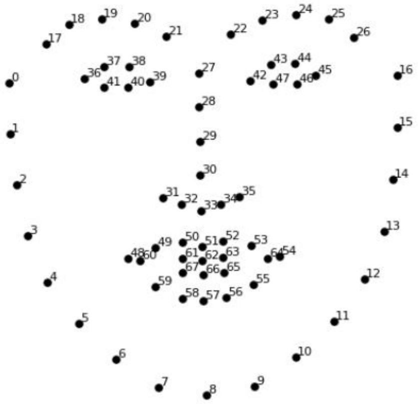
Facial Landmark Detection (keypoint detection)

- Face Recognition
- Driver Analysis
- Face Swap
- Head Pose Estimation
- Facial Region Extraction
- Emotion Detection



https://learnopencv.com/facial-landmark-detection

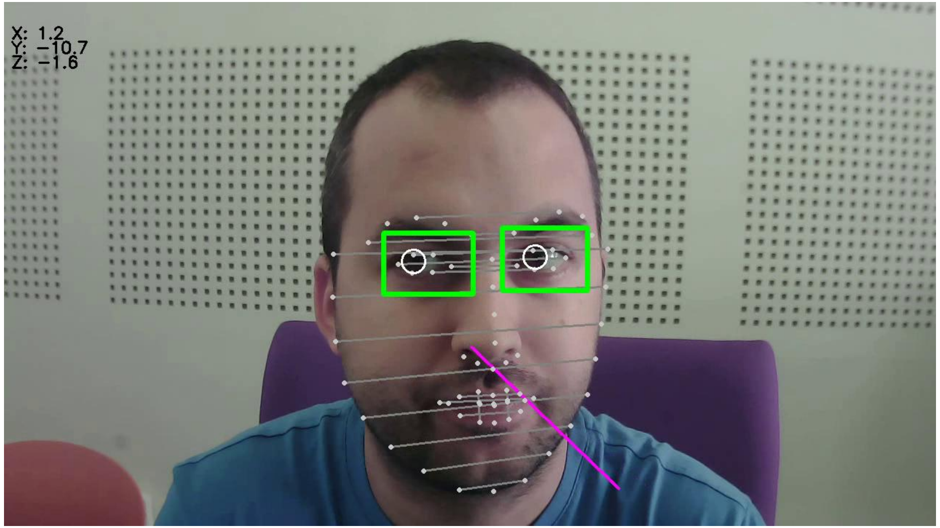
Applications **of** Facial landmark detection (**keypoint detection**)

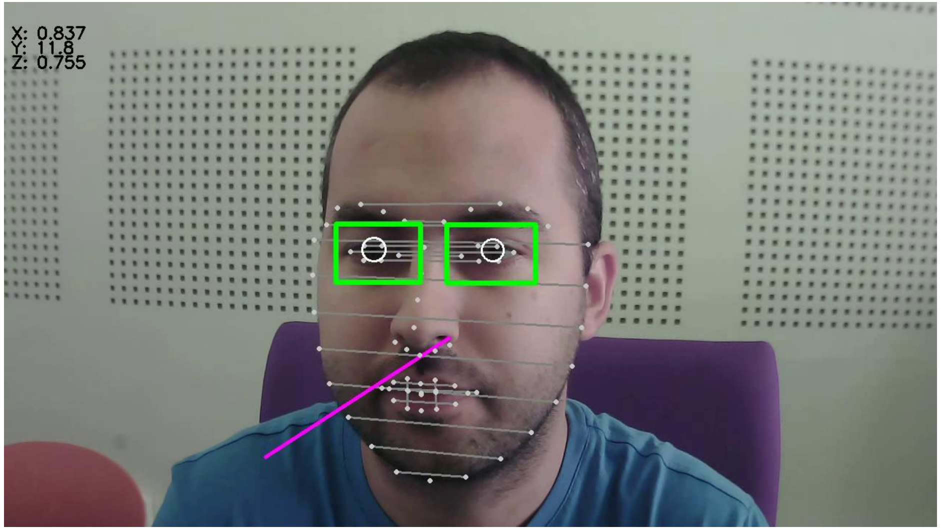
- Facial landmarks can be used to **align facial images** to improve **face recognition**

https://learnopencv.com/facial-landmark-detection
https://docs.opencv.org/3.0.0/da/d60/tutorial_face_main.html

Applications of Facial landmark detection (keypoint detection)

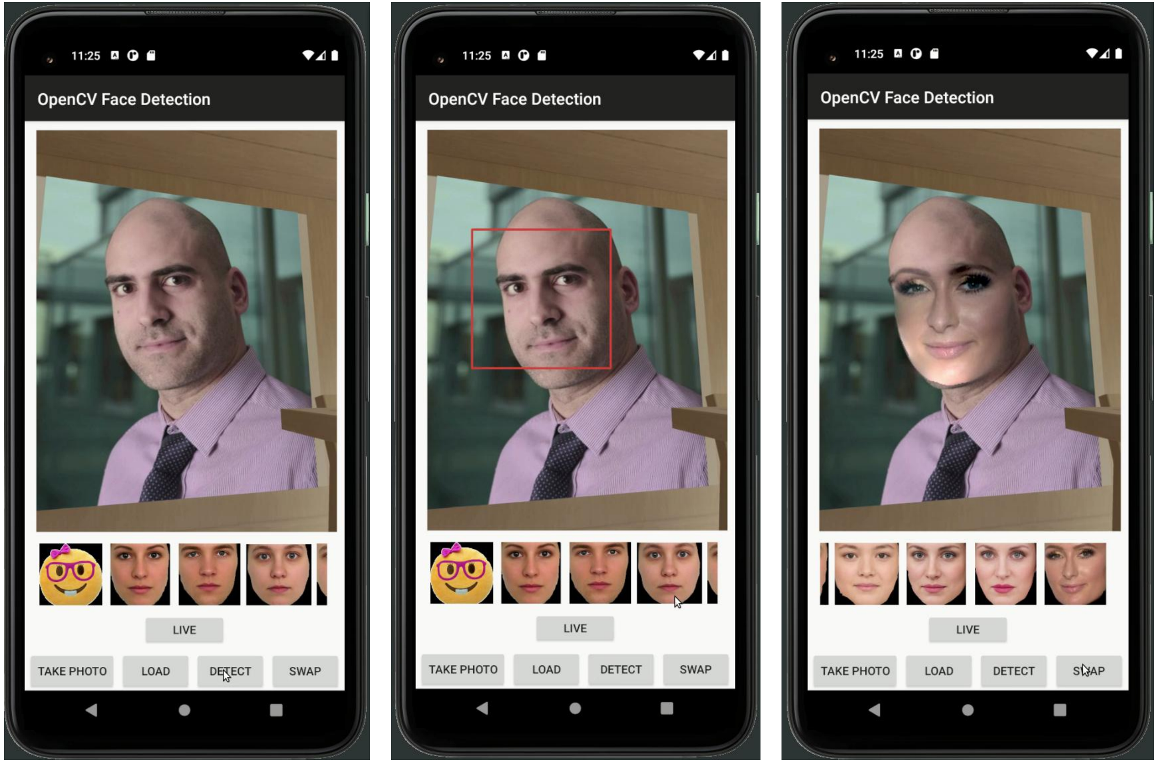
- We can estimate **Head pose** - where the person is looking





