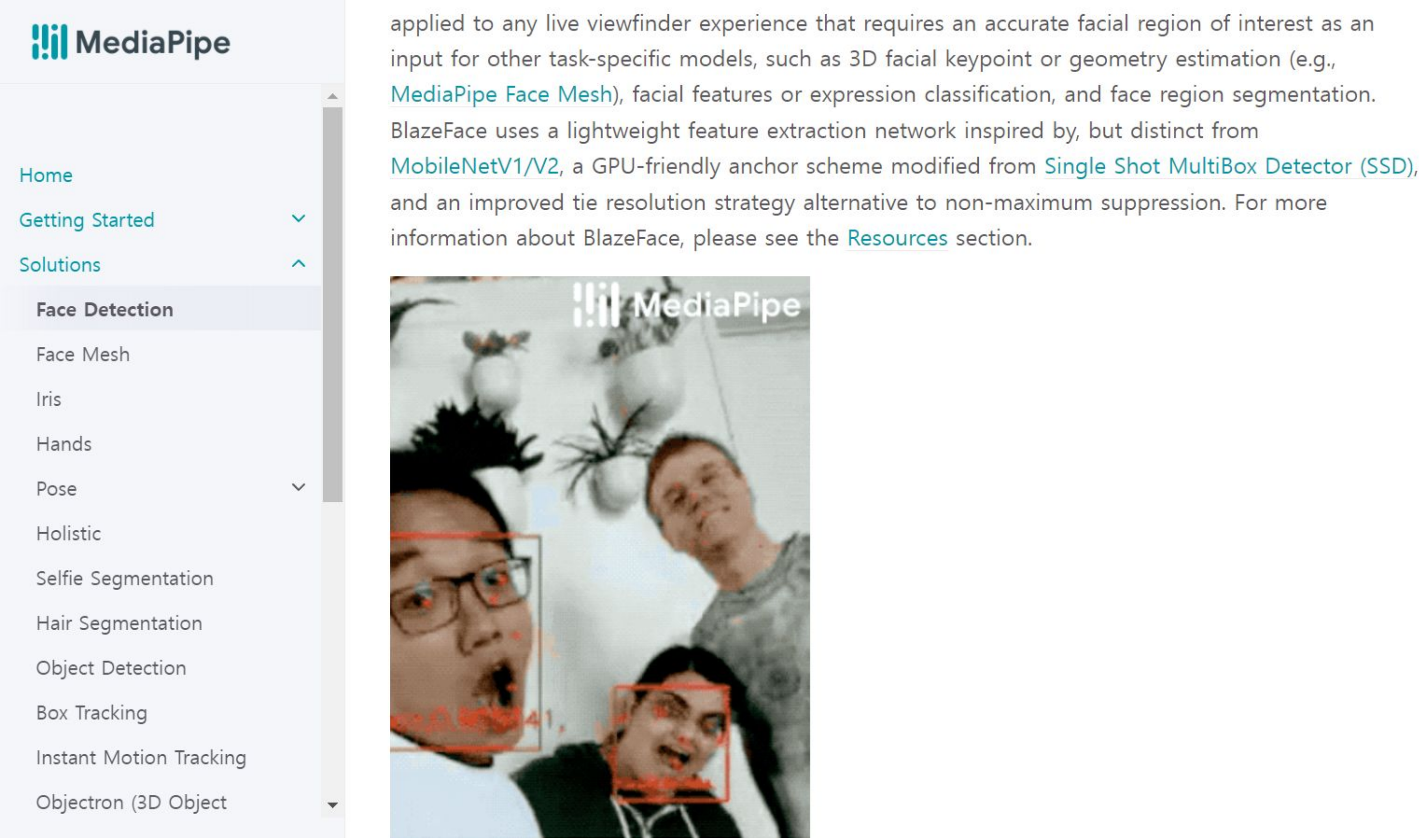
Face Recognition

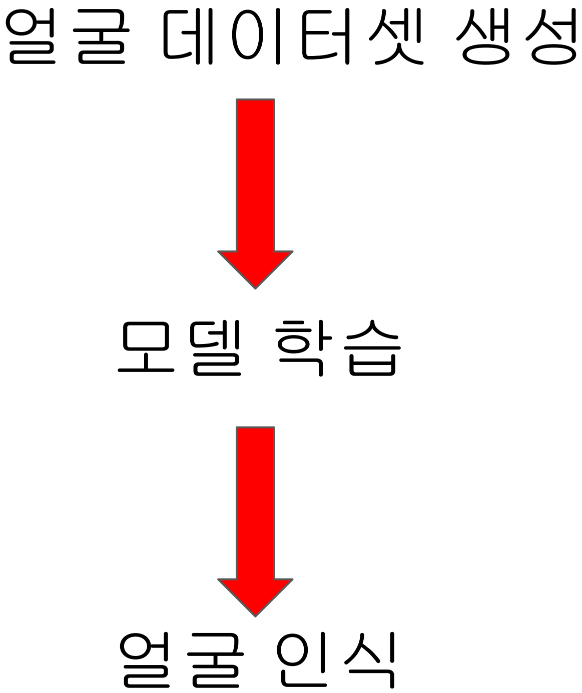
21 김민솔



MediaPipe

Home
Getting Started
Solutions
Face Detection
Face Mesh
Iris
Hands
Pose
Holistic
Selfie Segmentation
Hair Segmentation
Object Detection
Box Tracking
Instant Motion Tracking
Objectron (3D Object

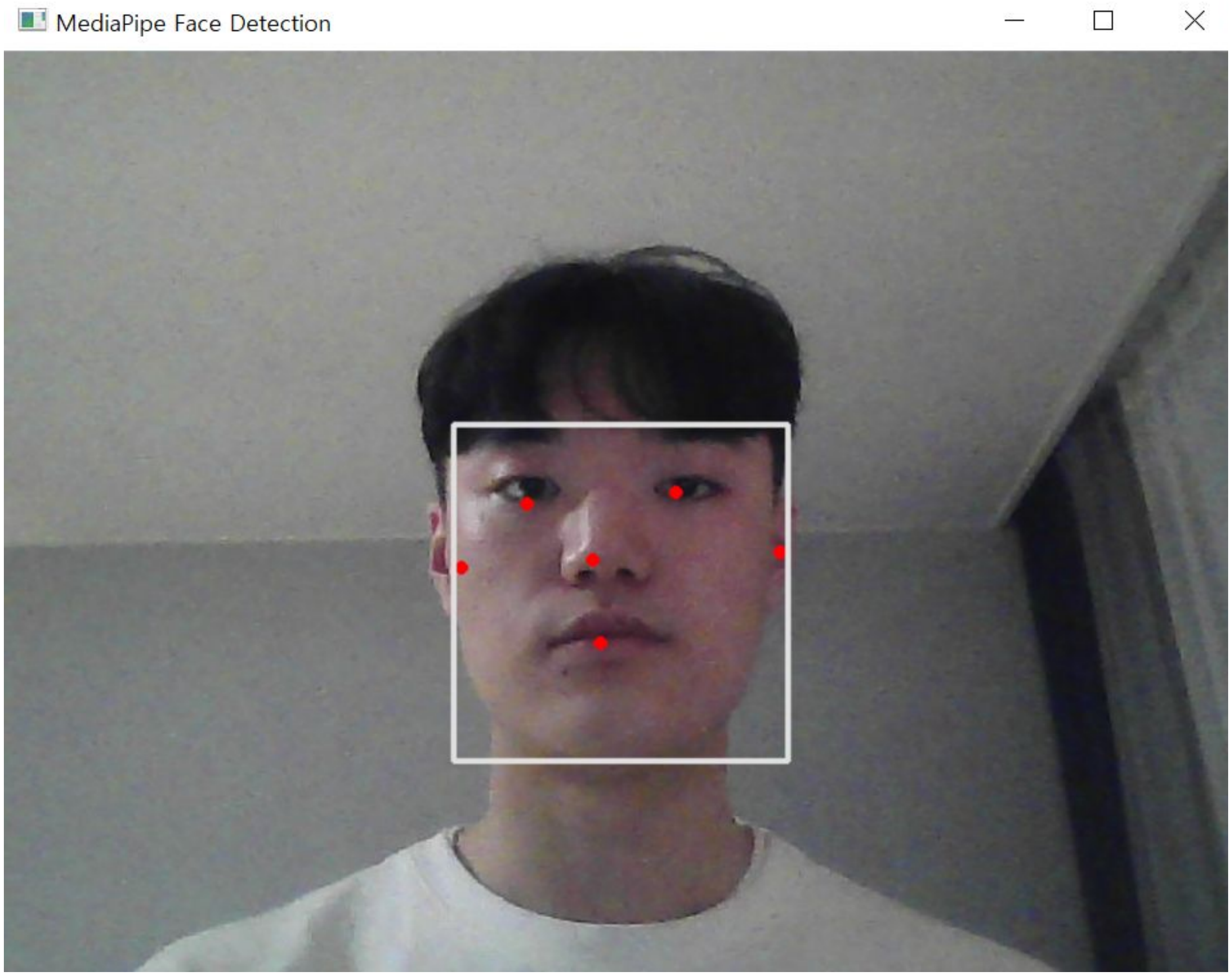
applied to any live viewfinder experience that requires an accurate facial region of interest as an input for other task-specific models, such as 3D facial keypoint or geometry estimation (e.g., MediaPipe Face Mesh), facial features or expression classification, and face region segmentation. BlazeFace uses a lightweight feature extraction network inspired by, but distinct from MobileNetV1/V2, a GPU-friendly anchor scheme modified from Single Shot MultiBox Detector (SSD), and an improved tie resolution strategy alternative to non-maximum suppression. For more information about BlazeFace, please see the Resources section.

