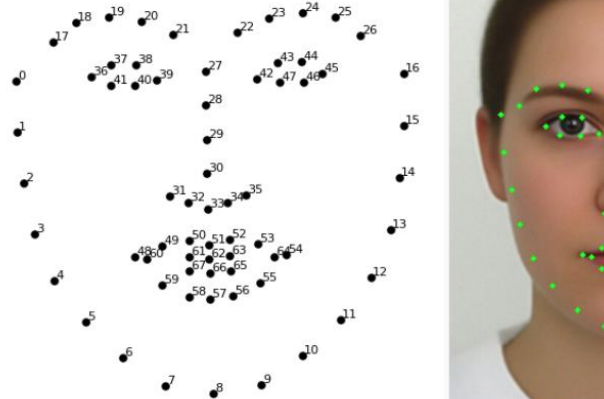


Facial Landmark Detection (keypoint detection)

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



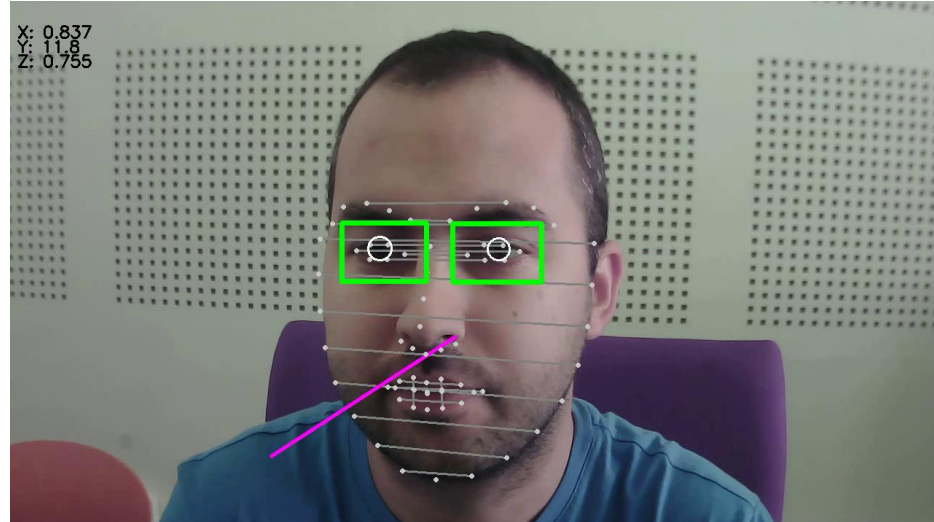
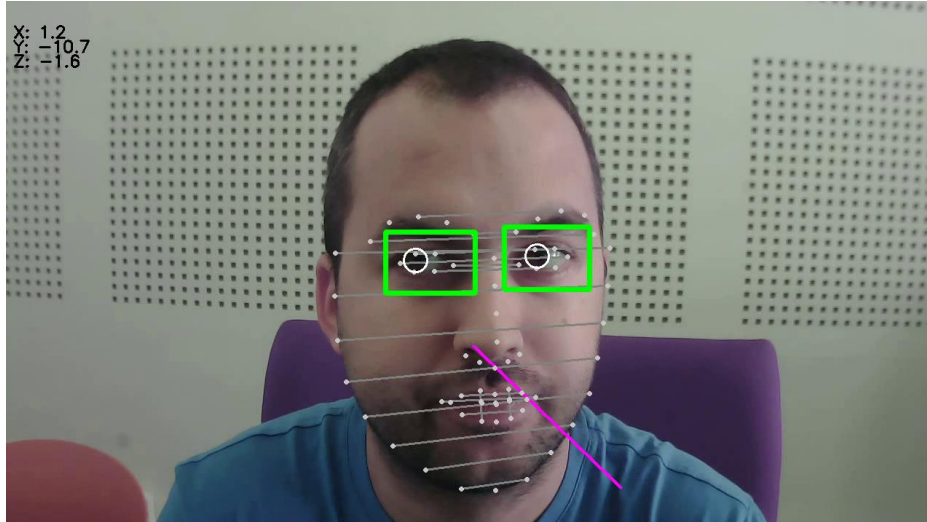
Applications of Facial landmark detection (keypoint detection)

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



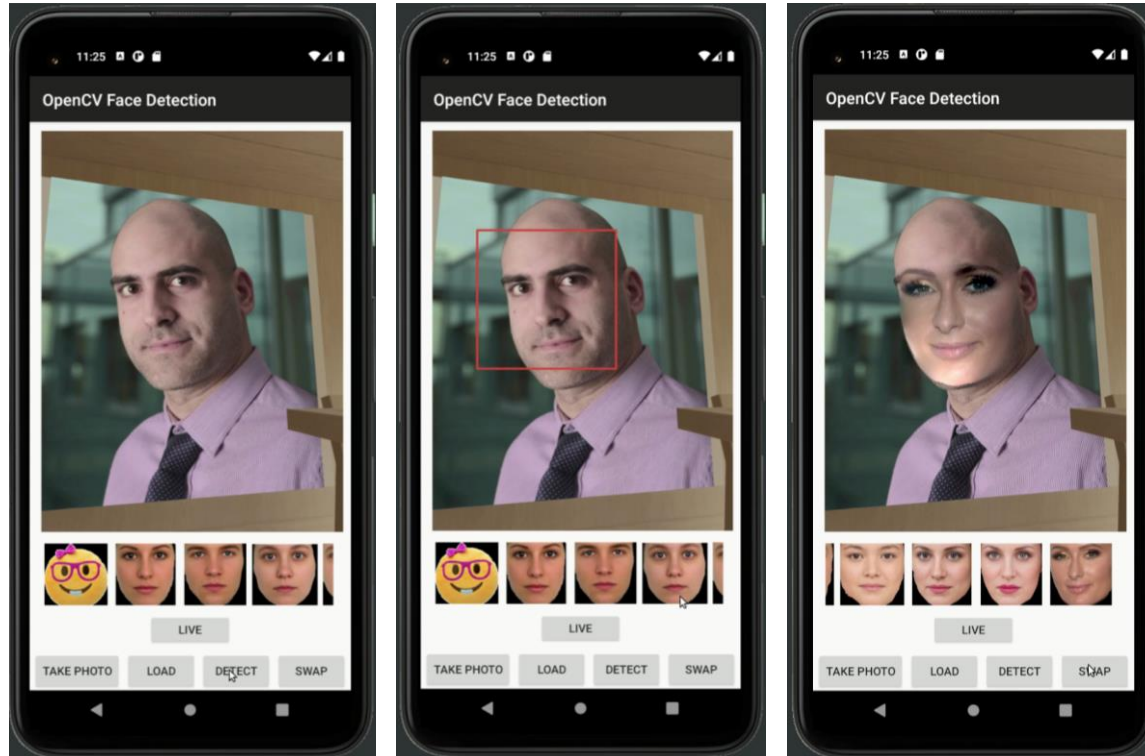
Applications of Facial landmark detection (keypoint detection)