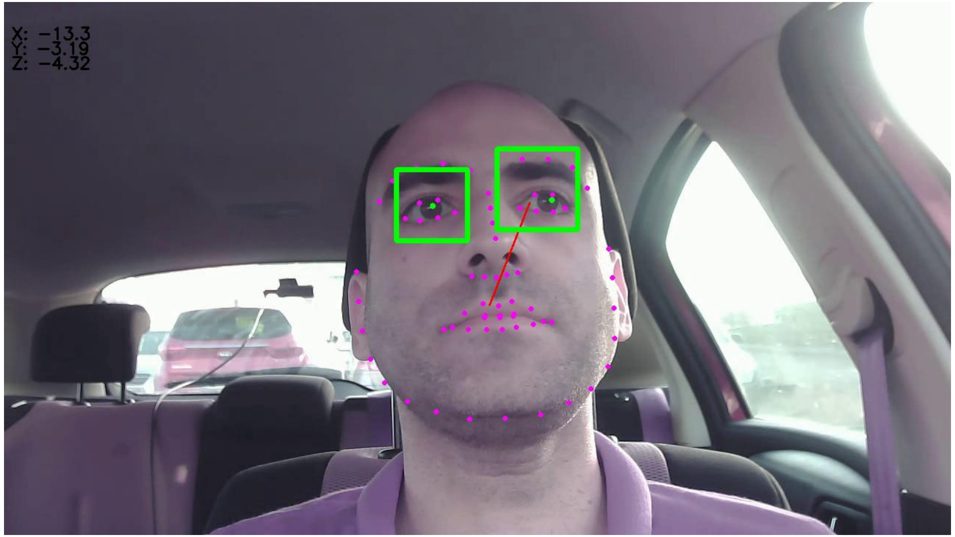
Applications of Facial landmark detection (keypoint detection)

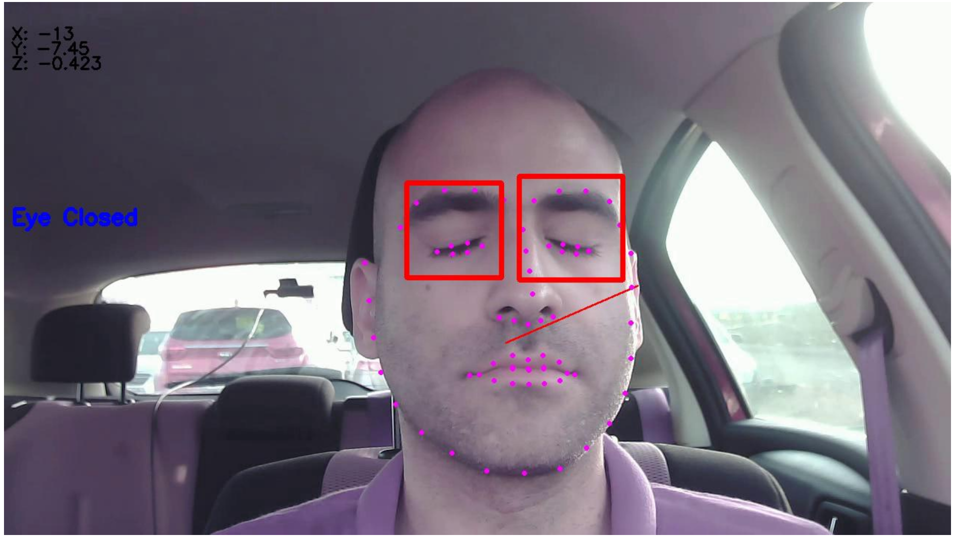
- Face Replacement/Face Swap



Applications of Facial landmark detection (keypoint detection)

- Eye Blink Detection





Challenges (can have a significant influence on the final detection)

(a) Pose (b) Occlusion (c) Expression (d) Illumination

Pose: Faces can have several different poses (e.g. frontal, profile, from the bottom)

Occlusion: For example, glasses, hair or other items can cover the faces

Expression: Different facial expressions can create deformation of parts of the face

Illumination: Different Illumination Intensity (covering certain parts of the face, shadow)

Jin, Xin & Tan, Xiaoyang. (2016). Face Alignment In-the-Wild: A Survey. Computer Vision and Image Understanding. 162. 10.1016/j.cviu.2017.08.008.

Facial Landmark Detection

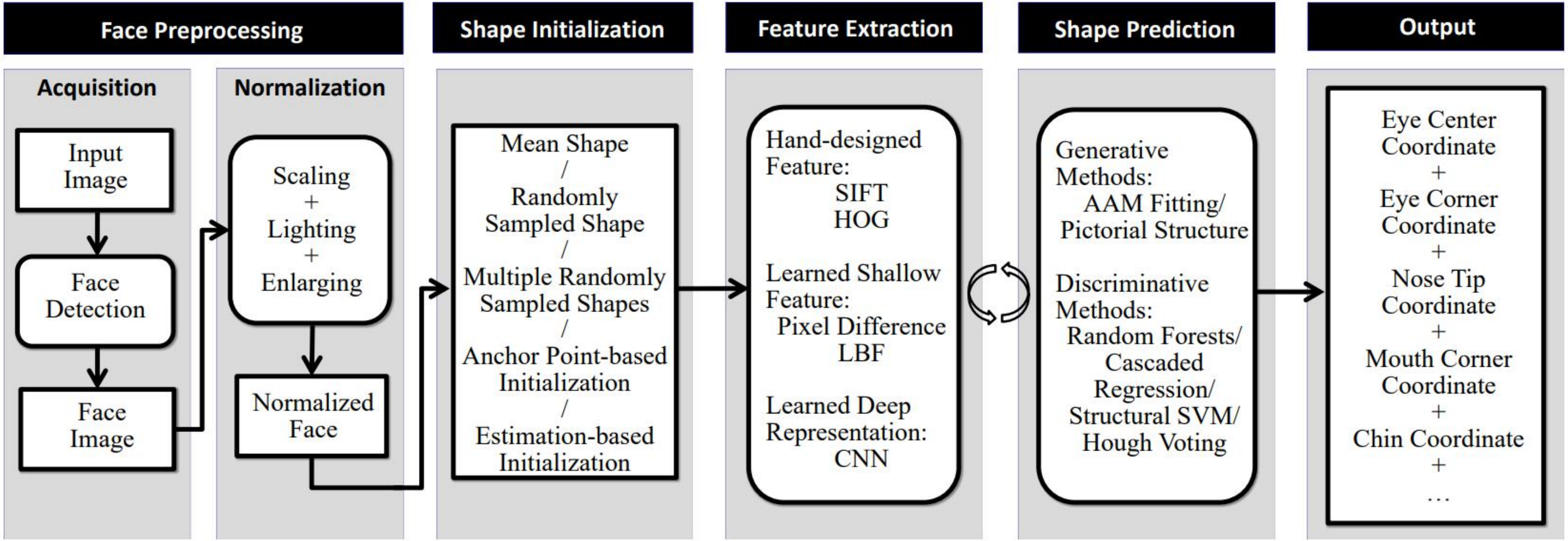


Figure 7: A global system architecture for face alignment.