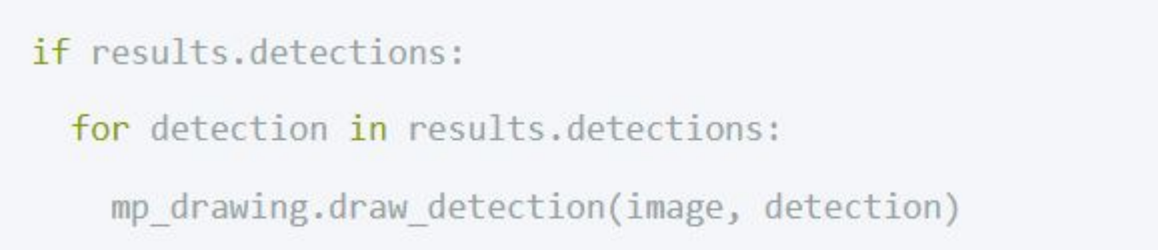


얼굴 데이터셋 생성

모델 학습

얼굴 인식

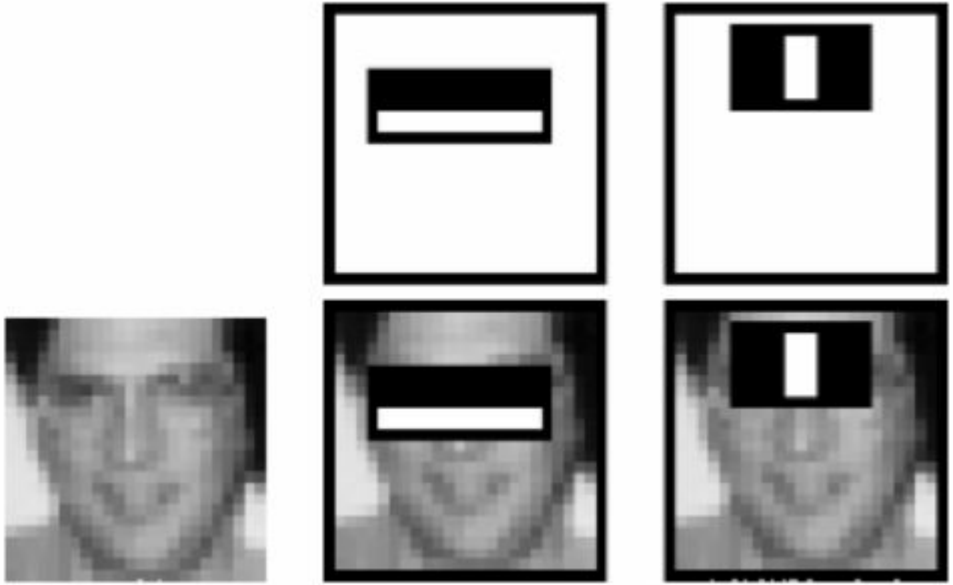

MediaPipe Face Detection

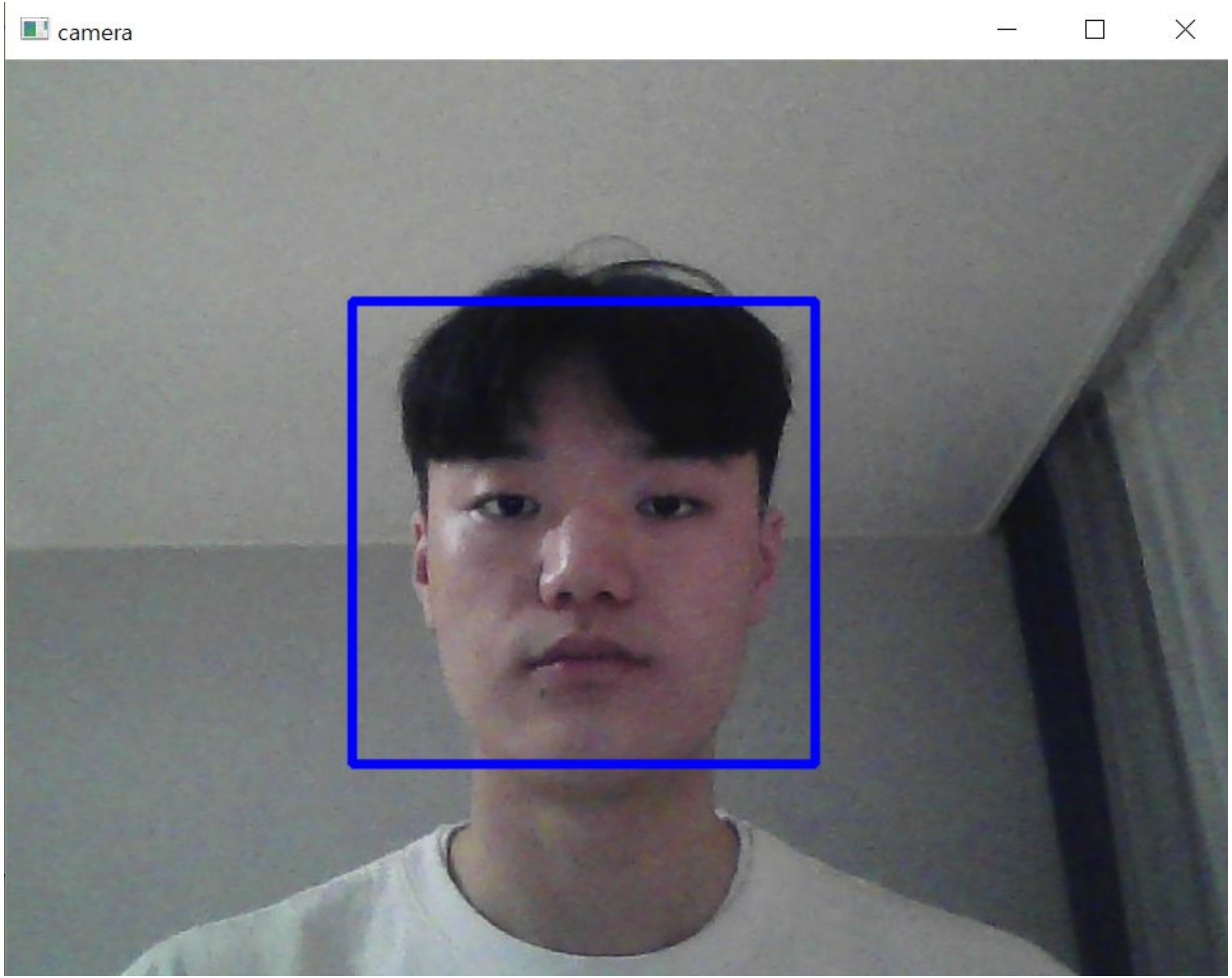


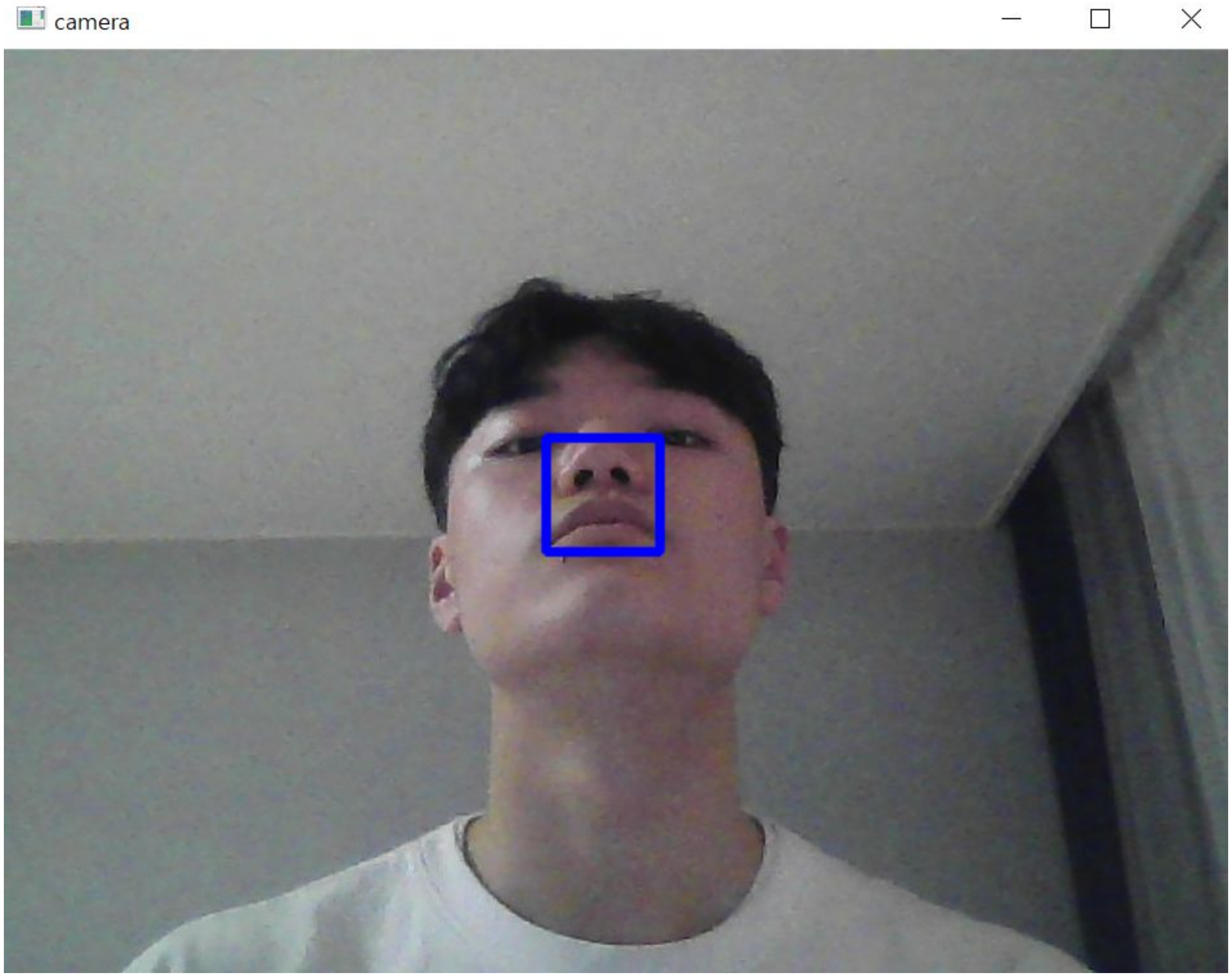
```
if results.detections:

  for detection in results.detections:

    mp_drawing.draw_detection(image, detection)
```

?