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



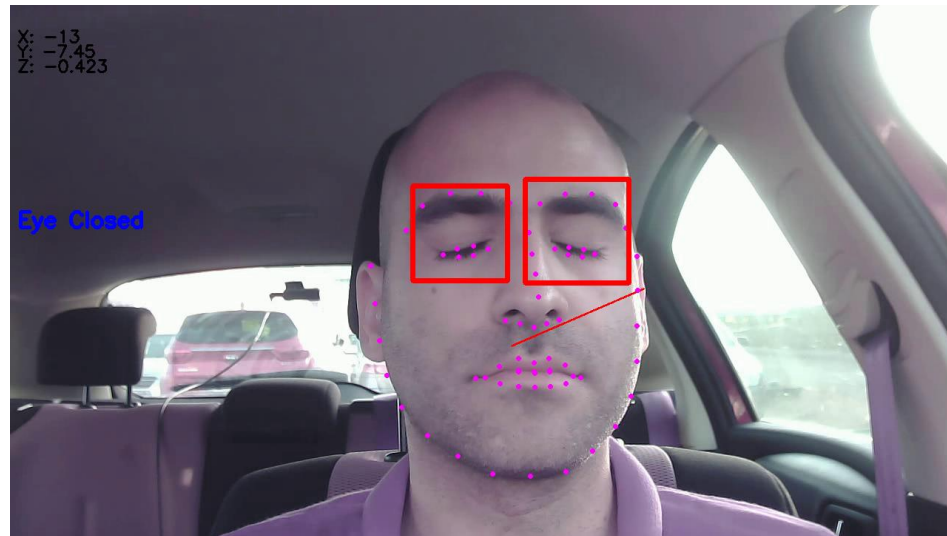
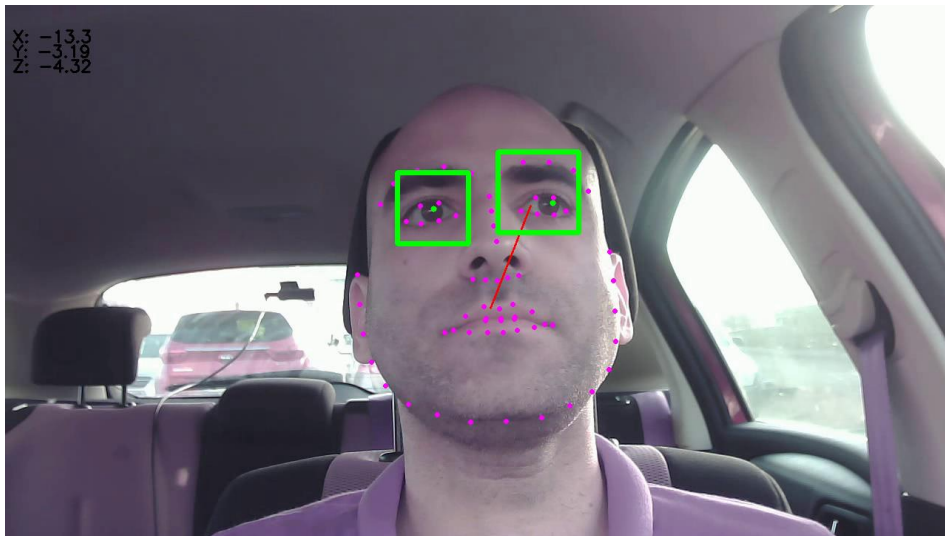
Applications of Facial landmark detection (keypoint detection)

- 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)

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.

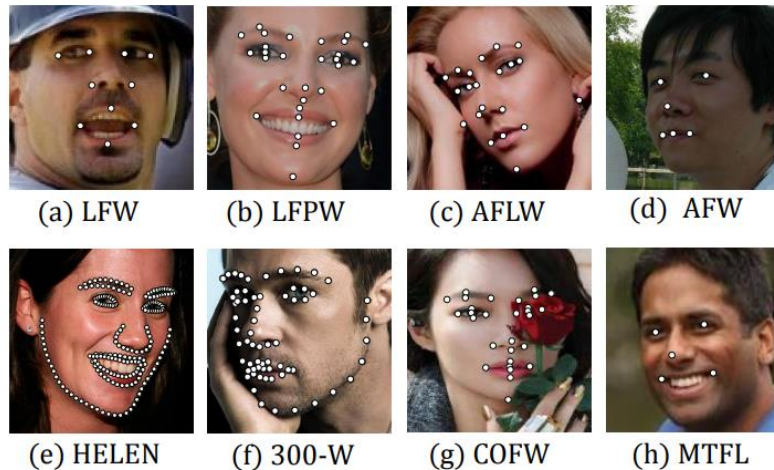


Figure 8: Illustration of the example face images from eight wide face databases with original annotation.

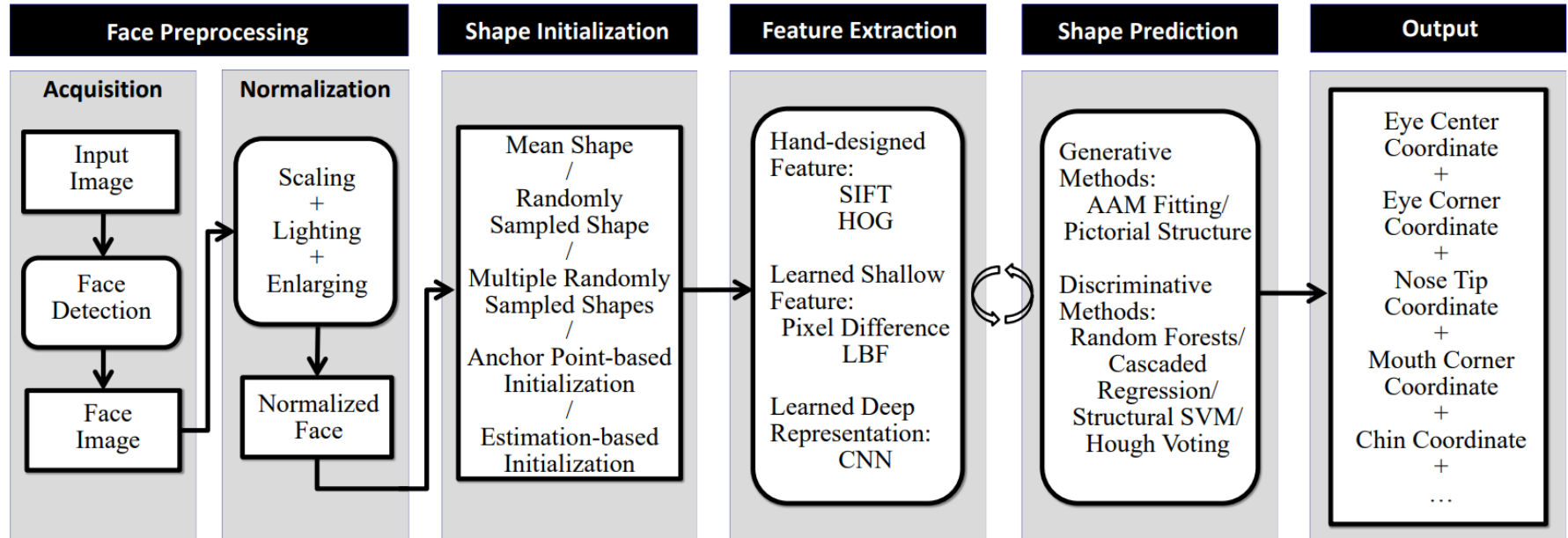


Figure 7: A global system architecture for face alignment.