V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867-1874, doi: 10.1109/CVPR.2014.241.
Jin, Xin & Tan, Xiaoyang. (2016). Face Alignment In-the-Wild: A Survey. Computer Vision and Image Understanding. 162. 10.1016/j.cviu.2017.08.008.

Facial Landmark Detection

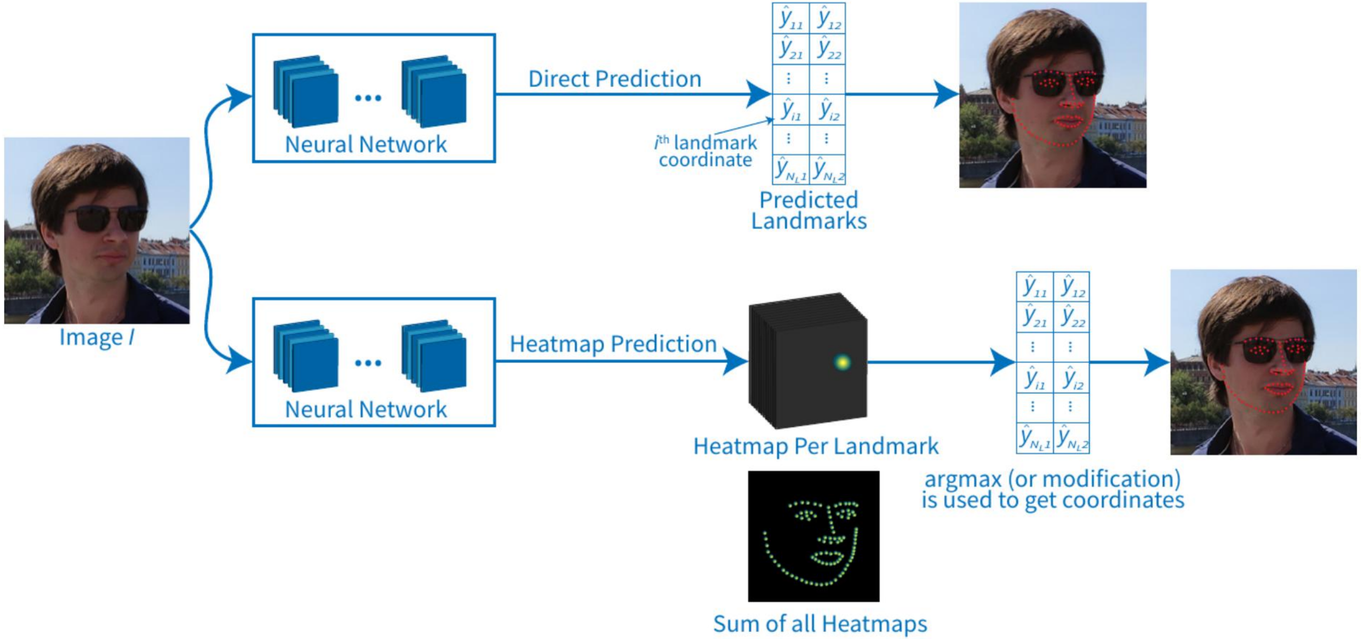


Figure 2. Direct landmark regression (upper row). The problem is solved in a form of regression, where actual landmark coordinates (x, y) are predicted directly by the algorithm. Heatmap-based (bottom row). The algorithm predicts probability distributions of landmark locations in a form of heatmaps. One heatmap per each of the landmarks is formed. Argmax (or its modification) is used to get each landmark coordinates.

Khabarlak, Konstantin & Koriashkina, Larysa. (2022). Fast Facial Landmark Detection and Applications: A Survey. Journal of Computer Science and Technology. 22. 12–41. 10.24215/16666038.22.e02.

Facial Landmark Detection

Table 1: Categorization of the popular approaches for face alignment.

Approach	Representative works
Generative methods	
Active appearance models (AAMs)	
Regression-based fitting	Original AAM [16]; Boosted Appearance Model [17]; Nonlinear discriminative approach [18]; Accurate regression procedures for AMMs [19]
Gradient descent-based fitting	Project-out inverse compositional (POIC) algorithm [20]; Simultaneous inverse compositional (SIC) algorithm [21]; Fast AAM [22]; 2.5D AAM [23]; Active Orientation Models [24]
Part-based generative deformable models	Original Active Shape Model (ASM) [25]; Gauss-Newton deformable part model [26]; Project-out cascaded regression [27]; Active pictorial structures [28]
Discriminative methods	
Constrained local models (CLMs)[a]	
PCA shape model	Regularized landmark mean-shift [29]; Regression voting-based shape model matching [30]; Robust response map fitting [31]; Constrained local neural field [32]
Exemplar shape model	Consensus of exemplar [11]; Exemplar-based graph matching [33]; Robust Discriminative Hough Voting [34]
Other shape models	Gaussian Process Latent Variable Model [35]; Component-based discriminative search [36]; Deep face shape model [37]
Constrained local regression	Boosted regression and graph model [38]; Local evidence aggregation for regression [39]; Guided unsupervised learning for model specific models [40]
Deformable part models (DPMs)	Tree structured part model [41]; Structured output SVM [42]; Optimized part model [43]; Regressive Tree Structured Model [44]
Ensemble regression-voting	Conditional regression forests [12]; Privileged information-based conditional regression forest [45]; Sieving regression forest votes[46]; Nonparametric context modeling [47]
Cascaded regression	
Two-level boosted regression	Explicit shape regression [48]; Robust cascaded pose regression [49]; Ensemble of regression trees [50]; Gaussian process regression trees [51];
Cascaded linear regression	Supervised descent method [52]; Multiple hypotheses-based regression [53]; Local binary feature [54]; Incremental face alignment [55]; Coarse-to-fine shape search [56]
Deep neural networks[b]	
Deep CNNs	Deep convolutional network cascade [57]; Tasks-constrained deep convolutional network [58]; Deep Cascaded Regression[59]
Other deep networks	Coarse-to-fine Auto-encoder Networks (CFAN) [60]; Deep face shape model [37]

[a] Classic Constrained Local Models (CLMs) typically refer to the combination of local detector for each facial point and the parametric Point Distribution Model [61, 62, 29]. Here we extend the range of CLMs by including some methods based on other shape models (i.e., exemplar-based model [11]). In particular, we will show that the exemplar-based method [11] can also be interpreted under the conventional CLM framework.

[b] We note that some deep learning-based systems can also be placed in other categories. For instance, some systems are constructed in a cascade manner [60, 59, 63], and hence can be naturally categorized as cascaded regression. However, to highlight the increasing important role of deep learning techniques for face alignment, we organize them together for more systematic introduction and summarization.

Generative :
These methods typically construct parametric models (or statistical models, **Active shape models**) and try to find the optimal parameters that gives best fit of the shape model to the test image.

Discriminative:
These methods typically use indep endent local detector or regressor for each facial point. In many cases, this requires more training data. As time goes on, larger and larger datasets are being created, and methods based on deep learning have achieved very promising results and play a dominant role in this area in recent years.