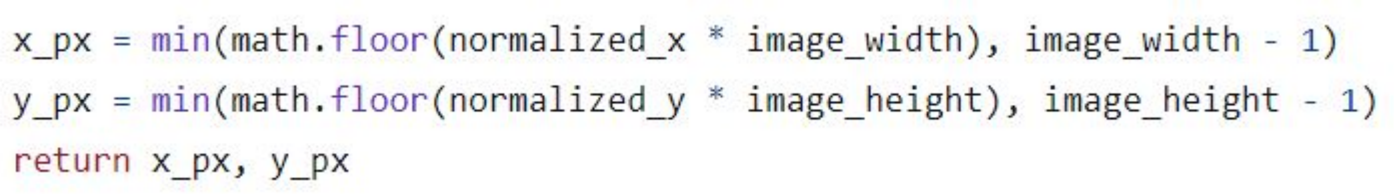
Harr cascade



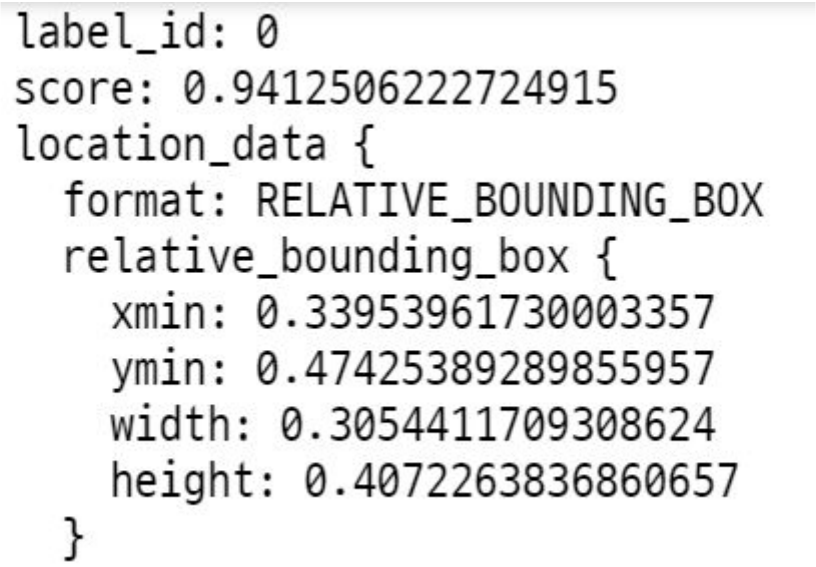

camera


camera

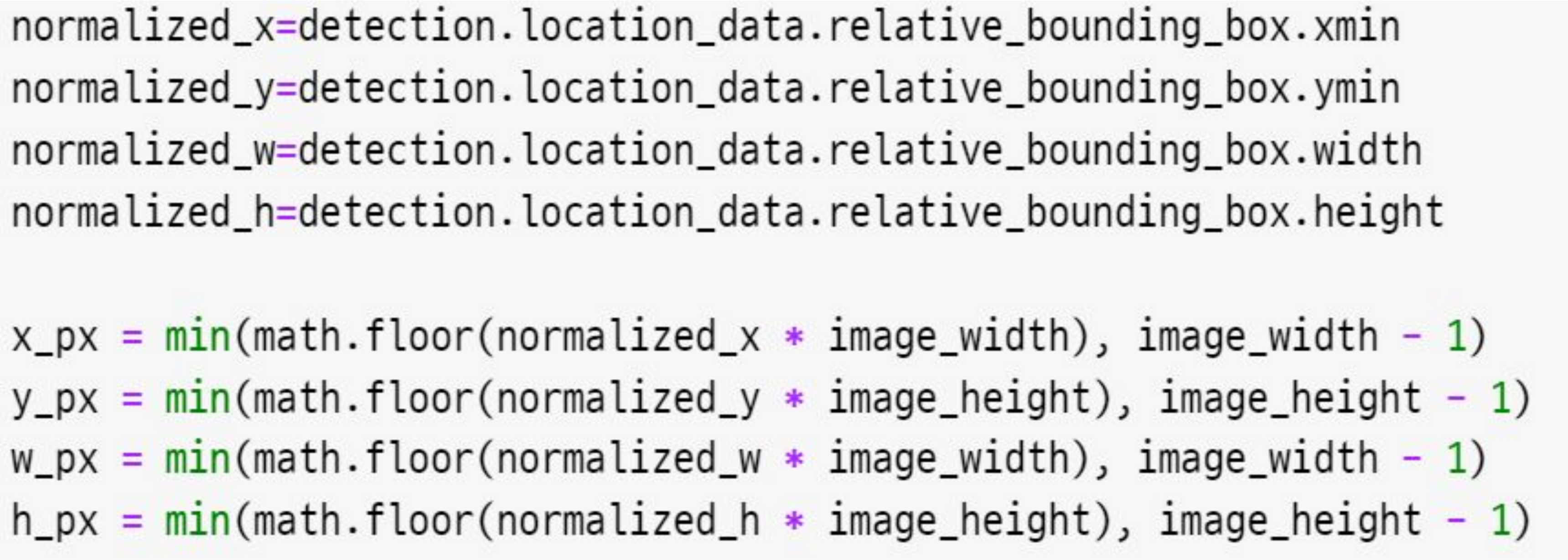
https://github.com/google/mediapipe/blob/master/mediapipe/python/solutions/



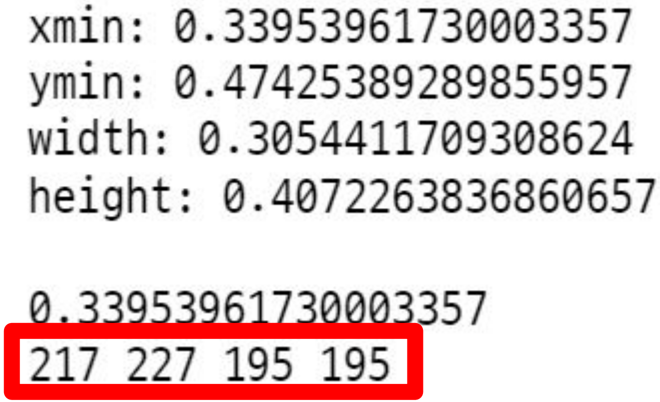
```
x_px = min(math.floor(normalized_x * image_width), image_width - 1)
y_px = min(math.floor(normalized_y * image_height), image_height - 1)
return x_px, y_px
```



```
label_id: 0
score: 0.9412506222724915
location_data {
  format: RELATIVE_BOUNDING_BOX
  relative_bounding_box {
    xmin: 0.33953961730003357
    ymin: 0.47425389289855957
    width: 0.3054411709308624
    height: 0.4072263836860657
  }
```

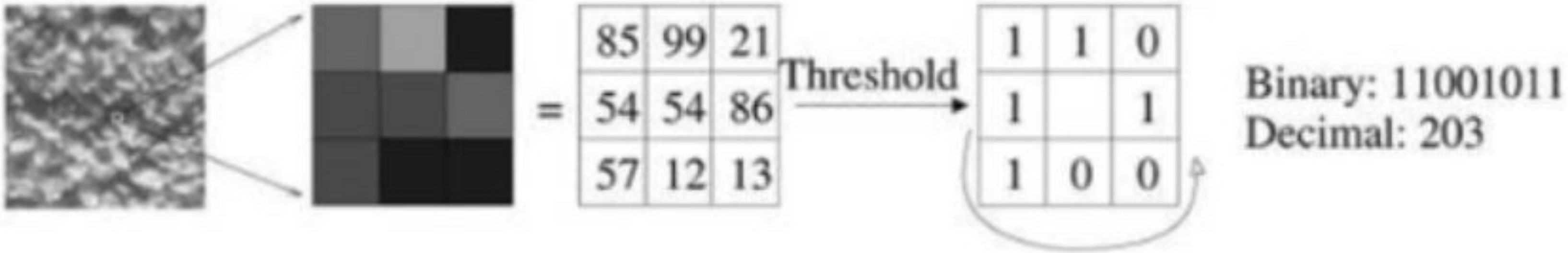


```
normalized_x=detection.location_data.relative_bounding_box.xmin
normalized_y=detection.location_data.relative_bounding_box.ymin
normalized_w=detection.location_data.relative_bounding_box.width
normalized_h=detection.location_data.relative_bounding_box.height

x_px = min(math.floor(normalized_x * image_width), image_width - 1)
y_px = min(math.floor(normalized_y * image_height), image_height - 1)
w_px = min(math.floor(normalized_w * image_width), image_width - 1)
h_px = min(math.floor(normalized_h * image_height), image_height - 1)
```



```
xmin: 0.33953961730003357
ymin: 0.4742538928985957
width: 0.3054411709308624
height: 0.4072263836860657

0.33953961730003357
217 227 195 195
```

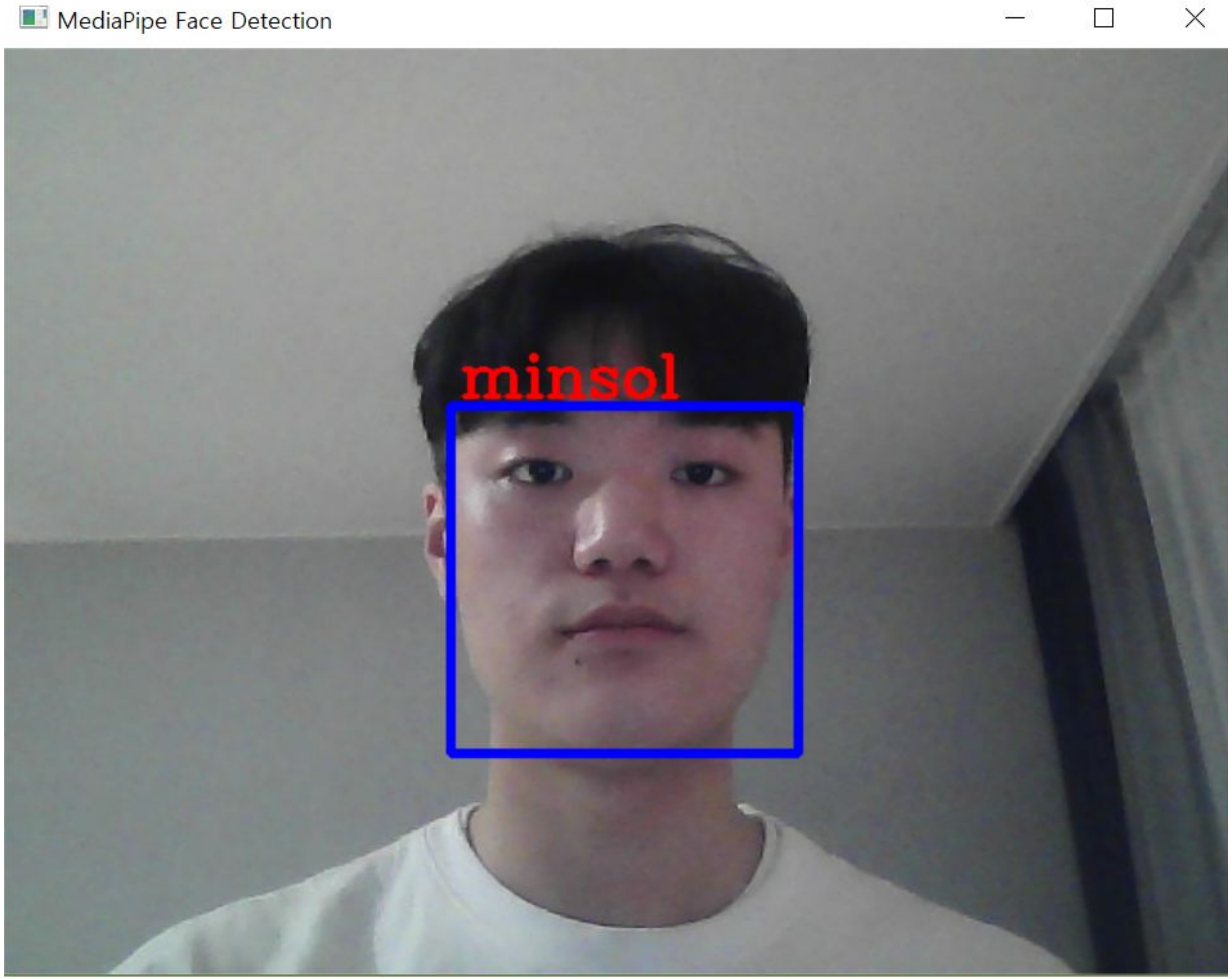
face_recog...