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

Approach	Representative works
Generative methods	
<i>Active appearance models (AAMs)</i>	
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]
<i>Part-based generative deformable models</i>	Original Active Shape Model (ASM) [25]; Gauss-Newton deformable part model [26]; Project-out cascaded regression [27]; Active pictorial structures [28]
Discriminative methods	
<i>Constrained local models (CLMs)^a</i>	
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]
<i>Constrained local regression</i>	Boosted regression and graph model [38]; Local evidence aggregation for regression [39]; Guided unsupervised learning for model specific models [40]
<i>Deformable part models (DPMs)</i>	Tree structured part model [41]; Structured output SVM [42]; Optimized part model [43]; Regressive Tree Structured Model [44]
<i>Ensemble regression-voting</i>	Conditional regression forests [12]; Privileged information-based conditional regression forest [45]; Sieving regression forest votes [46]; Nonparametric context modeling [47]
<i>Cascaded regression</i>	
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]
<i>Deep neural networks^b</i>	
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 independent 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.

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

Approach	Representative works
Generative methods	
<i>Active appearance models (AAMs)</i>	
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]
<i>Part-based generative deformable models</i>	
Original Active Shape Model (ASM) [25]; Gauss-Newton deformable part model [26]; Project-out cascaded regression [27]; Active pictorial structures [28]	
Discriminative methods	
<i>Constrained local models (CLMs)^a</i>	
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]
<i>Constrained local regression</i>	
Boosted regression and graph model [38]; Local evidence aggregation for regression [39]; Guided unsupervised learning for model specific models [40]	
<i>Deformable part models (DPMs)</i>	
Tree structured part model [41]; Structured output SVM [42]; Optimized part model [43]; Regressive Tree Structured Model [44]	
<i>Ensemble regression-voting</i>	
Conditional regression forests [12]; Privileged information-based conditional regression forest [45]; Sieving regression forest votes [46]; Nonparametric context modeling [47]	
<i>Cascaded regression</i>	
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]
<i>Deep neural networks^b</i>	
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.

Early Landmark Detection Algorithms

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

	Appearance model	Shape model	Highlights of the method
<i>Constrained local models</i>	Independently trained local detector that computes a pseudo probability of the target point occurring at a particular position.	Point Distribution Model; 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.
<i>Constrained local regression</i>	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.
<i>Deformable part models</i>	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.
<i>Ensemble regression-voting</i>	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 ensemble to form a robust prediction for the face shape.
<i>Cascaded regression</i>	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.
<i>Deep neural networks</i>	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].

Early Landmark Detection Algorithms

Available for example in:
OpenCV or Dlib

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.

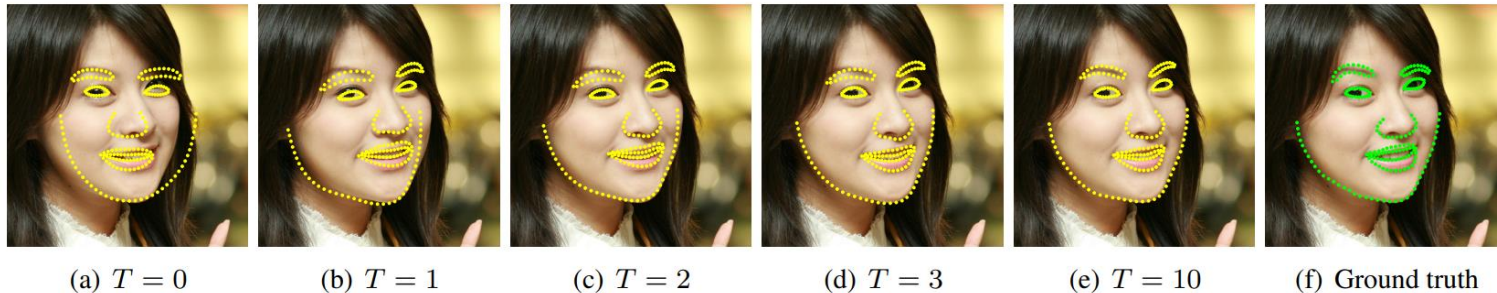


Figure 2. Landmark estimates at different levels of the cascade initialized with the mean shape centered at the output of a basic Viola & Jones[17] face detector. After the first level of the cascade, the error is already greatly reduced.

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

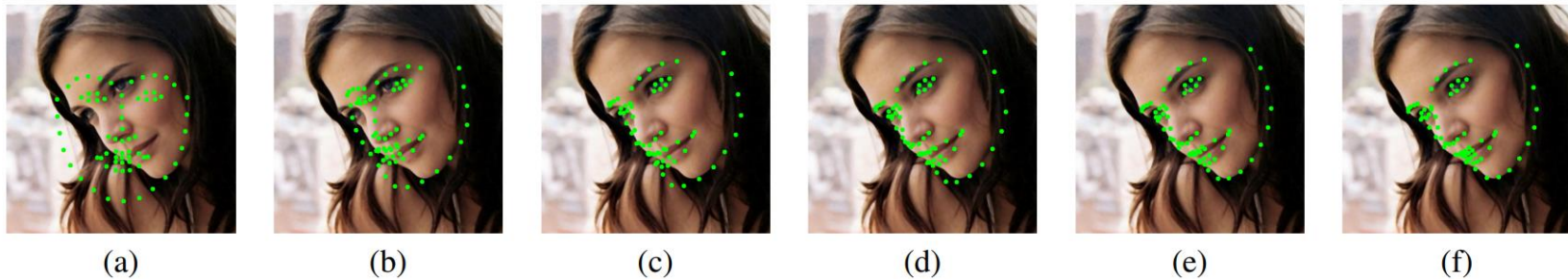


Figure 5: Illustration of face alignment results in different stages of cascaded regression (Fig. 1 in [51]). The shape estimate is initialized and iteratively updated through a cascade of regression trees: (a) initial shape estimate, (b)-(f) shape estimates at different stages.