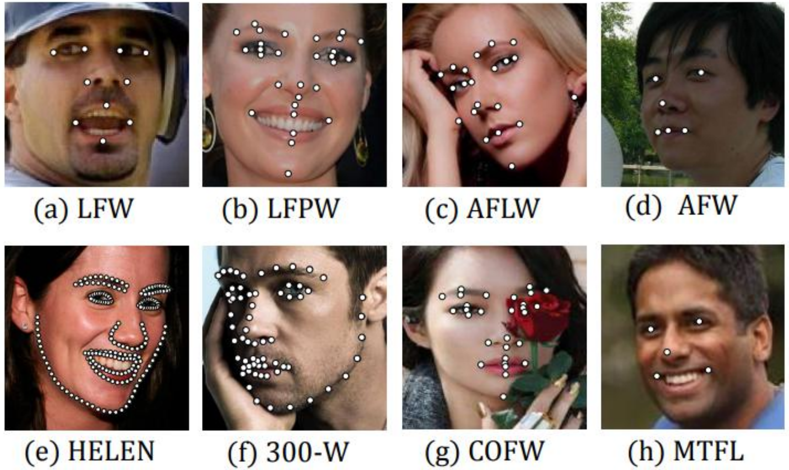
Jin, Xin & Tan, Xiaoyang. (2016). Face Alignment In-the-Wild: A Survey. Computer Vision and Image Understanding. 162. 10.1016/j.cviu.2017.08.008.

Facial Landmark Detection

Table 4: A list of sources of wild databases for face alignment.

Databases	Year	#Images	#Training	#Test	#Point	Links
LFW [96]	2007	13,233	1,100	300	10	http://www.dantone.me/datasets/facial-features-lfw/
LFPW [11]	2011	1,432[a]	-	-	35[b]	http://homes.cs.washington.edu/~neeraj/databases/lfpw/
AFLW [112]	2011	25,993	-	-	21	http://lrs.icg.tugraz.at/research/aflw
AFW [41]	2012	205	-	-	6	http://www.ics.uci.edu/~xzhu/face/
HELEN [84]	2012	2,330	2,000	300	194	http://www.ifp.illinois.edu/~vuongle2/helen/
300-W [113]	2013	3,837	3,148	689	68	http://ibug.doc.ic.ac.uk/resources/300-W/
COFW [49]	2013	1,007	-	-	29	http://www.vision.caltech.edu/xpburgos/ICCV13/
MTFL [58]	2014	12,995	-	-	5	http://mmlab.ie.cuhk.edu.hk/projects/TCDCN.html
MAFL [114]	2016	20,000	-	-	5	http://mmlab.ie.cuhk.edu.hk/projects/TCDCN.html

[a] LFPW is shared by web URLs, but some URLs are no longer valid.
[b] Each face image in LFPW is annotated with 35 points, but only 29 points defined in [11] are used for the face alignment.



(a) LFW (b) LFPW (c) AFLW (d) AFW (e) HELEN (f) 300-W (g) COFW (h) MTFL

Figure 8: Illustration of the example face images from eight wide face databases with original annotation.

Jin, Xin & Tan, Xiaoyang. (2016). Face Alignment In-the-Wild: A Survey. Computer Vision and Image Understanding. 162. 10.1016/j.cviu.2017.08.008.

Facial Landmark Detection

Table 1: Categorization of the popular approaches for face alignment.

Approach	Representative works
Generative methods	
Active appearance models (AAMs)	
Regression-based fitting	Original AAM [16]; Boosted Appearance Model [17]; Nonlinear discriminative approach [18]; Accurate regression procedures for AMMs [19]
Gradient descent-based fitting	Project-out inverse compositional (POIC) algorithm [20]; Simultaneous inverse compositional (SIC) algorithm [21]; Fast AAM [22]; 2.5D AAM [23]; Active Orientation Models [24]
Part-based generative deformable models	Original Active Shape Model (ASM) [25]; Gauss-Newton deformable part model [26]; Project-out cascaded regression [27]; Active pictorial structures [28]
Discriminative methods	
Constrained local models (CLMs)[a]	
PCA shape model	Regularized landmark mean-shift [29]; Regression voting-based shape model matching [30]; Robust response map fitting [31]; Constrained local neural field [32]
Exemplar shape model	Consensus of exemplar [11]; Exemplar-based graph matching [33]; Robust Discriminative Hough Voting [34]
Other shape models	Gaussian Process Latent Variable Model [35]; Component-based discriminative search [36]; Deep face shape model [37]
Constrained local regression	Boosted regression and graph model [38]; Local evidence aggregation for regression [39]; Guided unsupervised learning for model specific models [40]
Deformable part models (DPMs)	Tree structured part model [41]; Structured output SVM [42]; Optimized part model [43]; Regressive Tree Structured Model [44]
Ensemble regression-voting	Conditional regression forests [12]; Privileged information-based conditional regression forest [45]; Sieving regression forest votes[46]; Nonparametric context modeling [47]
Cascaded regression	
Two-level boosted regression	Explicit shape regression [48]; Robust cascaded pose regression [49]; Ensemble of regression trees [50]; Gaussian process regression trees [51];
Cascaded linear regression	Supervised descent method [52]; Multiple hypotheses-based regression [53]; Local binary feature [54]; Incremental face alignment [55]; Coarse-to-fine shape search [56]
Deep neural networks[b]	
Deep CNNs	Deep convolutional network cascade [57]; Tasks-constrained deep convolutional network [58]; Deep Cascaded Regression[59]
Other deep networks	Coarse-to-fine Auto-encoder Networks (CFAN) [60]; Deep face shape model [37]

[a] Classic Constrained Local Models (CLMs) typically refer to the combination of local detector for each facial point and the parametric Point Distribution Model [61, 62, 29]. Here we extend the range of CLMs by including some methods based on other shape models (i.e., exemplar-based model [11]). In particular, we will show that the exemplar-based method [11] can also be interpreted under the conventional CLM framework.

[b] We note that some deep learning-based systems can also be placed in other categories. For instance, some systems are constructed in a cascade manner [60, 59, 63], and hence can be naturally categorized as cascaded regression. However, to highlight the increasing important role of deep learning techniques for face alignment, we organize them together for more systematic introduction and summarization.

Jin, Xin & Tan, Xiaoyang. (2016). Face Alignment In-the-Wild: A Survey. Computer Vision and Image Understanding. 162. 10.1016/j.cviu.2017.08.008.

Facial Landmark Detection

Table 2: Overview of the six classes of discriminative methods in our taxonomy.