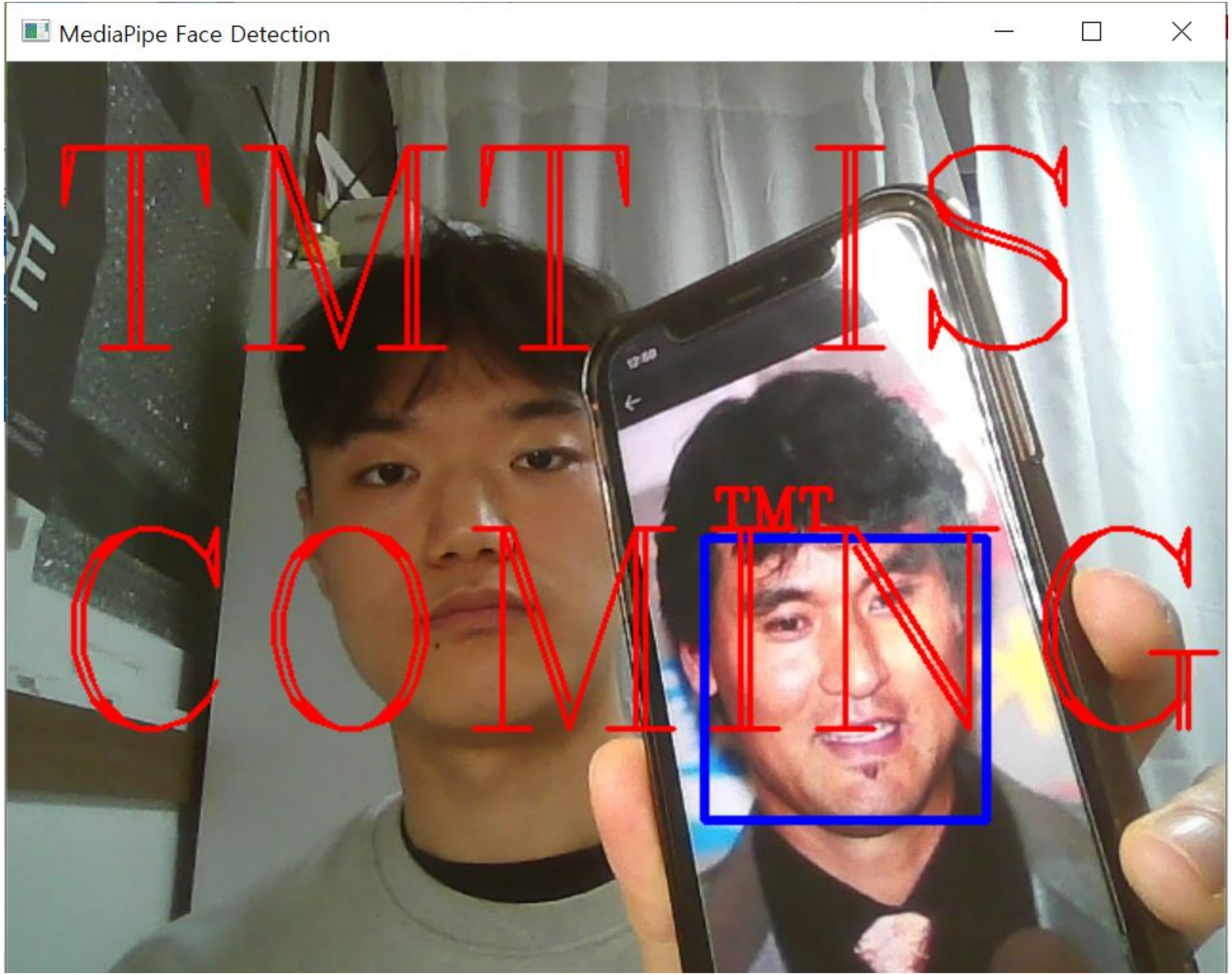
85	99	21
54	54	86
57	12	13

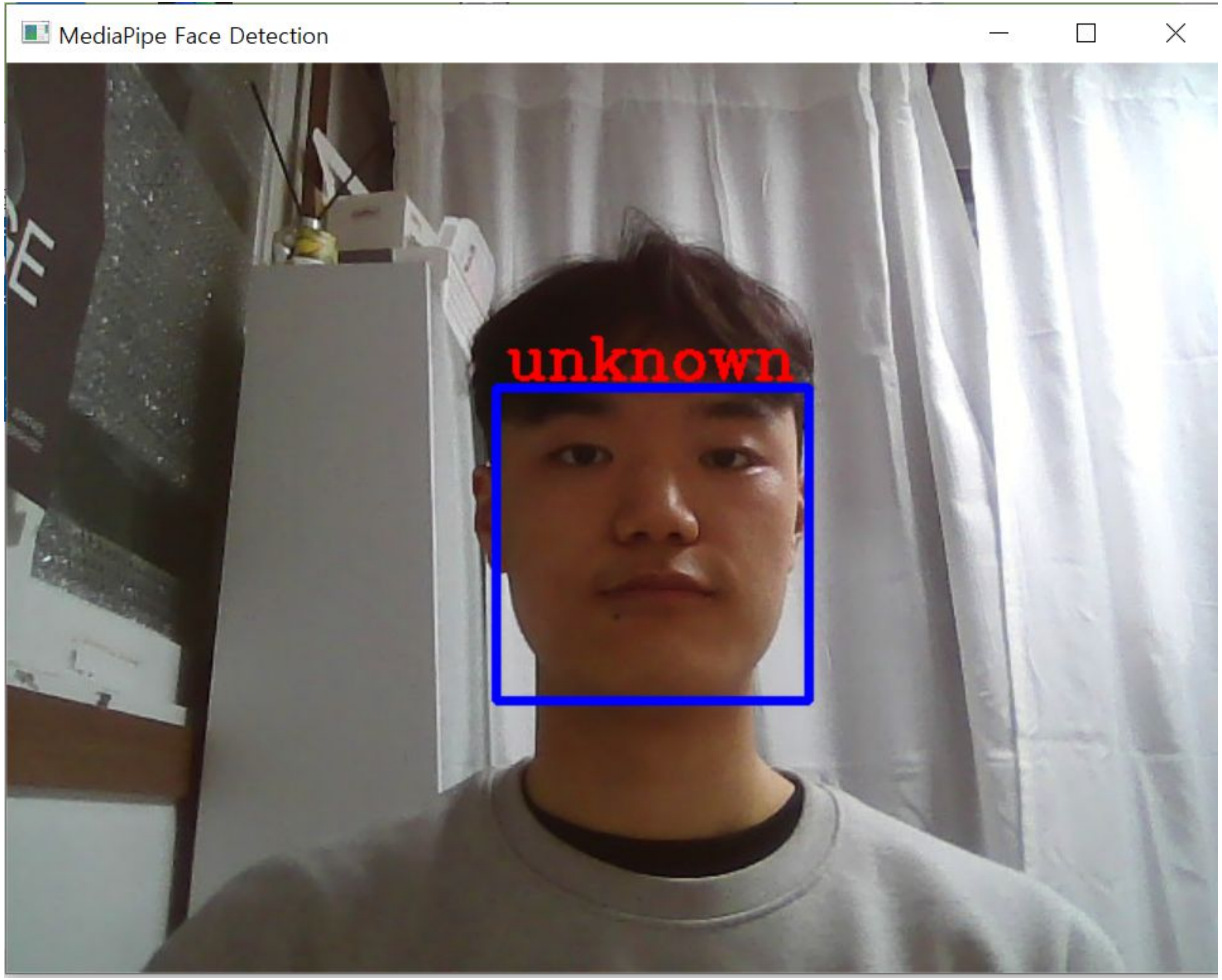
1	1	0
1		1
1	0	0

Binary: 11001011
Decimal: 203

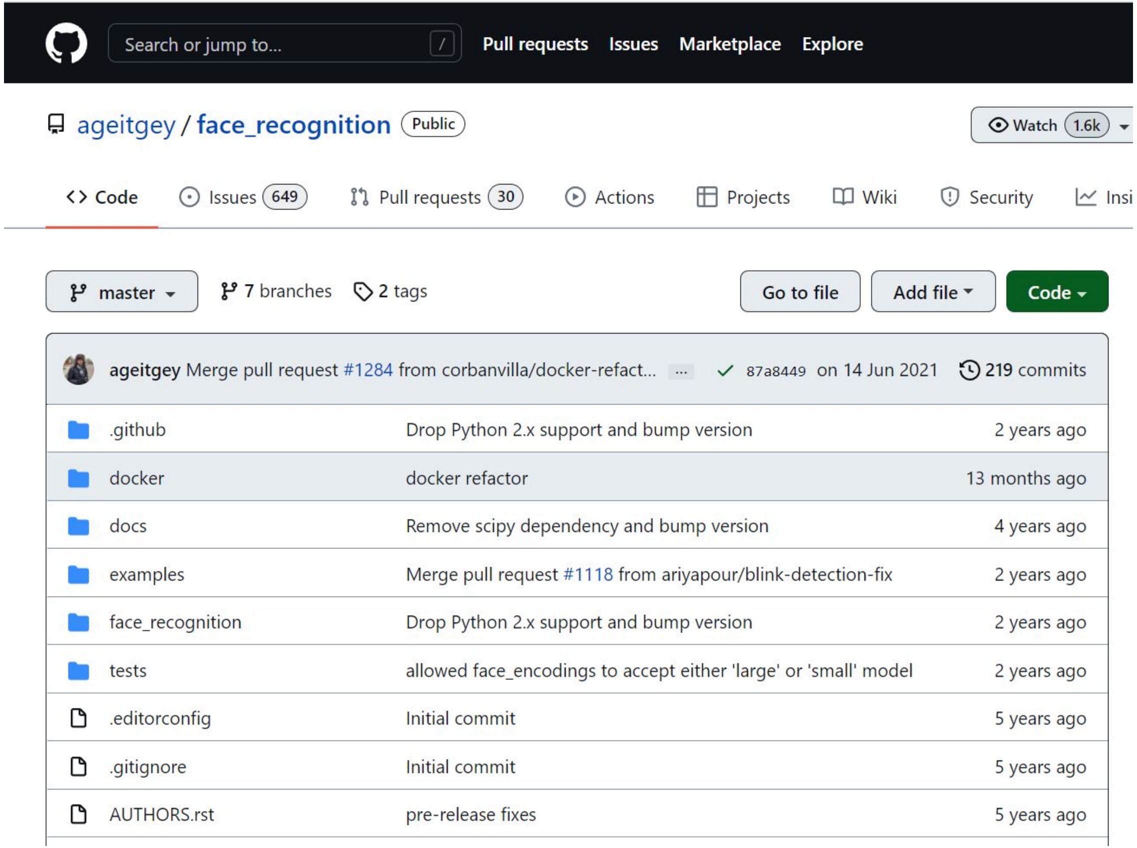

MediaPipe Face Detection
minsol

조이뉴스24


MediaPipe Face Detection
TMT IS
COMING
TMT

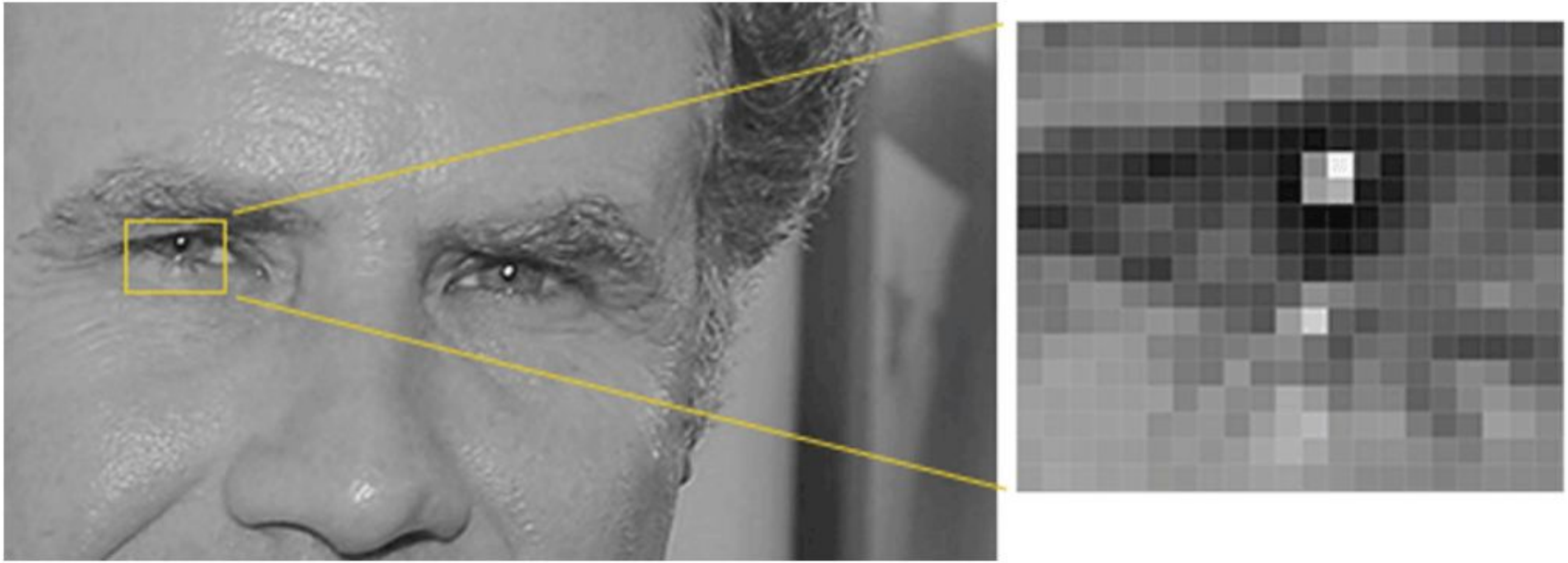

MediaPipe Face Detection
unknown

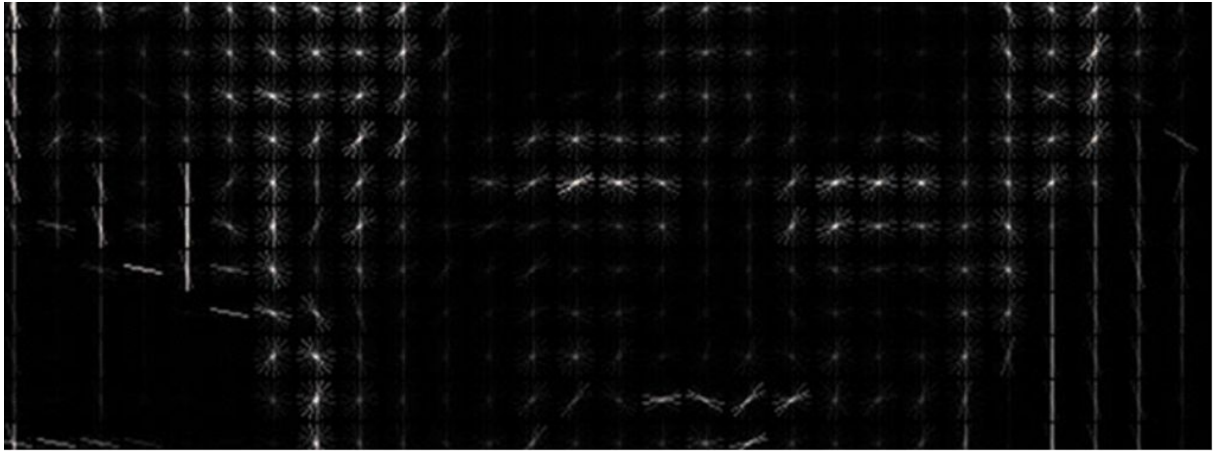
https://github.com/ageitgey/face_recognition

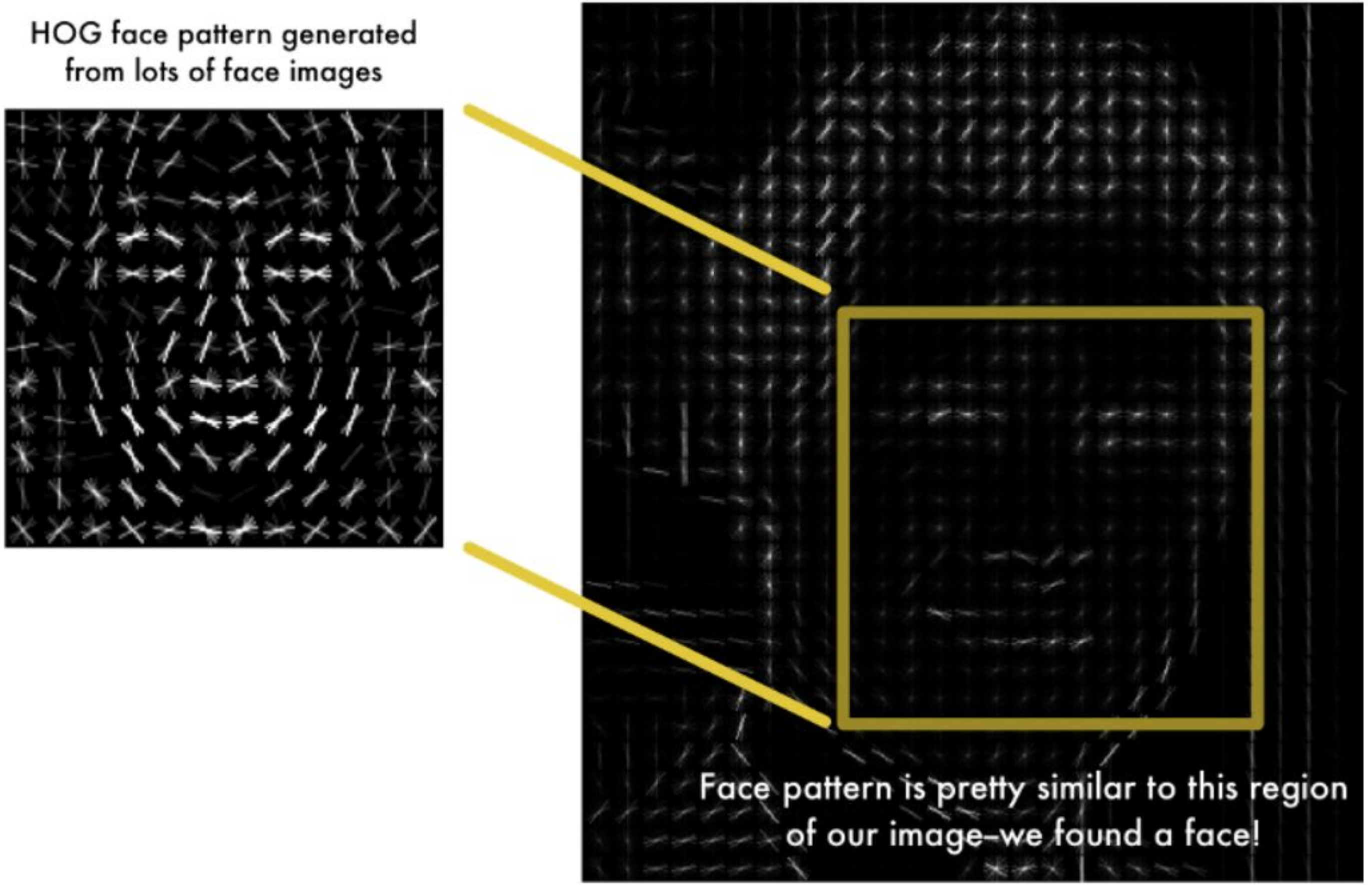


ageitgey Merge pull request #1284 from corbanvilla/docker-refact... ··· ✓ 87a8449 on 14 Jun 2021 219 commits

.github	Drop Python 2.x support and bump version	2 years ago
docker	docker refactor	13 months ago
docs	Remove scipy dependency and bump version	4 years ago
examples	Merge pull request #1118 from ariyapour/blink-detection-fix	2 years ago
face_recognition	Drop Python 2.x support and bump version	2 years ago
tests	allowed face_encodings to accept either 'large' or 'small' model	2 years ago
.editorconfig	Initial commit	5 years ago
.gitignore	Initial commit	5 years ago
AUTHORS.rst	pre-release fixes	5 years ago