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>

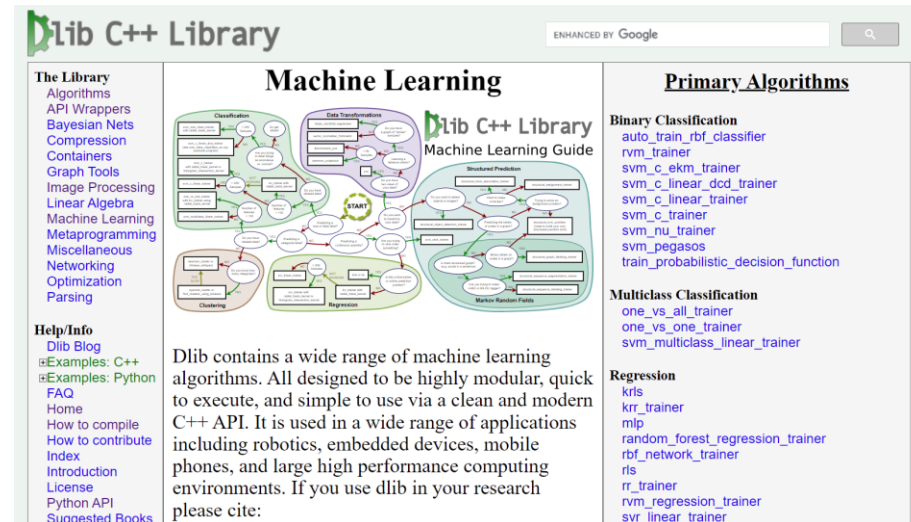
```

45 def facial_landmark():
46     cv2.namedWindow("face_detect", 0)
47     video_cap = cv2.VideoCapture("fusek_face_car_01.avi")
48     face_cascade = cv2.CascadeClassifier("lbpcascade_frontalface_improved.xml")
49     face_mark = cv2.face.createFacemarkLBF()
50     face_mark.loadModel("LBF555_6TX.yaml")
51
52 while True:
53     ret, frame = video_cap.read()
54     paint_frame = frame.copy()
55     if ret is True:
56         faces = face_cascade.detectMultiScale(frame,
57                                             scaleFactor=1.1,
58                                             minNeighbors=2,
59                                             minSize=(100, 100),
60                                             maxSize=(500, 500))
61
62         if len(faces) > 0:
63             status, landmarks = face_mark.fit(frame, faces)
64             for f in range(len(landmarks)):
65                 cv2.face.drawFacemarks(paint_frame, landmarks[f], (255, 255, 255))
66
67         for one_face in faces:
68             cv2.rectangle(paint_frame, one_face, (0, 0, 255), 12)
69             cv2.rectangle(paint_frame, one_face, (255, 255, 255), 4)
70
71     cv2.imshow("face_detect", paint_frame)
72     if cv2.waitKey(2) == ord("q"):
73         break

```

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.



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

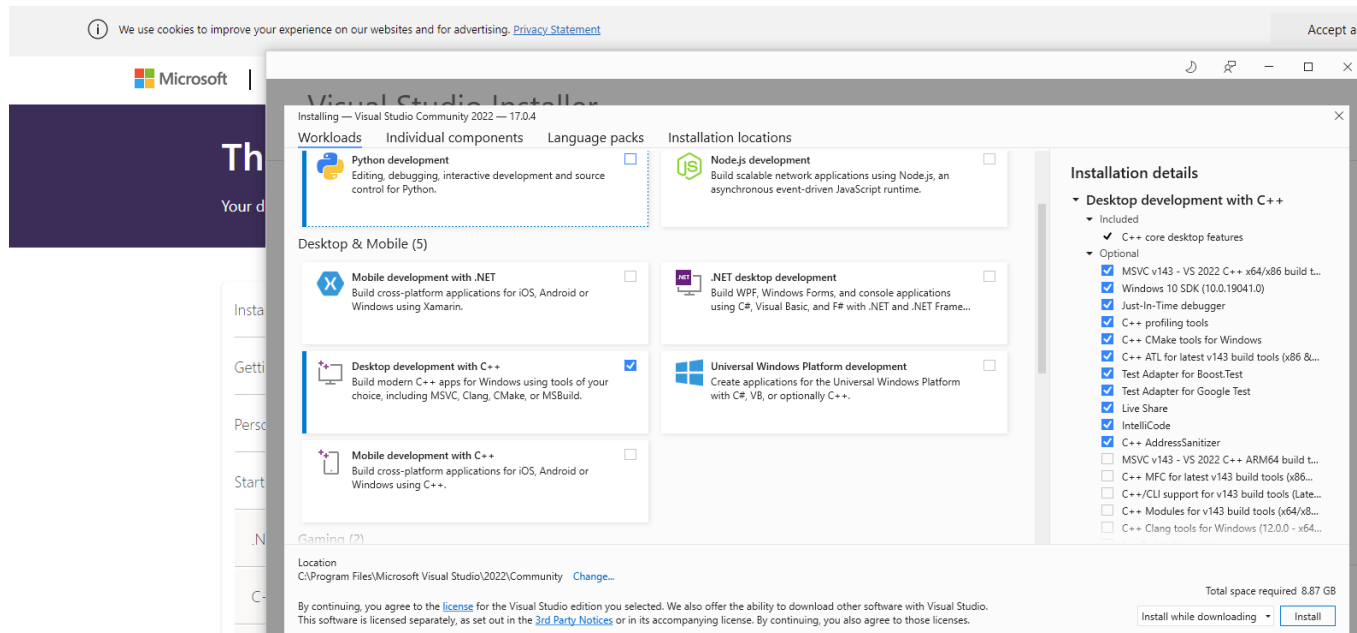
In my case, Dlib installation using pip in PyCharm works with **Python 3.7.9**

It means that you need to create a new python virtual venv with this python version
+ pip install cmake

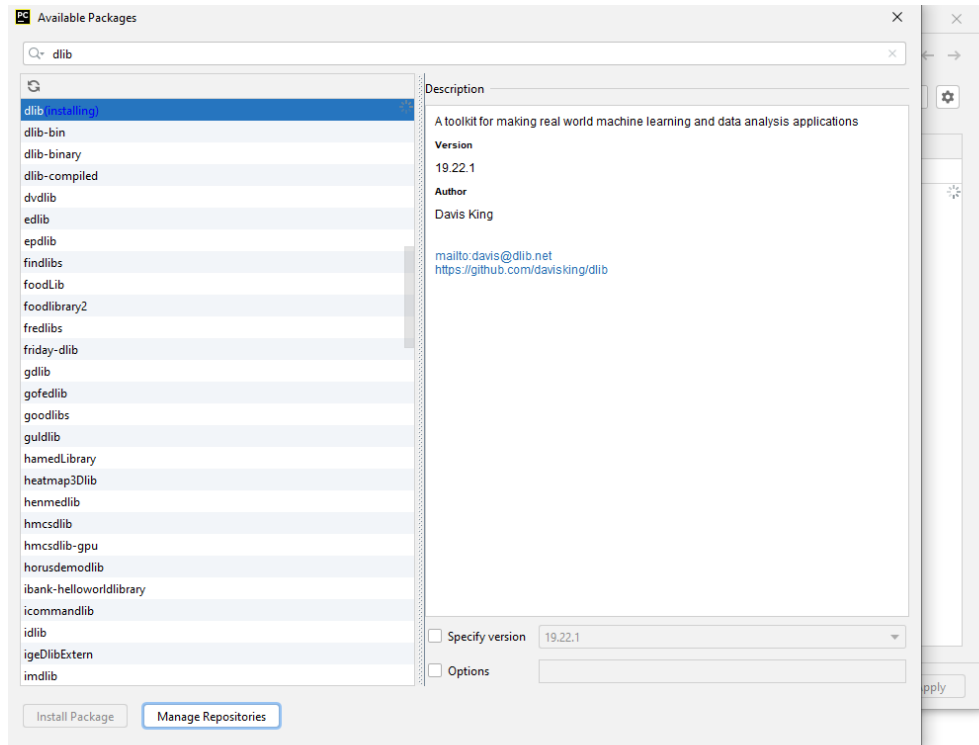
Python 3.7.9 - Aug. 17, 2020

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



```
> while True > if ret is True
hon Console [x] Terminal
e-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (8 minutes ago)
```