	Appearance model	**Shape model**	**Highlights of the method**
Constrained local models	Independently trained local detector that computes a pseudo probability of the target point occurring at a particular position.	Point Distribution Moldel; Exemplar model, etc[a]	The local detectors are first correlated with the image to yield a filter response for each facial point, and then shape optimization is performed over these filter responses.
Constrained local regression	Independently trained local regressor that predicts a distance vector relating to a patch location.	Markov Random Fields to model the relations between relative positions of pairs of points.	Graph model is used to constrain the search space of local regressors by exploiting the constellations that facial points can form.
Deformable part models	Part-based appearance model that computes the appearance evidence for placing a template for a facial part.	Tree-structured models that are easier to optimize than dense graph structures.	All parameters of the appearance model and shape model are discriminatively learned in a max-margin structured prediction framework; efficient dynamic programming algorithms can be used to find globally optimal solutions.
Ensemble regression-voting	Image patches to cast votes for all facial points relating to the patch centers; Local appearance features centered at facial points.	Implicit shape constraint that is naturally encoded into the multi-output function (e.g., regression tree).	Votes from different regions are ensembled to form a robust prediction for the face shape.
Cascaded regression	Shape-indexed feature that is related to current shape estimate (e.g., concatenated image patches centered at the facial points).	Implicit shape constraint that is naturally encoded into the regressor in a cascaded learning framework.	Cascaded regression typically starts from an initial shape (e.g., mean shape), and refines the holistic shape through sequentially trained regressors.
Deep neural networks	Whole face region that is typically used to estimate the whole face shape jointly; Shape-indexed feature[b]	Implicit shape constraint that is encoded into the networks since all facial points are predicted simultaneously.	Deep network is a good choice to model the nonlinear relationship between the facial appearance and the shape update. Among others, deep CNNs have the capacity to learn highly discriminative features for face alignment.

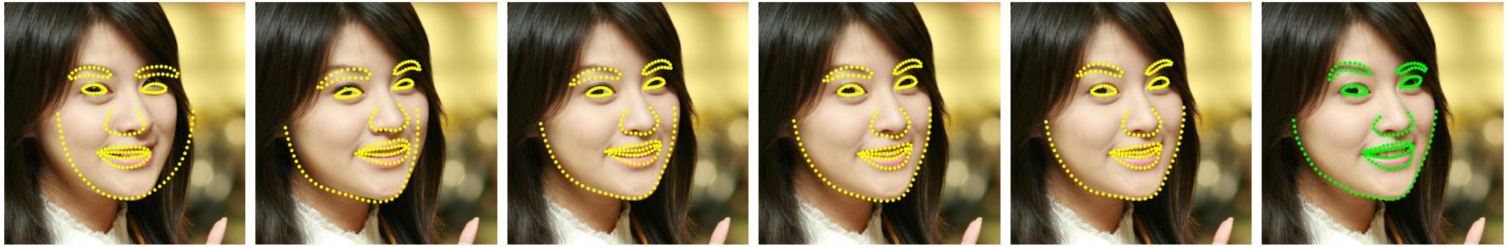
[a] Constrained Local Models (CLMs) typically employ a parametric (PCA-based) shape model [29], but we will show that the exemplar-based method [11] can also be derived from the CLM framework. Furthermore, we extend the range of CLMs by including some methods that combine independently local detector and other face shape model [35, 36, 37].
[b] Some deep network-based systems follow the cascaded regression framework, and use the shape-indexed feature [60].

V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867-1874, doi: 10.1109/CVPR.2014.241.

Jin, Xin & Tan, Xiaoyang. (2016). Face Alignment In-the-Wild: A Survey. Computer Vision and Image Understanding. 162. 10.1016/j.cviu.2017.08.008.

Cascaded Regression

- This method starts from the average face shape.
- Estimates the amount of movement based on the image features for each detected point.
- Iteratively moves the detected points to obtain the predicted face shape.



(a) $T = 0$ (b) $T = 1$ (c) $T = 2$ (d) $T = 3$ (e) $T = 10$ (f) Ground truth

Figure 2. Landmark estimates at different levels of the cascade initialized with the mean shape centered at the output of a basic Viola & Jones[17] face detector. After the first level of the cascade, the error is already greatly reduced.

V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867-1874, doi: 10.1109/CVPR.2014.241.

Jin, Xin & Tan, Xiaoyang. (2016). Face Alignment In-the-Wild: A Survey. Computer Vision and Image Understanding. 162. 10.1016/j.cviu.2017.08.008.

Facial Landmark Detection

In the training phase, the stage regressors $(\mathcal{R}^1, ..., \mathcal{R}^T)$ are sequentially learnt to reduce the alignment errors on training set, during which geometric constraints among points are *implicitly* encoded.

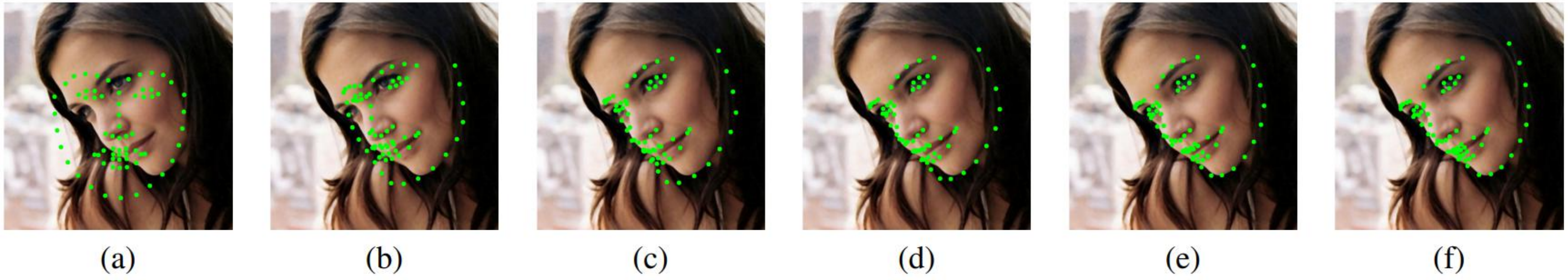


Figure 5: Illustration of face alignment results in different stages of cascaded regression (Fig. 1 in [51]). The shape estimate is initialized and iteratively updated through a cascade of regression trees: (a) initial shape estimate, (b)-(f) shape estimates at different stages.

D. Lee, H. Park and C. D. Yoo, "Face alignment using cascade Gaussian process regression trees," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4204-4212, doi: 10.1109/CVPR.2015.7299048.

V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867-1874, doi: 10.1109/CVPR.2014.241.

Jin, Xin & Tan, Xiaoyang. (2016). Face Alignment In-the-Wild: A Survey. Computer Vision and Image Understanding. 162. 10.1016/j.cviu.2017.08.008.

Facial Landmark Detection - OpenCV

In OpenCV, we can use **Facemark API and pre-trained model**

C++ Documentation:
https://docs.opencv.org/4.5.5/db/dd8/classcv_1_1face_1_1Facemark.html

Python interface:
https://gist.github.com/saiteja-talluri/1d0e4fc4c75774b936b99c7c52b65fe6

https://github.com/saiteja-talluri/GSoC-OpenCV

More about models and methods:
https://towardsdatascience.com/faster-smoother-smaller-more-accurate-and-more-robust-face-alignment-models-d8cc867efc5