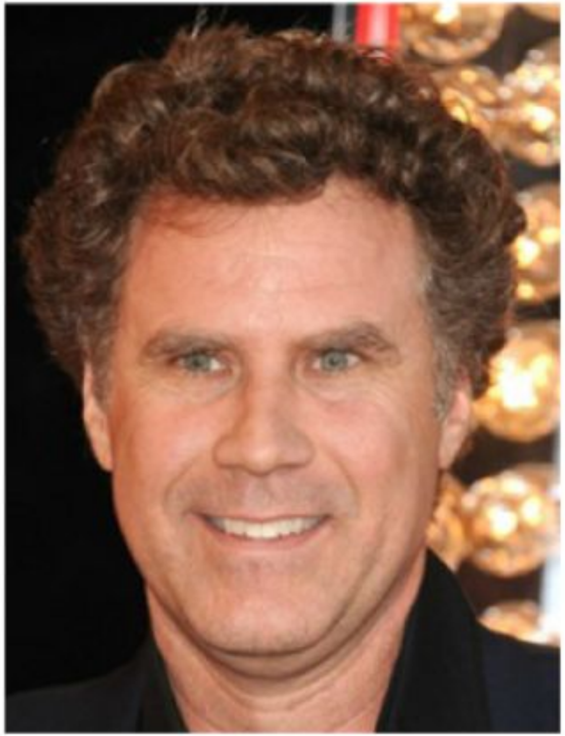


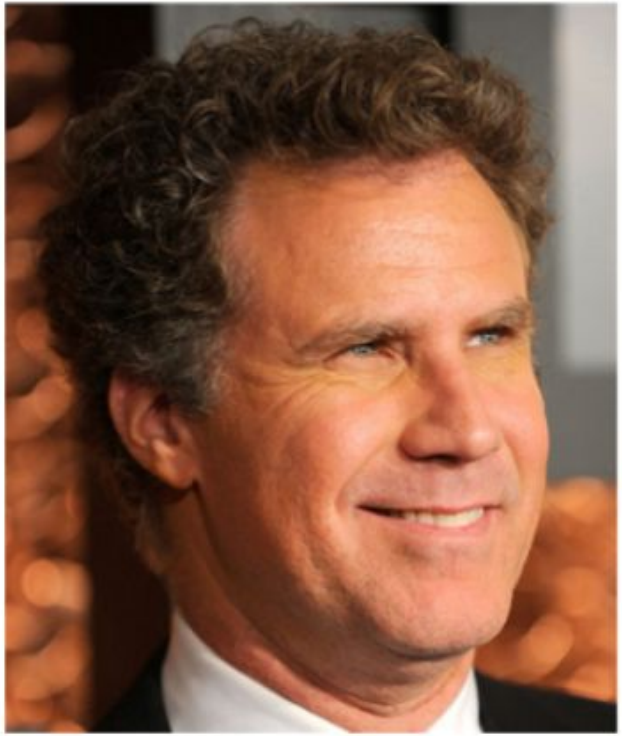


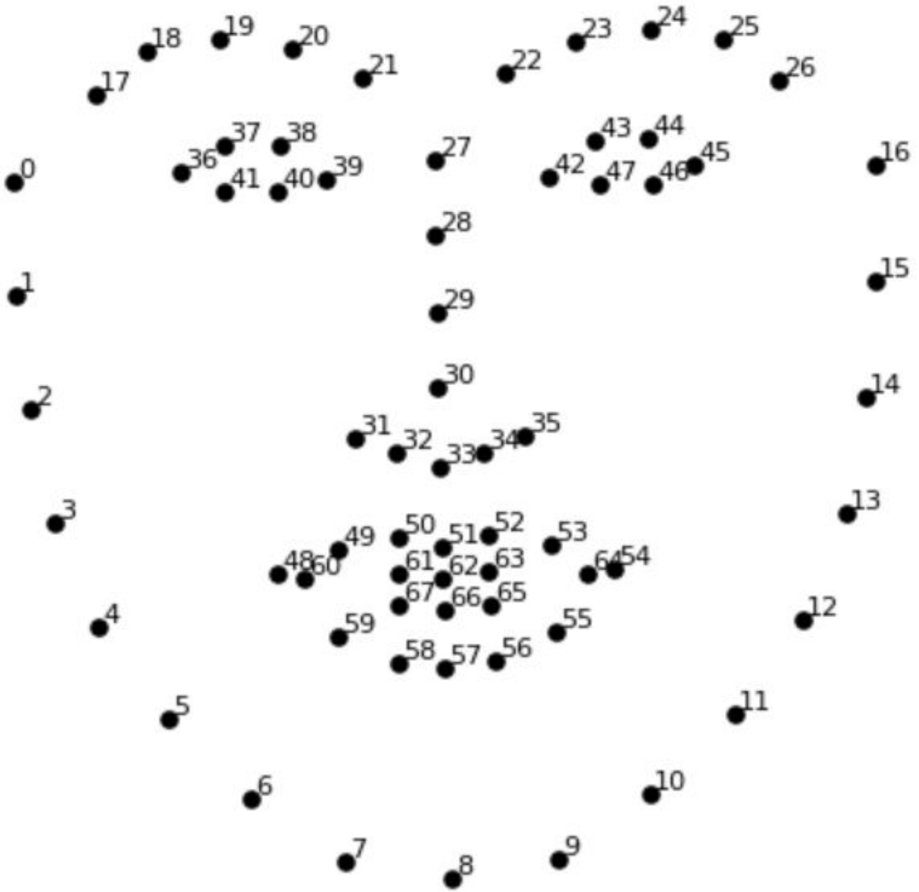
HOG face pattern generated from lots of face images

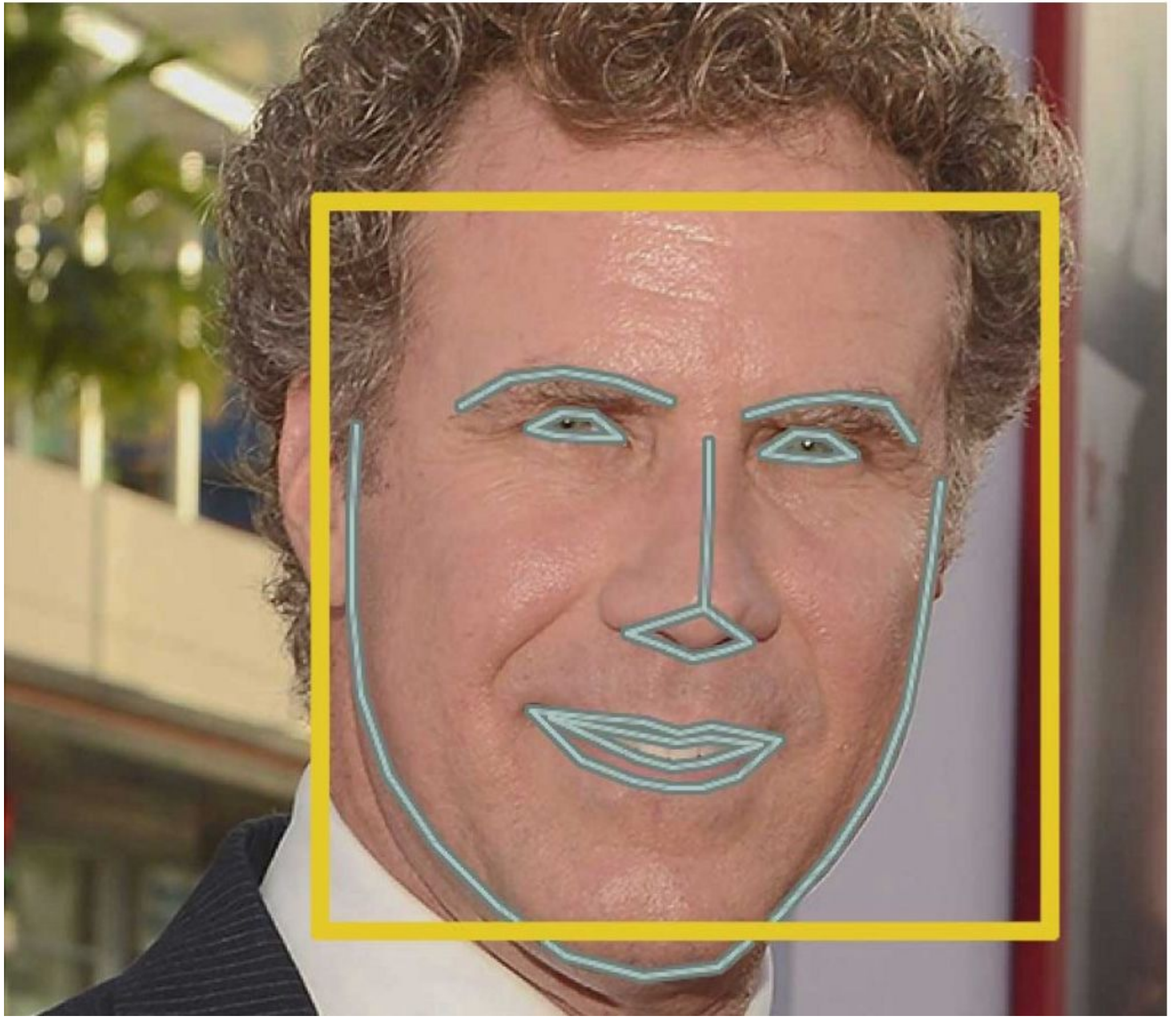
HOG version of our image

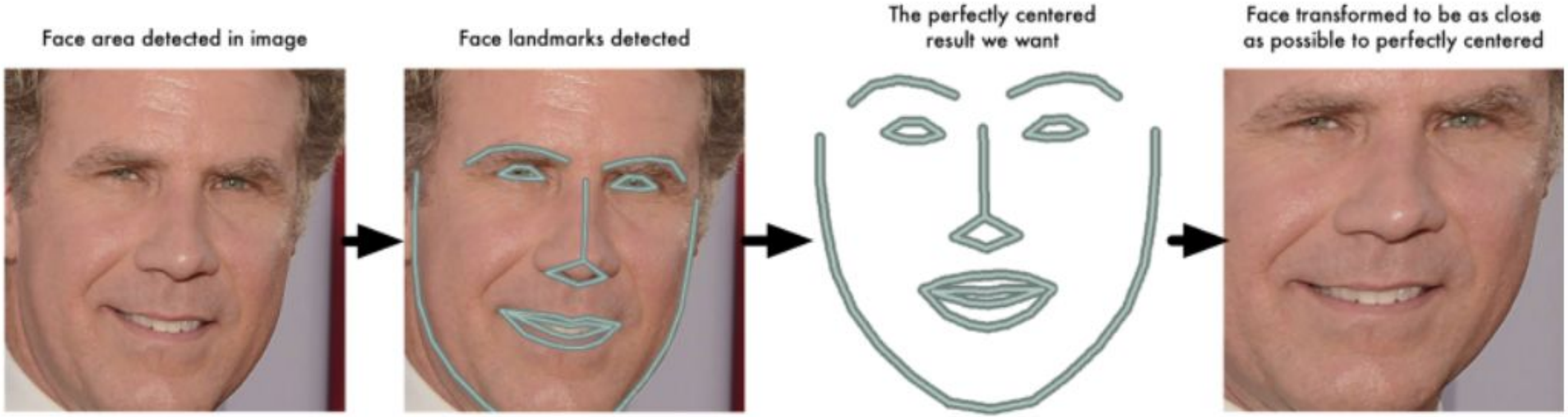
Face pattern is pretty similar to this region of our image–we found a face!