Installing package '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, HP/UX, 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.

```

5  def facial_landmark_dlib():
6      detector = dlib.get_frontal_face_detector()
7      landmark_predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
8      video_cap = cv2.VideoCapture("fusek_face_car_01.avi")
9
10
11     while video_cap.isOpened():
12         ret, frame = video_cap.read()
13         if ret is True:
14             paint_frame = frame.copy()
15             faces = detector(frame, 0)
16
17             for i, f in enumerate(faces):
18                 pt1 = (f.left(), f.top())
19                 pt2 = (f.right(), f.bottom())
20                 print("id-top-bot {} - {} - {}".format(i, pt1, pt2))
21                 cv2.rectangle(paint_frame, pt1, pt2, (0, 0, 255), 12)
22                 cv2.rectangle(paint_frame, pt1, pt2, (255, 255, 255), 4)
23
24                 shape = landmark_predictor(frame, f)
25                 print(shape.parts())
26                 for ip, p in enumerate(shape.parts()):
27                     #if ip in [20, 25, 30]:
28                     point = (p.x, p.y)
29                     cv2.circle(paint_frame, point, 5, (255, 255, 255), -1)
30                     cv2.circle(paint_frame, point, 2, (0, 0, 0), -1)
31
32                 cv2.imshow("opencv_frame", paint_frame)
33                 if cv2.waitKey(2) == ord("q"):
34                     break
35             else:
36                 break

```

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

Approach	Representative works
Generative methods	
<i>Active appearance models (AAMs)</i>	
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]
<i>Part-based generative deformable models</i>	
Original Active Shape Model (ASM) [25]; Gauss-Newton deformable part model [26]; Project-out cascaded regression [27]; Active pictorial structures [28]	
Discriminative methods	
<i>Constrained local models (CLMs)^a</i>	
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]
<i>Constrained local regression</i>	
Boosted regression and graph model [38]; Local evidence aggregation for regression [39]; Guided unsupervised learning for model specific models [40]	
<i>Deformable part models (DPMs)</i>	
Tree structured part model [41]; Structured output SVM [42]; Optimized part model [43]; Regressive Tree Structured Model [44]	
<i>Ensemble regression-voting</i>	
Conditional regression forests [12]; Privileged information-based conditional regression forest [45]; Sieving regression forest votes [46]; Nonparametric context modeling [47]	
<i>Cascaded regression</i>	
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]
<i>Deep neural networks^b</i>	
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.

Modern Landmark Detection Algorithms

Modern in-the-wild face landmark detection algorithms are based on neural networks. They are divided into 2 main categories: *direct* (or *coordinate*) regression methods, when the model predicts x , y coordinates directly for each landmark; *heatmap*-based regression methods, where a 2D heatmap is built for each landmark. The values in the heatmap can be interpreted as probabilities of landmark location at a certain image location. Typically, argmax or its modification is used to acquire exact landmark coordinates from the heatmap. Fig. 2 illustrates the two approaches. As neural network architectures have become more complex, algorithms typically base on a pre-defined network architecture (called backbone). Facial landmark detection algorithms, described in the following subsection, propose modifications to training, inference procedure or the backbone itself. Here we introduce main backbones for landmark detection problem. Note, that in many cases backbones for face landmark and human pose (whole body) landmark detection are the same. Direct regression methods typically use widely known backbones from ImageNet challenge [31], such as ResNet [32], MobileNetV2 [2], MobileNetV3 [33], ShuffleNet-V2 [34]. Heatmap-based methods commonly use Hourglass [35] network architecture, but

also HRNet [36] and CU-Net [37]. Such backbones are less known. Thus, we describe them here.

Hourglass [35] architecture has been initially designed for human pose estimation. The network takes a 256×256 image as an input. The authors note that processing the image at full resolution would require too much computation and memory. This is why a convolutional block is used to quickly process the image to obtain feature map of resolution 64×64 , which remains the maximum feature map resolution till the end of the network. The feature map is then processed via Hourglass modules. An illustration of Hourglass network is shown in Fig. 3. Note, that architecture allows *stacking*, i.e., Hourglass modules can be repeated sequentially multiple times. Typically stacks of 1, 2 or 4 Hourglass modules are used. The network outputs heatmaps at a resolution of 64×64 , a single heatmap is produced for each of the landmarks.

Hourglass module follows encoder-decoder architecture. Input image is processed via convolutional blocks at different feature map resolutions. First, feature map resolution is decreased after each convolutional block (encoder part), then feature map resolution is restored (increased) after each block (decoder part). Accuracy of human pose estimation, facial landmark detection and several other tasks is improved by processing image at multiple resolutions.