Facial Landmark Detection - OpenCV

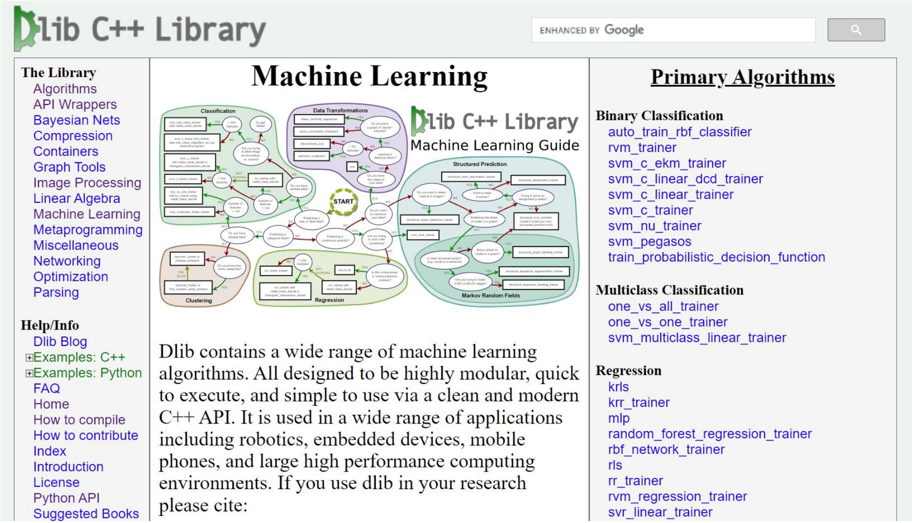
- face_python_v01 D:\vyuka\zao\face_pyt
  - dlib-19.22.99-cp310-cp310-win_amd6
  - fusek_face_car_01.avi
  - haarcascade_frontalface_alt2.xml
  - haarcascade_frontalface_default.xml
  - LBF.yaml
  - LBF555_GTX.yaml
  - LBF686_GTX.yaml
  - lbfmodel.yaml
  - lbpcascade_frontalface_improved.xml
  - main.py
- External Libraries
- Scratches and Consoles

```python
def facial_landmark():
    cv2.namedWindow("face_detect", 0)
    video_cap = cv2.VideoCapture("fusek_face_car_01.avi")
    face_cascade = cv2.CascadeClassifier("lbpcascade_frontalface_improved.xml")
    face_mark = cv2.face.createFacemarkLBF()
    face_mark.loadModel("LBF555_GTX.yaml")

    while True:
        ret, frame = video_cap.read()
        paint_frame = frame.copy()
        if ret is True:
            faces = face_cascade.detectMultiScale(frame,
                                                  scaleFactor=1.1,
                                                  minNeighbors=2,
                                                  minSize=(100, 100),
                                                  maxSize=(500, 500))

            if len(faces) > 0:
                status, landmarks = face_mark.fit(frame, faces)
                for f in range(len(landmarks)):
                    cv2.face.drawFacemarks(paint_frame, landmarks[f], (255, 255, 255))

            for one_face in faces:
                cv2.rectangle(paint_frame, one_face, (0, 0, 255), 12)
                cv2.rectangle(paint_frame, one_face, (255, 255, 255), 4)

        cv2.imshow("face_detect", paint_frame)
        if cv2.waitKey(2) == ord("q"):
            break
```

Object Detection - Dlib

Alternatively, we can use http://dlib.net/

Dlib [27] is an open-source machine learning library. Among others, it has **Ensemble of Regression Trees (ERT)** [26] facial landmark detection algorithm, which is a cascade, based on gradient boosting. The authors use a "mean" face template as an initial approximation, then the template is refined over several iterations. The algorithm requires the face to be first detected in the frame (Viola-Jones [28] face detector is used). Note, that most facial landmark detection algorithms require face to be first detected. High speed is the main advantage of ERT (according to the authors, around 1 millisecond per face). The library contains ERT implementation, trained on 300W dataset. The algorithm is still actively used in the modern research thanks to an open implementation and speed. However, not so long ago it has been shown that neural networks are preferred in terms of quality for faces with large pose [29]. Mobile-friendly implementations of ERT are available.



Khabarlak, Konstantin & Koriashkina, Larysa. (2022). Fast Facial Landmark Detection and Applications: A Survey. Journal of Computer Science and Technology. 22. 12–41. 10.24215/16666038.22.e02.

DLIB (to build in Windows, we need VS C++ and Python)

In my case, Dlib installation using pip in PyCharm works with **Python** 3.7.9

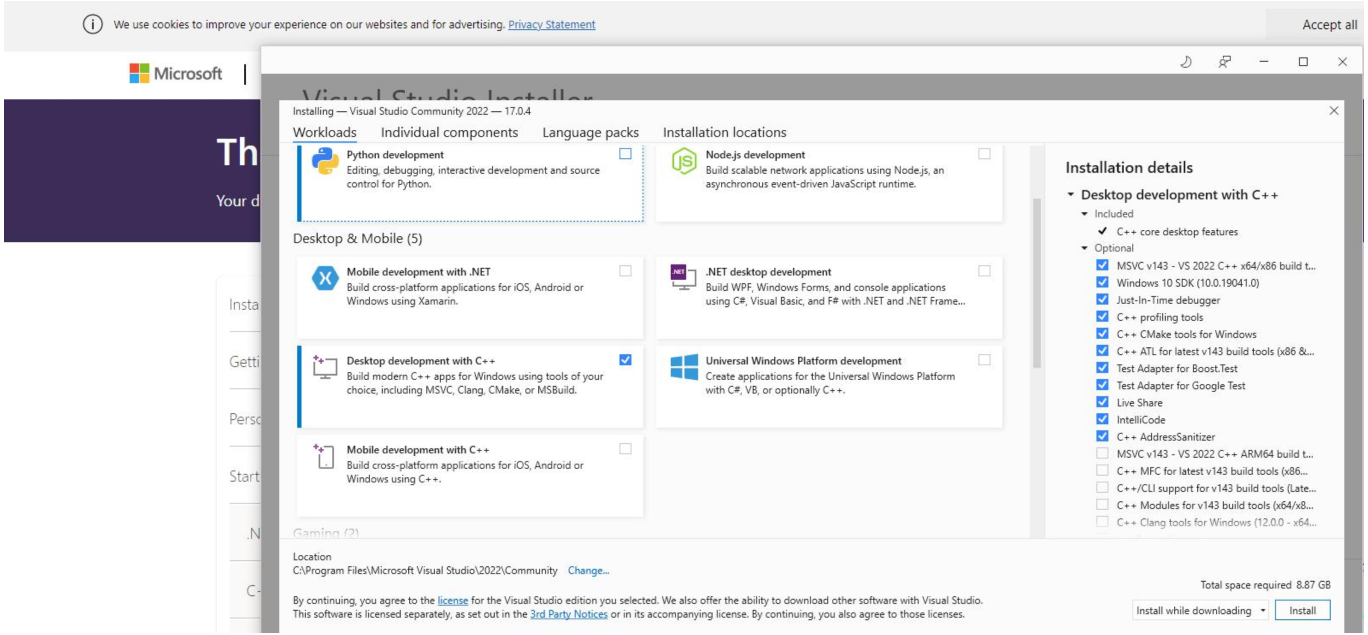
It means that you need to create a new python virtual venv with this python versio