0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67





Face area detected in image

Face landmarks detected

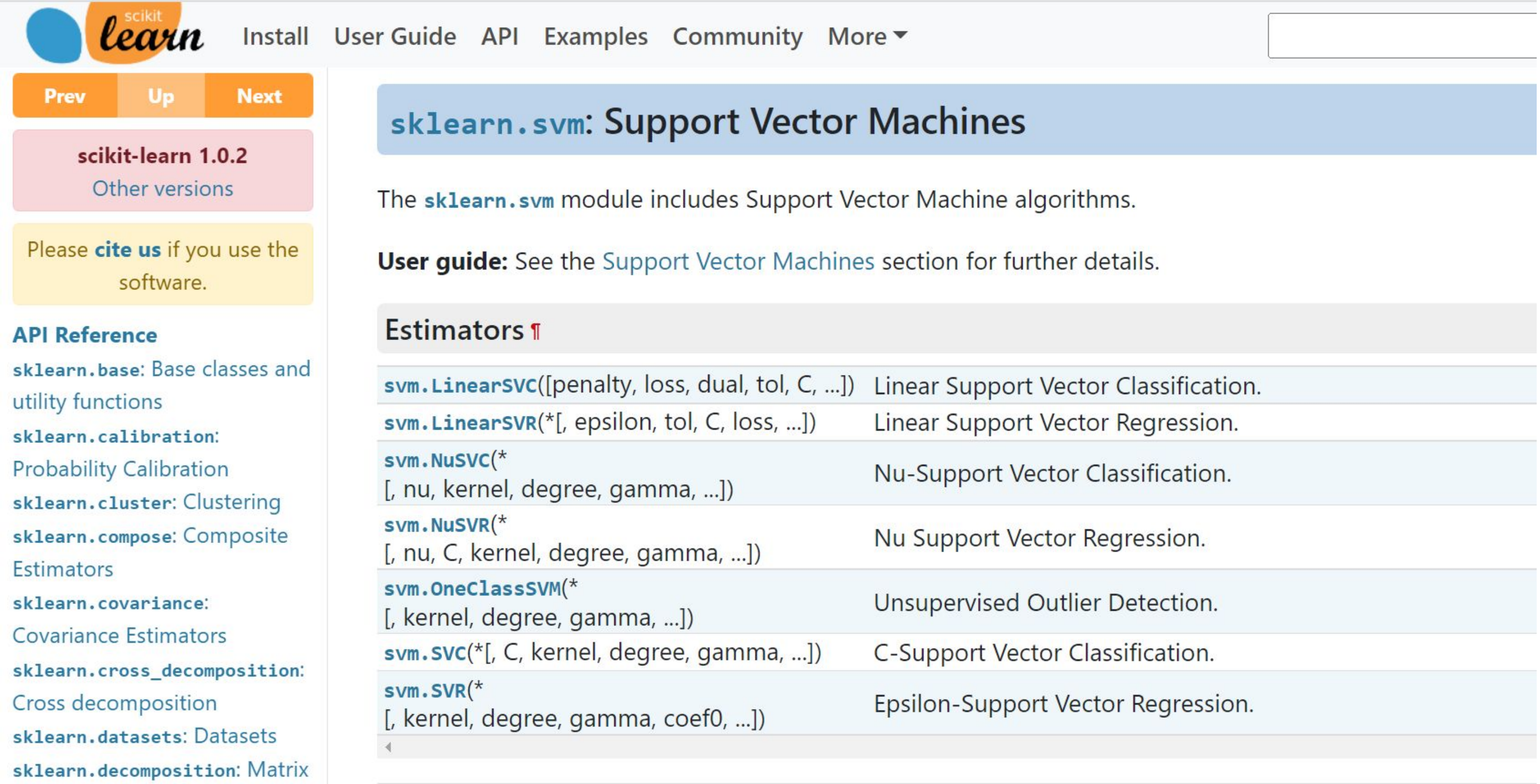
The perfectly centered result we want

Face transformed to be as close as possible to perfectly centered

Input Image

128 Measurements Generated from Image

```
0.097496084868908
0.12529824674129
0.030809439718723
0.036050599068403
-0.097486883401871
-0.0066401711665094
-0.14131525158882
-0.048540540039539
-0.12567175924778
-0.061418771743774
0.046741496771574
-0.12113650143147
0.061606746166945
0.061989940702915
0.10904195904732
-0.019414527341723
0.15245945751667
-0.12216668576002
0.083934605121613
0.087945111095905
-0.021407851949334
-0.018298890441656
-0.011014151386917
0.0093679334968328
0.058139257133007
-0.024210374802351
-0.057223934680223
0.023535015061498
-0.0098039731383324
0.020220354199409
0.0040337680839002
0.051597066223621
0.045223236083984
0.060309179127216
-0.01981477253139
0.065554238855839
0.1226262897253
0.036750309169292
0.14114324748516
-0.061901587992907
-0.10568545013666
-0.074287034571171
0.0061761881224811
-0.21055991947651
0.11345765739679
0.19372203946114
0.084853030741215
0.0064811296761036
-0.16582328081131
-0.0072777755558491
-0.059730969369411
0.11478432267904
0.14841195940971
0.049525424838066
-0.051016297191381
-0.062812767922878
0.0048638740554452
-0.11443792283535
0.014683869667351
-0.081752359867096
0.037022035568953
0.12788131833076
-0.094398014247417
-0.10034311562777
-0.1281466782093
0.17521631717682
0.10801389068365
0.0731306001544
-0.029626874253154
-0.15958009660244
-0.031351584941149
-0.15042643249035
-0.12728653848171
-0.065365232527256
0.14746543765068
0.0041091227903962
0.021352224051952
-0.086726233363152
0.09463594853878
0.21180312335491
-0.035577941685915
-0.036901291459799
-0.070026844739914
-0.089621491730213
0.078333757817745
0.13227833807468
-0.14132921397686
-0.13407498598099
-0.039491076022387
0.071997955441475
0.05228154733777
-0.031709920614958
0.11009479314089
0.18632389605045
-0.11768248677254
-0.040977258235216
0.032084941864014
0.020976085215807
-0.00052163278451189
-0.1318951100111
-0.0059557510539889
0.043374512344599
-0.053343612700701
0.078198105096817
-0.076289616525173
0.12369467318058
0.056418422609568
0.089727647602558
-0.0085843298584223
-0.022388197481632
0.020696049556136
-0.050584398210049
-0.072376452386379
-0.034365277737379
-0.045013956725597
-0.013955107890069
-0.17898085713387
-0.072600327432156
0.0050511928275228
-0.014829395338893
-0.043765489012003
-0.012062266469002
0.012774495407939
0.069833360612392
0.11638788878918
-0.015336792916059
0.10281457751989
-0.082041338086128
```


scikit learn
Install
User Guide
API
Examples
Community
More
Prev
Up
Next
scikit-learn 1.0.2
Other versions
Please cite us if you use the software.
API Reference
sklearn.base: Base classes and utility functions
sklearn.calibration: Probability Calibration
sklearn.cluster: Clustering
sklearn.compose: Composite Estimators
sklearn.covariance: Covariance Estimators
sklearn.cross_decomposition: Cross decomposition
sklearn.datasets: Datasets
sklearn.decomposition: Matrix

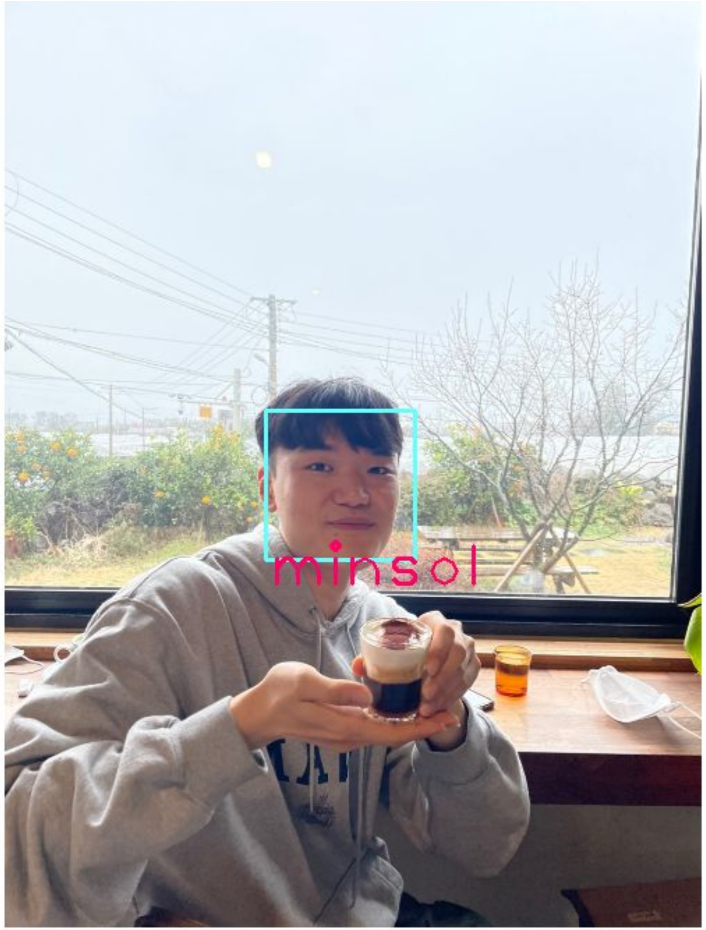
`sklearn.svm`: Support Vector Machines

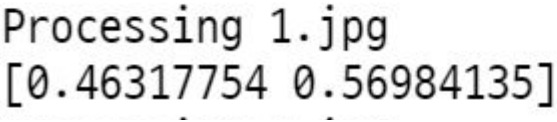
The `sklearn.svm` module includes Support Vector Machine algorithms.

User guide: See the Support Vector Machines section for further details.