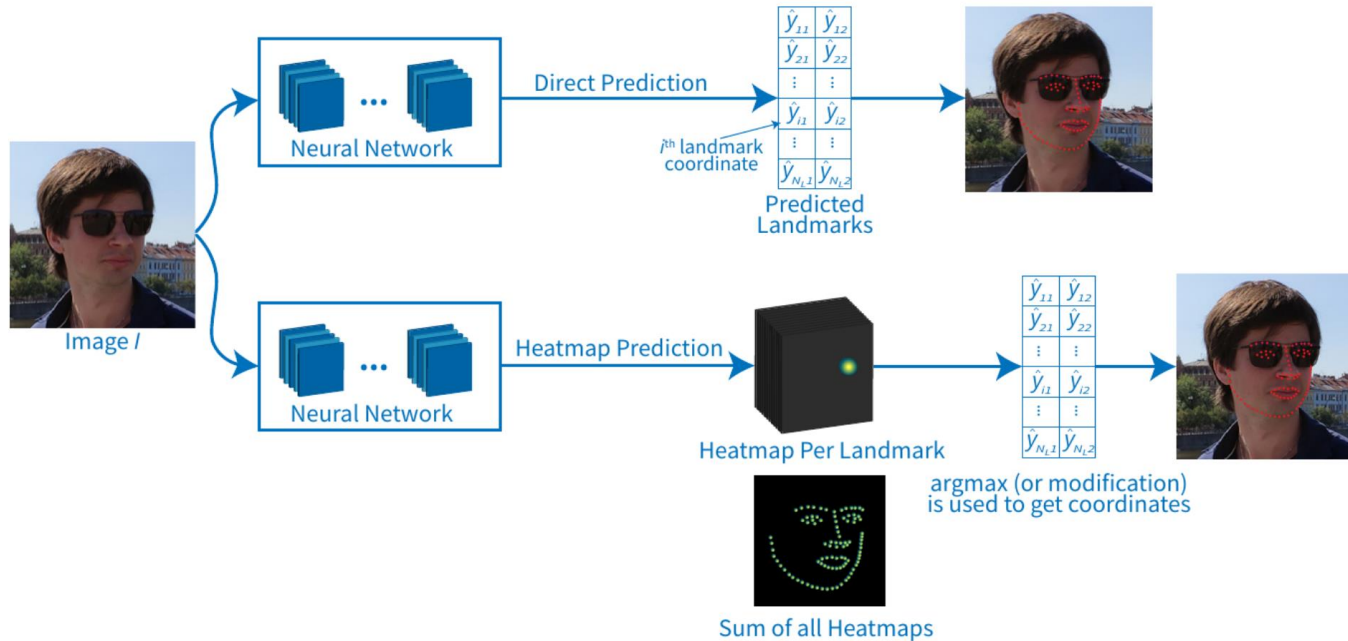


Figure 2. Direct landmark regression (upper row). The problem is solved in a form of regression, where actual landmark coordinates (x, y) are predicted directly by the algorithm. Heatmap-based (bottom row). The algorithm predicts probability distributions of landmark locations in a form of heatmaps.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


Table 4. Face landmark detection normalized mean error (NME) on 300-W dataset. Inter-ocular normalization is used. The best result is shown in red, second best in blue. Note, that significant qualitative improvement has been achieved over the past few years, but still Challenge subset error is quite high.

Model	Year	Common	Challenge	Full
DeFA [39]	2017	5.37	9.38	6.10
SAN [40]	2018	3.34	6.60	3.98
LAB [5]		2.98	5.19	3.49
AVS [16]	2019	3.21	6.49	3.86
PFLD 0.25X [46]		3.03	5.15	3.45
PFLD 1X		3.01	5.08	3.40
PFLD 1X+ (extra data)		2.96	4.98	3.37
AWing-1HG [7]		2.81	4.72	3.18
AWing-2HG		2.77	4.58	3.12
AWing-3HG		2.73	4.58	3.10
AWing		2.72	4.52	3.07
MobileFAN (0.5) [49]	2020	4.22	6.87	4.74
MobileFAN		2.98	5.34	3.45
GEAN (extra data) [50]		2.68	4.71	3.05
HRNetV2 [38]		2.87	5.15	3.32
LUVLi [6]		2.76	5.16	3.23
DAG [52]		2.62	4.77	3.04
PropagationNet [54]		2.67	3.99	2.93
SAAT [14]		2.87	5.03	3.29
LDDMM-Face [56]	2021	3.07	5.40	3.53
AnchorFace [59]		3.12	6.19	3.72
PIPNet (MobileNetV2) [60]		2.94	5.30	3.40
PIPNet (MobileNetV3)		2.94	5.07	3.36
PIPNet (ResNet-18)		2.91	5.18	3.36
PIPNet (ResNet-101)		2.78	4.89	3.19
ADNet [61]		2.53	4.58	2.93
HIH _C [62]		2.95	5.04	3.36
HIH _T		2.93	5.00	3.33
SubpixelHeatmap [63]		2.61	4.13	2.94


Modern Landmark Detection Algorithms

Facial Landmark Detection MediaPipe

Modern Landmark Detection Algorithms

 MediaPipe

[Solutions](#)
[Framework](#)

 English

Solutions

[Overview](#)
[Guide](#)
[Examples](#)
[API](#)



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](#)

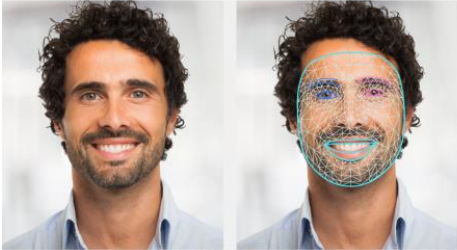
Home > MediaPipe > Solutions > Guide

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.

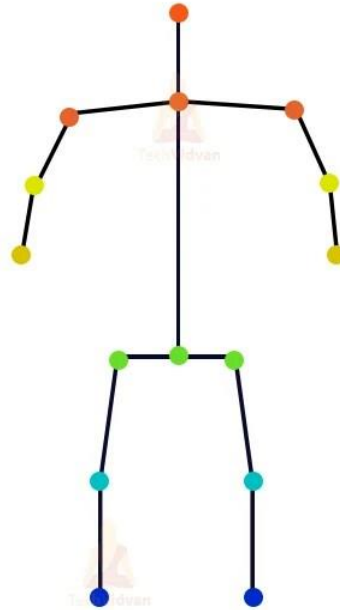
[Try it!](#)



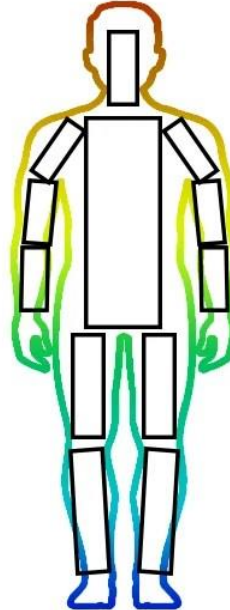
On this page

- [Get Started](#)
- Task details
- Features
- Configurations options
- Models

Types of Human Pose Estimation Models



(a) Kinematic



(b) Planar

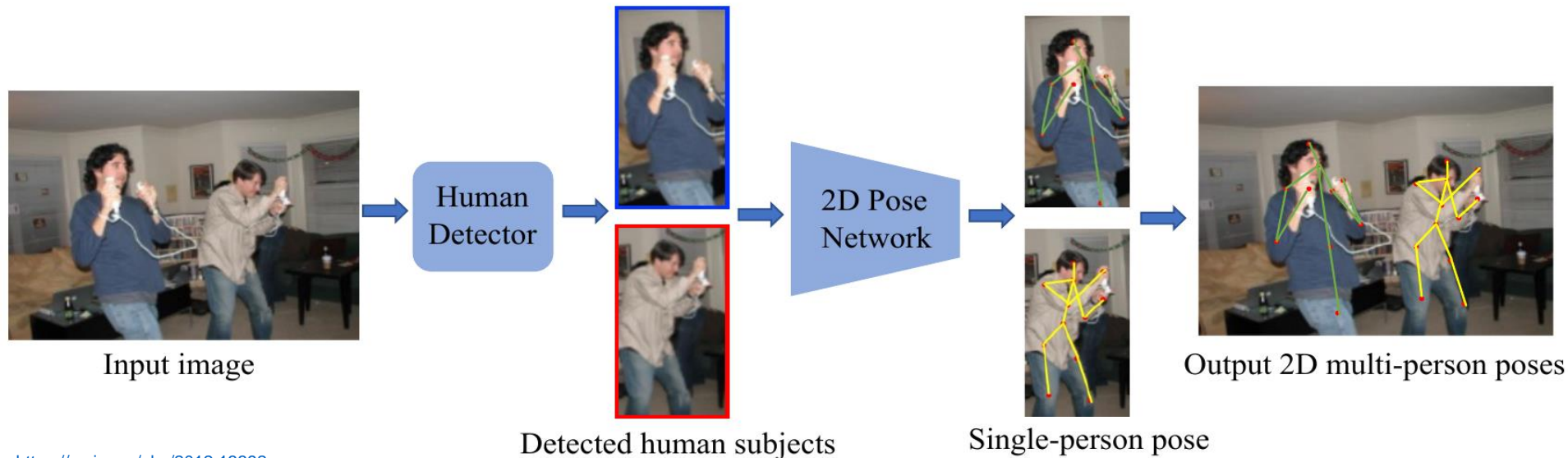


(c) Volumetric

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

Top-down approaches have two sub-tasks:

- (1) human detection
- (2) pose estimation in the region of a single human.