+ pip install cmake

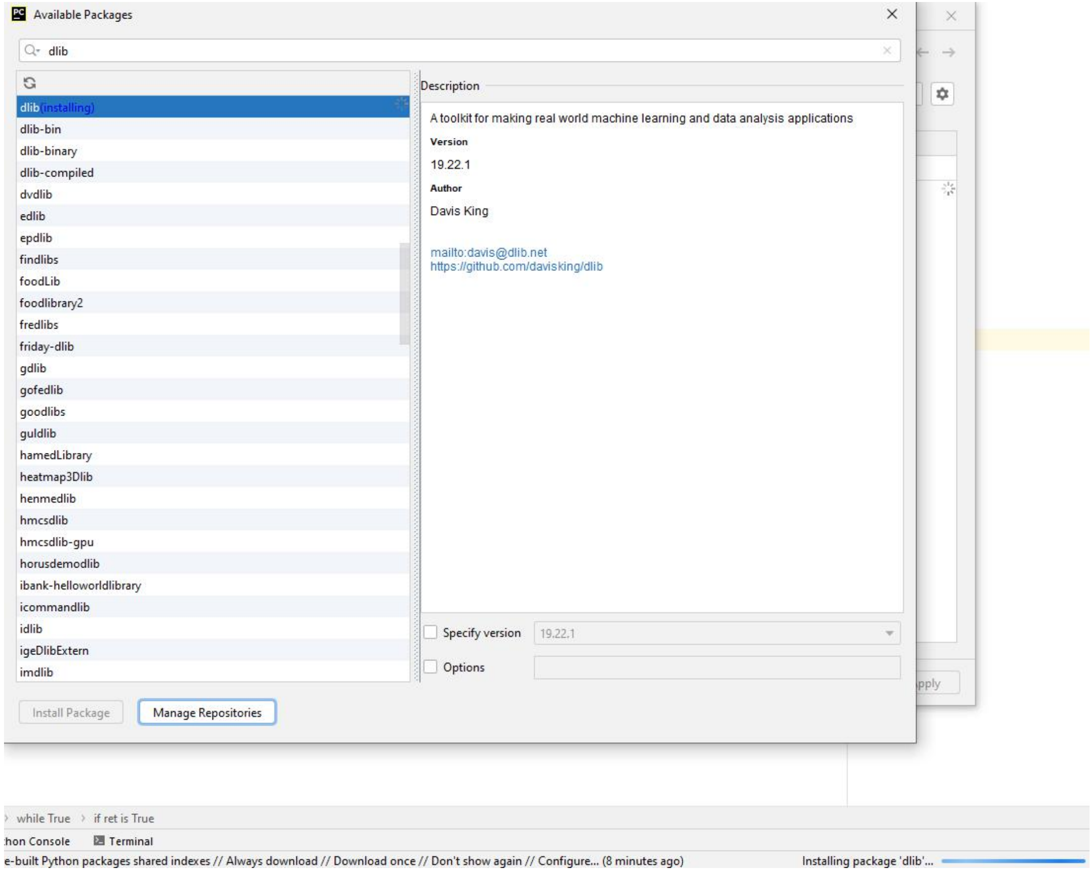
- Python 3.7.9 - Aug. 17, 2020

  Note that Python 3.7.9 *cannot* be used on Windows XP or earlier.

  - Download Windows help file
  - Download Windows x86-64 embeddable zip file
  - Download Windows x86-64 executable installer
  - Download Windows x86-64 web-based installer
  - Download Windows x86 embeddable zip file
  - Download Windows x86 executable installer
  - Download Windows x86 web-based installer



DLIB


Available Packages
dlib
dlib (installing)
dlib-bin
dlib-binary
dlib-compiled
dvdlib
edlib
epdlib
findlibs
foodLib
foodlibrary2
fredlibs
friday-dlib
gdlib
gofedlib
goodlibs
guldlib
hamedLibrary
heatmap3Dlib
henmedlib
hmcsdlib
hmcsdlib-gpu
horusdemodlib
ibank-helloworldlibrary
icommandlib
idlib
igeDlibExtern
imdlib
Description
A toolkit for making real world machine learning and data analysis applications
Version
19.22.1
Author
Davis King
mailto:davis@dlib.net
https://github.com/davisking/dlib
Specify version
19.22.1
Options
Install Package
Manage Repositories
while True
if ret is True
hon Console
Terminal
e-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (8 minutes ago)
Installing package 'dlib'...

Facial Landmark Detection Opencv + Dlib

Not secure | dlib.net

Compression
Containers
Graph Tools
Image Processing
Linear Algebra
Machine Learning
Metaprogramming
Miscellaneous
Networking
Optimization
Parsing

Help/Info

Dlib Blog
⊞Examples: C++
⊟Examples: Python
- Binary Classification
- CNN Face Detector
- Face Alignment
- Face Clustering
- Face Detector
- Face Jittering/Augmentation
- Face Landmark Detection
- Face Recognition
- Find Candidate Object Locations
- Global Optimization
- Linear Assignment Problems
- Sequence Segmenter
- Structural Support Vector Machines
- SVM-Rank
- Train Object Detector
- Train Shape Predictor
- Video Object Tracking

FAQ

environments. Dlib's open source licensing allows you to use it in any application, free of charge.

To follow or participate in the development of dlib subscribe to dlib on github. Also be sure to read the how to contribute page if you intend to submit code to the project.

To quickly get started using dlib, follow these instructions to build dlib.

Major Features

- **Documentation**
  - Unlike a lot of open source projects, this one provides complete and precise documentation for every class and function. There are also debugging modes that check the documented preconditions for functions. When this is enabled it will catch the vast majority of bugs caused by calling functions incorrectly or using objects in an incorrect manner.
  - Lots of example programs are provided
  - *I consider the documentation to be the most important part of the library*. So if you find anything that isn't documented, isn't clear, or has out of date documentation, tell me and I will fix it.
- **High Quality Portable Code**
  - Good unit test coverage. The ratio of unit test lines of code to library lines of code is about 1 to 4.
  - The library is tested regularly on MS Windows, Linux, and Mac OS X systems. However, it should work on any POSIX system and has been used on Solaris, HPUX, and the BSDs.
  - No other packages are required to use the library. Only APIs that are provided by an out of the box OS are needed.
  - There is no installation or configure step needed before you can use the library. See the How to compile page for details.

Facial Landmark Detection Opencv + Dlib

- dlib_p37 D:\vyuka\zao\dlib_p37
  - venv library root
  - fusek_face_car_01.avi
  - main.py
  - shape_predictor_68_face_landm
  - shape_predictor_68_face_landm
- External Libraries
- Scratches and Consoles

```python
def facial_landmark_dlib():
    detector = dlib.get_frontal_face_detector()
    landmark_predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
    video_cap = cv2.VideoCapture("fusek_face_car_01.avi")

    while video_cap.isOpened():
        ret, frame = video_cap.read()
        if ret is True:
            paint_frame = frame.copy()
            faces = detector(frame, 0)

            for i, f in enumerate(faces):
                pt1 = (f.left(), f.top())
                pt2 = (f.right(), f.bottom())
                print("id-top-bot {} - {} - {}".format(i, pt1, pt2))
                cv2.rectangle(paint_frame, pt1, pt2, (0, 0, 255), 12)
                cv2.rectangle(paint_frame, pt1, pt2, (255, 255, 255), 4)

                shape = landmark_predictor(frame, f)
                print(shape.parts())
                for ip, p in enumerate(shape.parts()):
                    #if ip in [20, 25, 30]:
                    point = (p.x, p.y)
                    cv2.circle(paint_frame, point, 5, (255, 255, 255), -1)
                    cv2.circle(paint_frame, point, 2, (0, 0, 0), -1)

            cv2.imshow("opencv_frame", paint_frame)
            if cv2.waitKey(2) == ord("q"):
                break
        else:
            break
```