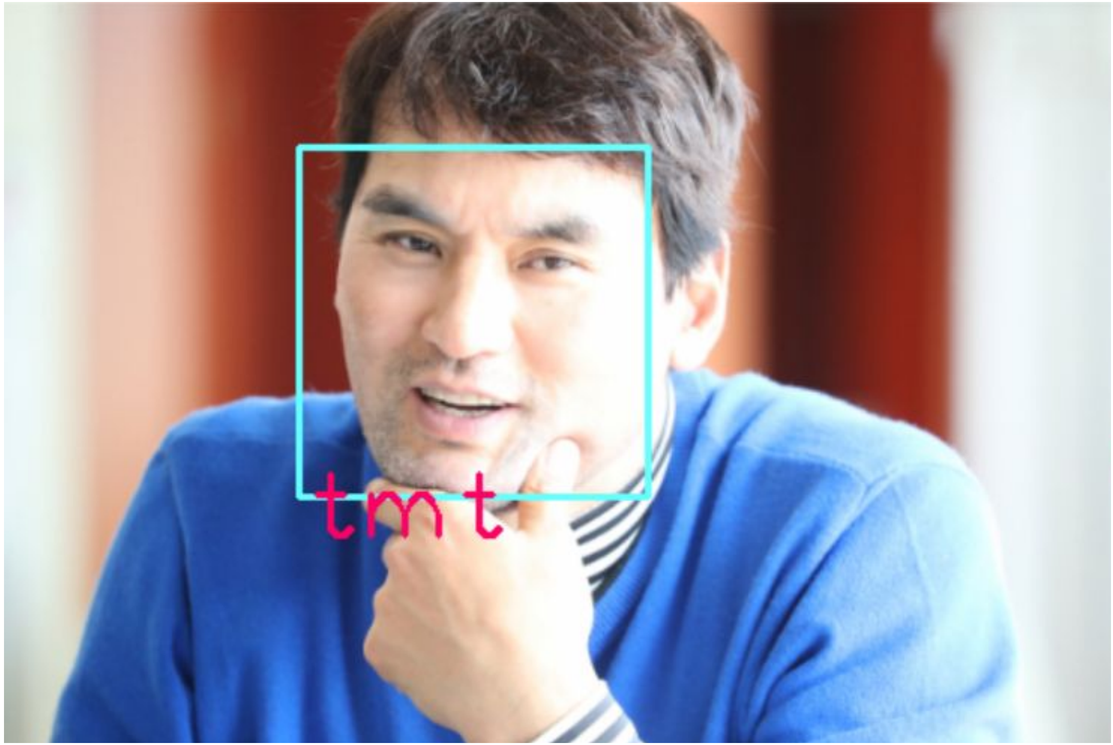
Estimators ¶

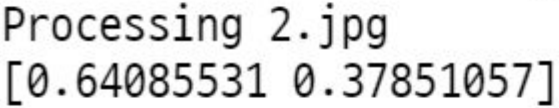
`svm.LinearSVC`([penalty, loss, dual, tol, C, ...])	Linear Support Vector Classification.
`svm.LinearSVR`(*[, epsilon, tol, C, loss, ...])	Linear Support Vector Regression.
`svm.NuSVC`(* [, nu, kernel, degree, gamma, ...])	Nu-Support Vector Classification.
`svm.NuSVR`(* [, nu, C, kernel, degree, gamma, ...])	Nu Support Vector Regression.
`svm.OneClassSVM`(* [, kernel, degree, gamma, ...])	Unsupervised Outlier Detection.
`svm.SVC`(*[, C, kernel, degree, gamma, ...])	C-Support Vector Classification.
`svm.SVR`(* [, kernel, degree, gamma, coef0, ...])	Epsilon-Support Vector Regression.



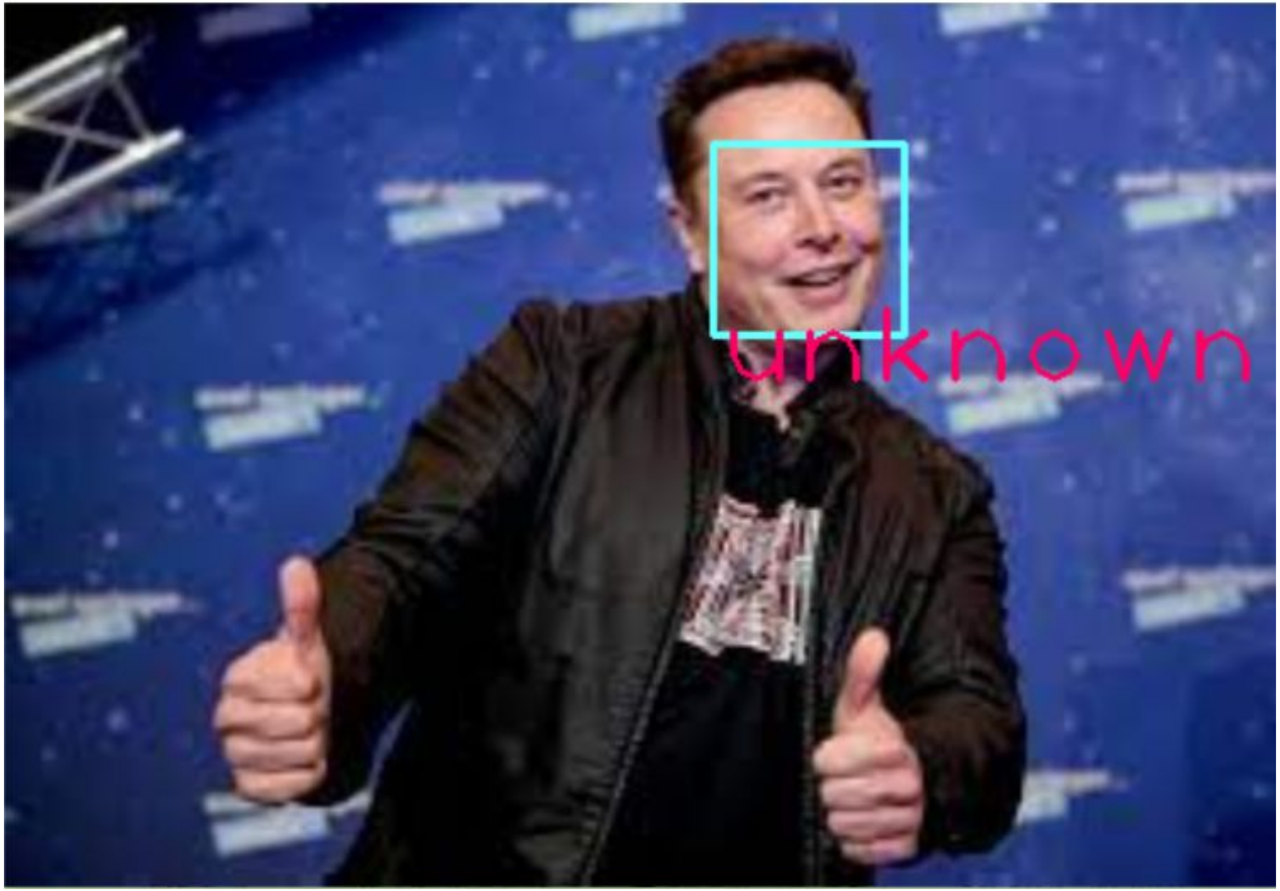


```
Processing 1.jpg
[0.46317754 0.56984135]
```

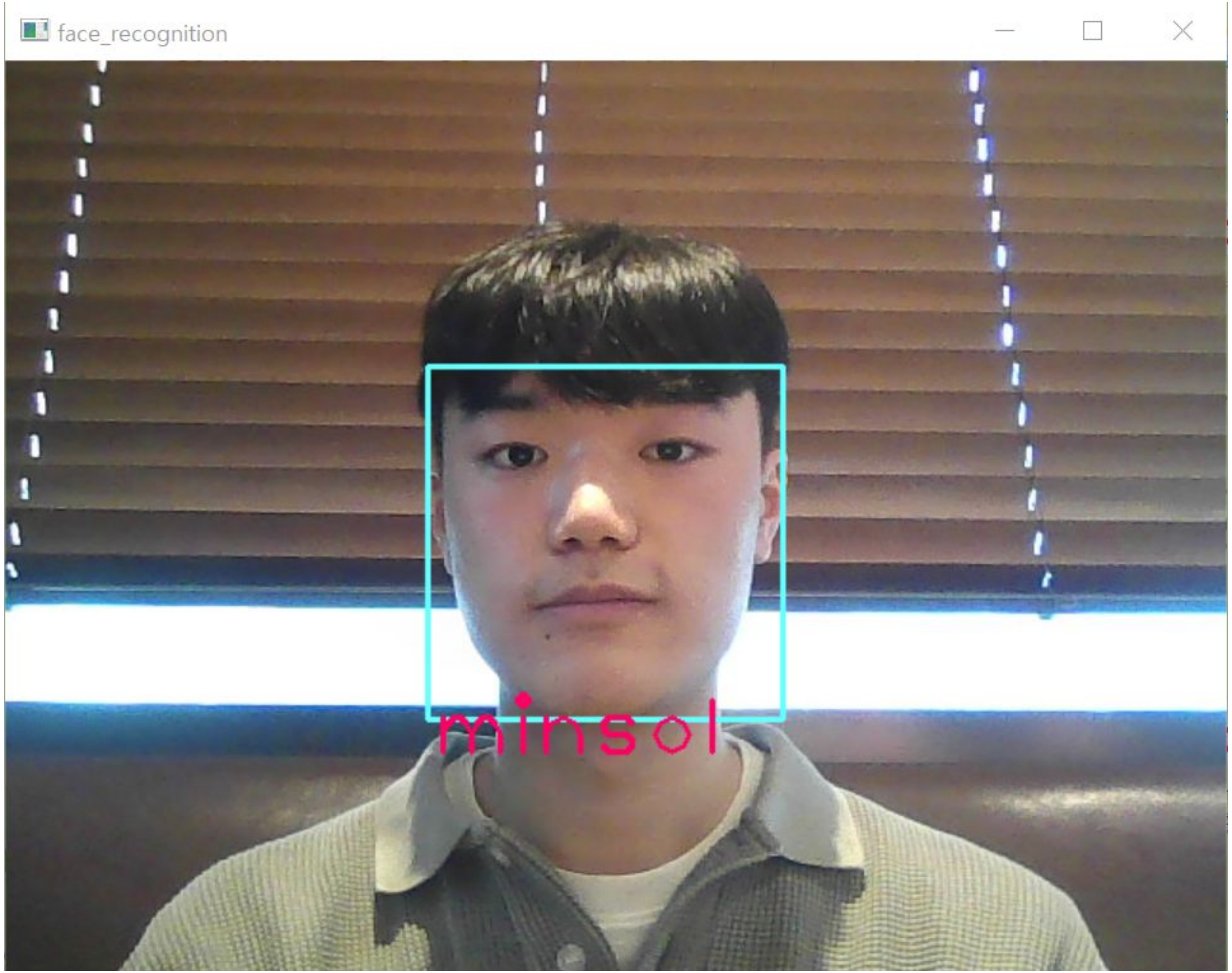




```
Processing 2.jpg
[0.64085531 0.37851057]
```



```
Processing 3.jpg
[0.90024815 0.74262822]
```


face_recognition
minsol

하지만…