(a) Top-Down Approaches

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
- (2) associate body parts in different human bodies and assemble them into individual pose representations.



(b) Bottom-Up Approaches

Google AI for Developers Models Solutions Code assistance Showcase Community Search

Google AI Edge MediaPipe LiteRT Model Explorer API Reference

Filter

Vision tasks

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

Introducing LiteRT: Google's high-performance runtime for on-device AI, formerly known as TensorFlow Lite. Learn more

Home > Google AI Edge > Solutions

Was this helpful?

Pose landmark detection guide

The MediaPipe Pose Landmarker task lets you detect landmarks of human bodies in an image or video. You can use this task to identify key body locations, analyze posture, and categorize movements. This task uses machine learning (ML) models that work with single images or video. The task outputs body pose landmarks in image coordinates and in 3-dimensional world coordinates.

Try it! →

Get Started

Send feedback

Where to go from here?

Table 1. Applications of PoseNet, MoveNet, OpenPose, and MediaPipe Pose in different domains.

Domain	HPE Library	Year	Purpose of Application
Video Surveillance	OpenPose [6]	2018	Kidnapping detection—using HPE to classify kidnapping cases and normal cases in an intelligent video surveillance system.
	OpenPose [5]	2019	A child abuse prevention decision-support system—using OpenPose to classify adults and children in CCTV.
Medical Assistance	OpenPose [15]	2020	A fall detection system—using OpenPose to extract features of human body.
	PoseNet [8]	2020	Automatic feedback on incorrect posture for physiotherapy exercises.
	PoseNet [10]	2021	A telehealth system providing in-home rehabilitation.
	OpenPose [11]	2021	Measure joint angles and conduct semi-automatic ergonomic postural assessments to evaluate the risk of musculoskeletal disorders.
	MoveNet [14]	2021	A healthcare system that measures patient's strength, balance, and range of motion during physical therapy activities.
	MediaPipe Pose [12]	2022	A fall detection system.
	MediaPipe Pose [13]	2022	A posture corrector system—to notify people who are spending most of their time sitting in front of the computer with bad posture to avoid long-term health issues.
Sport Motion Analysis	OpenPose [28]	2018	A basketball free-throw shooting prediction system—using OpenPose to generate body keypoints.
	OpenPose [21]	2020	A real-time push-up counter—to classify the correct and incorrect push-ups.
	OpenPose [20]	2021	A system to evaluate baseball swinging poses and help baseball players correct their poses.
	MediaPipe Pose [24]	2021	A mobile application—to analyze, improve, and track cricket players' batting performance.
	PoseNet [25]	2021	A real-time workout analyzer—allows fitness enthusiast to perform their workout accurately at home and with proper guidance.
	PoseNet [26]	2021	A fitness tutor—to maintain the correctness of the posture during workout exercises.
	PoseNet [27]	2021	A fitness application—provides instant feedback to users to ensure the accuracy of their workout exercise poses.
	MediaPipe Pose [22]	2022	To score the human body's balance ability on the wobble board.
	MediaPipe Pose [23]	2022	A free weight exercise tracking software—allows users to learn and correct their exercise poses.

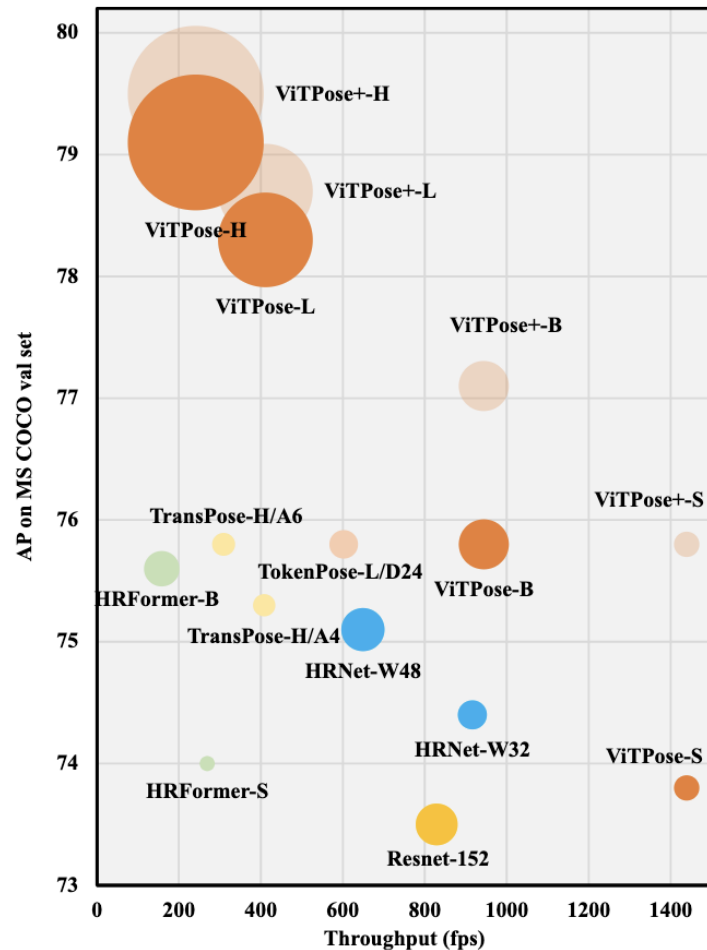
Where to go from here?

<https://github.com/ViTAE-Transformer/ViTPose>

https://huggingface.co/docs/transformers/en/model_doc/vitpose



Human Pose Estimation



Where to go from here?

