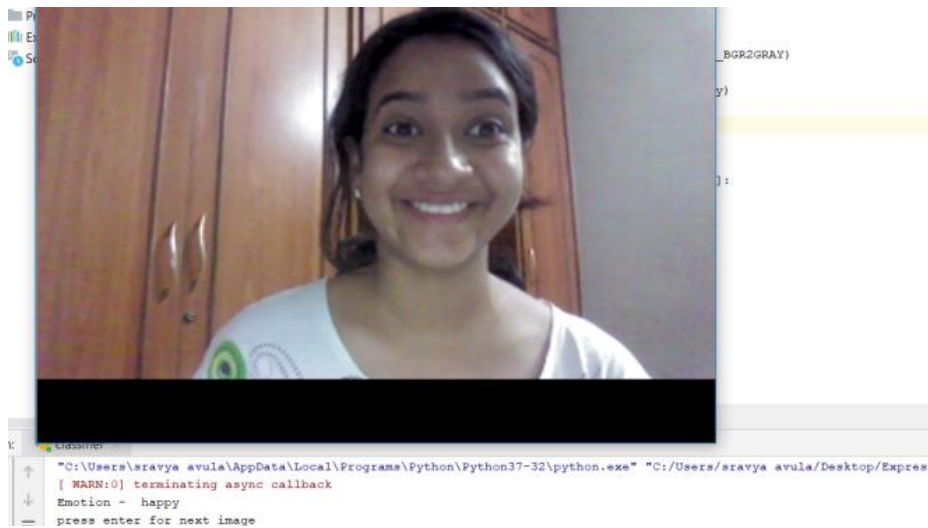
REAL TIME EXPRESSION 5: DISGUST



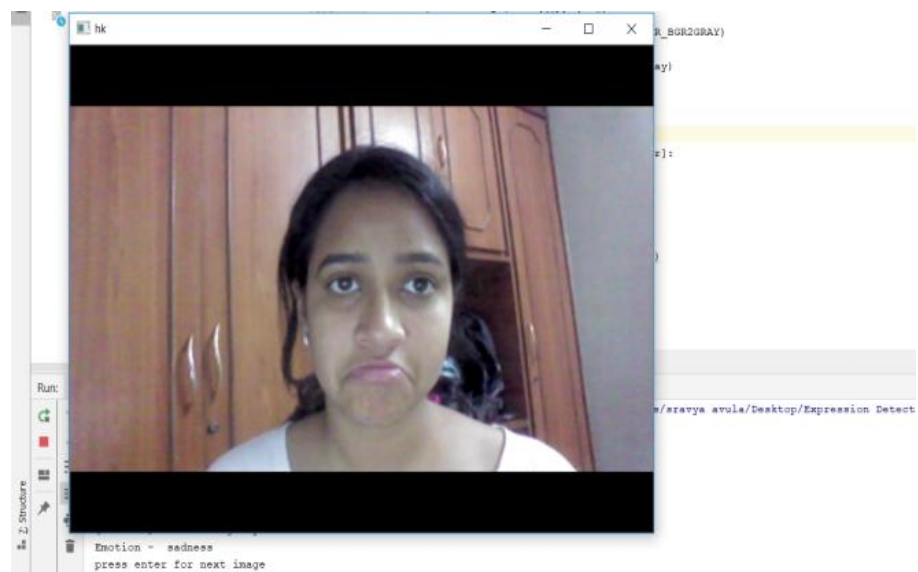
REAL TIME EXPRESSION 6: FEAR



REAL TIME EXPRESSION 7: HAPPY



REAL TIME EXPRESSION 8: SADNESS



**4.1 OBSERVATIONS**

**Table 1: Accuracy prediction for existing subjects**

<b>Training data set (as % of final dataset)</b>	<b>Prediction data set (as % of final dataset)</b>	<b>Accuracy of emotion detection</b>
<b>20</b>	20	73.85
<b>40</b>	40	79.83
<b>60</b>	40	80.92
<b>80</b>	20	83.08

**Table 2: Accuracy prediction for real time face subject**

**V. APPLICATIONS**

Loads of tasks and processes can be performed if one can become aware about the intricacies and endless possibilities offered under the field of emotion detection.

Some of the widely used applications are:

1. Improvised web development

Internet is expanding in a large scale and the service providers are interested in collecting tons of data from the users. Correspondingly, all the content and advertisements are played based on the users' profile. Subsequently, adding intricate details about the different human emotions can provide much more precise behavioral models of different types of users.

2. App and product development

In engineering processes, testing of a product can be done with ease by understanding the emotions. It's a long-established fact that level of comfort with different software products depends hugely upon human emotions. A products overall look and feel can also alter human feelings which in turn makes the person buy the product or not. Thus, researching about different emotional states of a human body and how it is influenced by usage of different products is a matter of prime importance for related industry professionals.

<b>Actual expression</b>	<b>Predicted Expression</b>
Neutral	Neutral
Anger	Neutral
surprise	Surprise
Contempt	Fear
Disgust	Surprise
Fear	Surprise
Happy	Happy
Sadness	Sadness
<b>Total Accuracy</b>	<b>57.14</b>

**VI. CONCLUSION AND FUTURE WORK**

Artificial Intelligence can be used to solve intriguing tasks such as emotion detection, although this task was quite convolute even more when using a great number of images.

We have developed a computer vision system that automatically recognizes facial expressions with subtle differences and also estimates expression intensity. Feature point tracking together with the principal component analysis, and high gradient component analysis, and is then used to discriminate subtle differences in facial expressions.

In Present Work, we recognized different types of human emotions using Python 2.7, Open CV & (CK and CK+) Database [6] and got some interesting insight about it.

For existing subjects of eight different expressions, the result was approximately 70-80% accurate.



The classifier has tried to learn the dataset and has given us some satisfying results. The results show that our system has achieved high accuracy in facial expression recognition.

We have implemented the same methodology for a real time subject with the same eight expressions and we have achieved an accuracy of 57.14%. We are going to push further to increase the efficiency and produce more accurate results. Emotion detection is an inseparable part of computer vision. Loads of tasks and processes can be performed if one can become aware about the intricacies and endless possibilities offered under the field of emotion detection.

## VII. REFERENCES

- [1] [https://docs.opencv.org/3.4.2/d7/d8b/tutorial\\_py\\_face\\_detection.html](https://docs.opencv.org/3.4.2/d7/d8b/tutorial_py_face_detection.html)  
<http://www.rroj.com/open-access/comparison-between-face-recognition-algorithmeigenfaces-fisherfaces-and-elastic-bnch-graph-matching-187-193.php?aid=37590J>
- [2] France,46-53- Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z.,& Matthews, I. (2010). The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, 94-101. Vision with OpenCV. *Queue* 10, 4, Pages \ Kari Pulli, Anatoly Baksheev, Kirill Korniyakov, and Victor Eruhimov.2012. Realtime Computer
- [3] <https://opencv.org/>
- [4] <https://www.tutorialspoint.com/dip/>
- [5] <https://pshychmnemonics.wordpress.com/2015/07/03/primary-entions>
- [6] <https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/>
- [7] [http://dsp-book.narod.ru/PWSA/8276\\_01.pdf](http://dsp-book.narod.ru/PWSA/8276_01.pdf)