Facial Landmark Detection MediaPipe

MediaPipe Solutions Framework Search English

Solutions

Overview Guide Examples API

Filter

Studio

Vision tasks
- Object detection
- Image classification
- Image segmentation
- Interactive segmentation
- Gesture recognition
- Hand landmark detection
- Image embedding
- Face detection
- Face landmark detection
  - Overview
  - Android
  - Web
  - Python
- Pose landmark detection
- Face stylization
- Holistic landmark detection

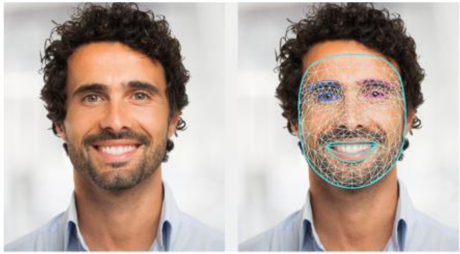
Attention: This MediaPipe Solutions Preview is an early release. Learn more

Home > MediaPipe > Solutions > Guide

Was this helpful?

Face landmark detection guide

The MediaPipe Face Landmarker task lets you detect face landmarks and facial expressions in images and videos. You can use this task to identify human facial expressions, apply facial filters and effects, and create virtual avatars. This task uses machine learning (ML) models that can work with single images or a continuous stream of images. The task outputs 3-dimensional face landmarks, blendshape scores (coefficients representing facial expression) to infer detailed facial surfaces in real-time, and transformation matrices to perform the transformations required for effects rendering.



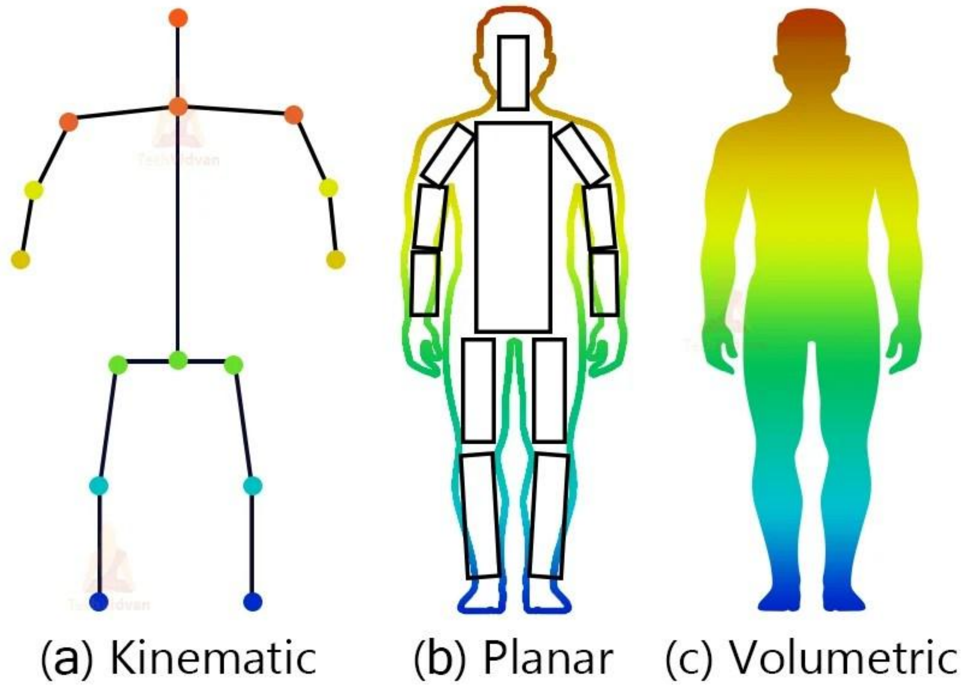
Try it! →

On this page
- Get Started
- Task details
  - Features
  - Configurations options
- Models

https://developers.google.com/mediapipe/solutions/vision/face_landmarker#models

https://github.com/googlesamples/mediapipe/tree/main/examples

Types of Human Pose Estimation Models

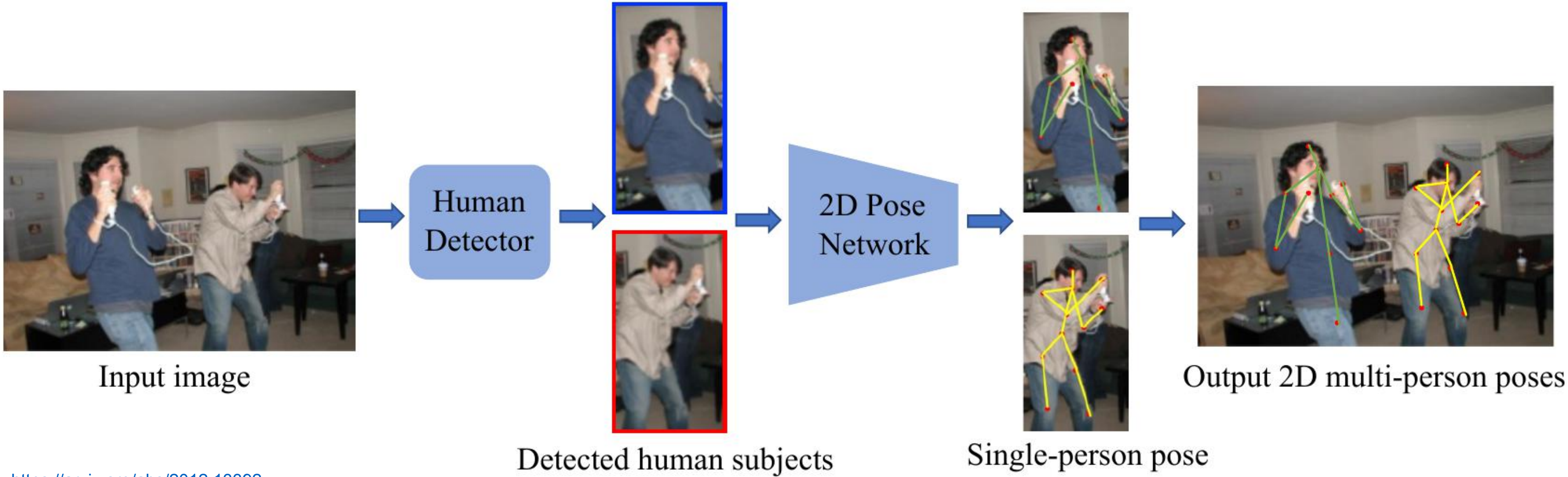


https://arxiv.org/abs/2012.13392
https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/

Human Pose Estimation

2D multi-person pose estimation: top-down and **bottom-up** methods.

Top-down approaches have two sub-tasks:
(1) human detection and
(2) pose estimation in the region of a single human.



(a) Top-Down Approaches

https://arxiv.org/abs/2012.13392
https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/

Human Pose Estimation

2D multi-person pose estimation: top-down and **bottom-up** methods.

Bottom-up approaches also have two sub-tasks:
(1) detect all keypoints candidates of body parts and
(2) associate body parts in different human bodies and assemble them into individual pose representations.

(b) Bottom-Up Approaches

https://arxiv.org/abs/2012.13392
https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/