<https://github.com/ViTAE-Transformer/ViTPose>

https://huggingface.co/docs/transformers/en/model_doc/vitpose

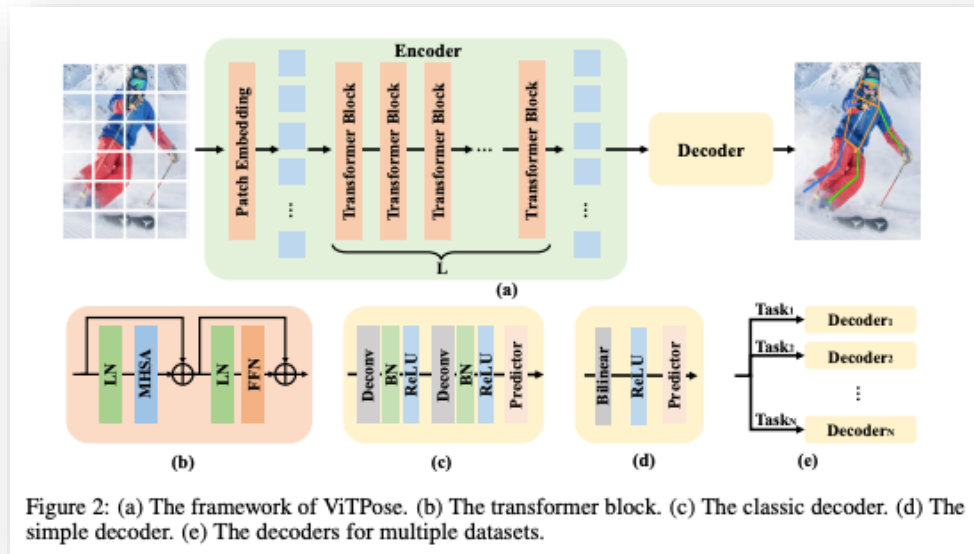


Figure 2: (a) The framework of ViTPose. (b) The transformer block. (c) The classic decoder. (d) The simple decoder. (e) The decoders for multiple datasets.

Human Pose Estimation

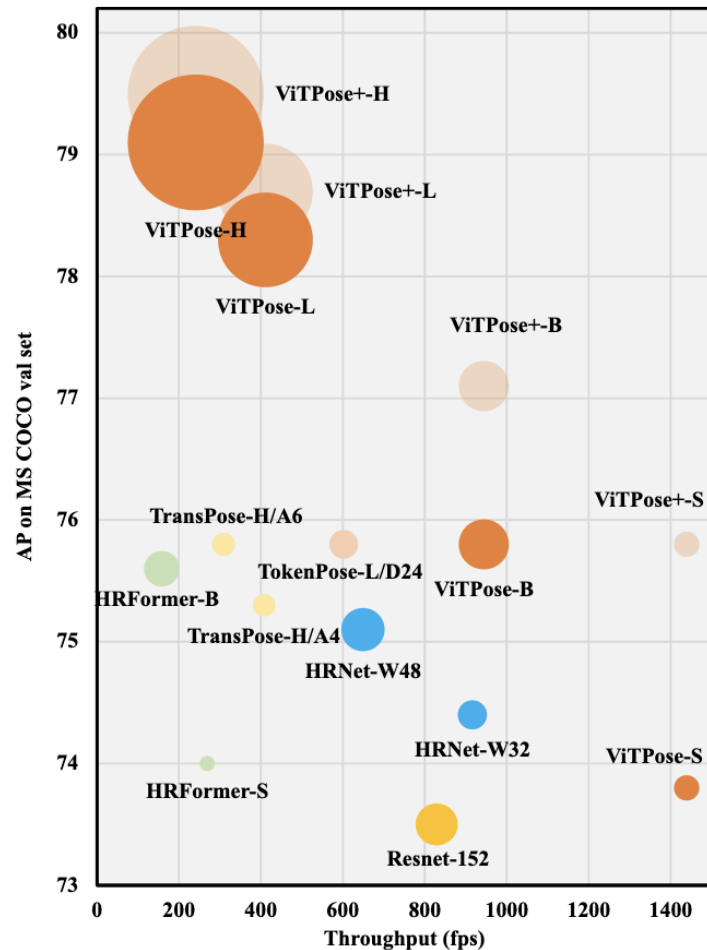


Fig 2. Using YOLO11 to detect and estimate the poses of people in an office.

Where to go from here?

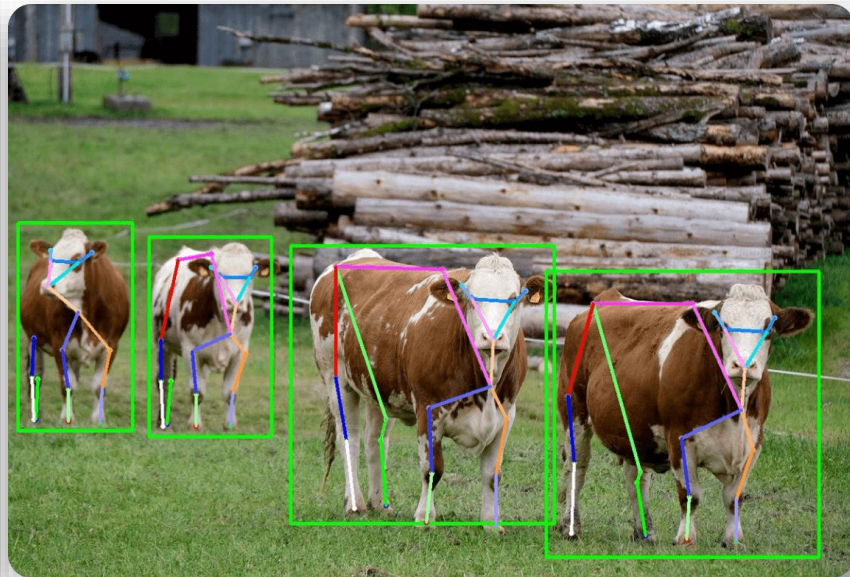
<https://www.ultralytics.com/blog/how-to-use-ultralytics-yolo11-for-pose-estimation>

Pose estimation with YOLO11 for livestock monitoring

Farmers and researchers can use YOLO11 to study the movement and behavior of farm animals, like cattle, to detect early signs of diseases such as lameness. Lameness is a condition where an animal struggles to move properly due to pain in its legs or feet. In cattle, illnesses like lameness not only affect their health and welfare but also lead to production issues on dairy farms. Studies show that lameness affects between 8% of cattle in pasture-based systems and 15% to 30% in confined systems across the global dairy industry. Detecting and addressing lameness early can help improve animal welfare and reduce the production losses associated with this condition.

YOLO11's pose estimation features can help farmers track the animal's gait patterns and quickly identify any abnormalities that might signal health problems, such as joint issues or infections. Catching these problems early allows for faster treatment, reducing the animals' discomfort and helping farmers avoid economic losses.

Vision AI enabled monitoring systems can also help analyze resting behavior, social interactions, and feeding patterns. Farmers can also use pose estimation to get observations on signs of stress or aggression. These insights can be used to cultivate better living conditions for animals and increase their well-being.



Where to go from here?

<https://github.com/ultralytics/ultralytics>

Ultralytics supports a wide range of YOLO models, from early versions like [YOLOv3](#) to the latest [YOLO11](#). The tables below showcase YOLO11 models pretrained on the [COCO](#) dataset for [Detection](#), [Segmentation](#), and [Pose Estimation](#). Additionally, [Classification](#) models pretrained on the [ImageNet](#) dataset are available. [Tracking](#) mode is compatible with all Detection, Segmentation, and Pose models. All [Models](#) are automatically downloaded from the latest Ultralytics [release](#) upon first use.

