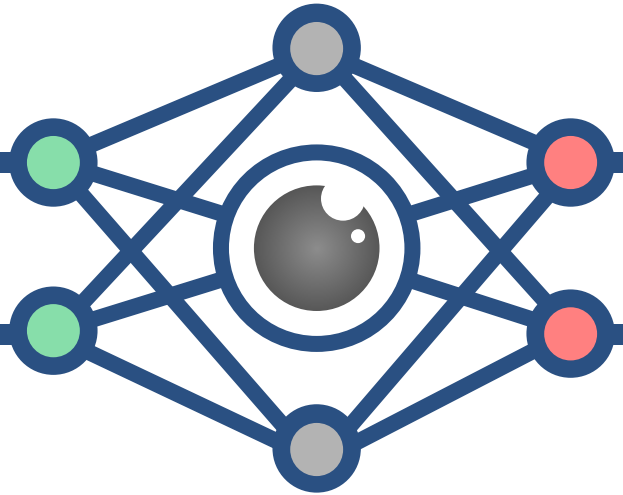
CS3485

Deep Learning for Computer Vision



Lec 16: Applications of Detection and Segmentation

Announcements

- Next Lecture: unfortunately recorded…
- Don’t forget Quiz #7

(Tentative) Lecture Roadmap

Basics of Deep Learning

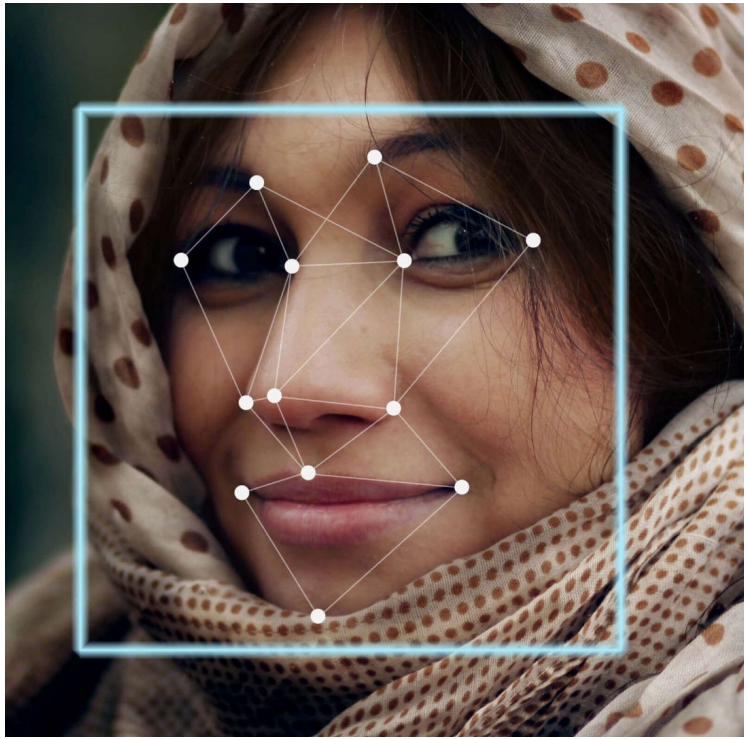
- Intro to Computer Vision
- Linear Classifiers and Perceptron
- Multilayer Perceptron
- Optimization in Deep Learning
- Pytorch I – MLPs
- Pytorch II – Images and Regularization
- Convolutional Neural Networks
- Data Augmentation and Deep CNNs
- Transfer Learning and Residual Nets
- Inception Net and what CNNs learn
- Adversarial Examples and Self-supervision

Deep Learning and Computer Vision in Practice

- Intro to MLOps
- Intro to Object Detection
- Fast Object Detection
- Intro to Image Segmentation
- Applications of Detection and Segmentation
- Autoencoders
- Image Generation with GANs
- Advanced GANs
- The Attention Mechanism
- Transformers and ChatGPT
- Image Generation by Prompt
- Misc. Topics

What we can do for now

- In the previous lectures, we saw two of the most important tasks in Computer Vision: **Object Detection** and **Image Segmentation**.
- Today we'll see how these two tasks can be generalized and applied to other important tasks involving images and videos.
- Specifically, we'll (briefly) cover the tasks of:
  - Keypoint Detection and Pose estimation,
  - Face Recognition,
  - 3D Object Detection,
  - Gaze Detection.
- Keep in mind that there are many more CV tasks won't see here due to the lack of time.



Pose estimation

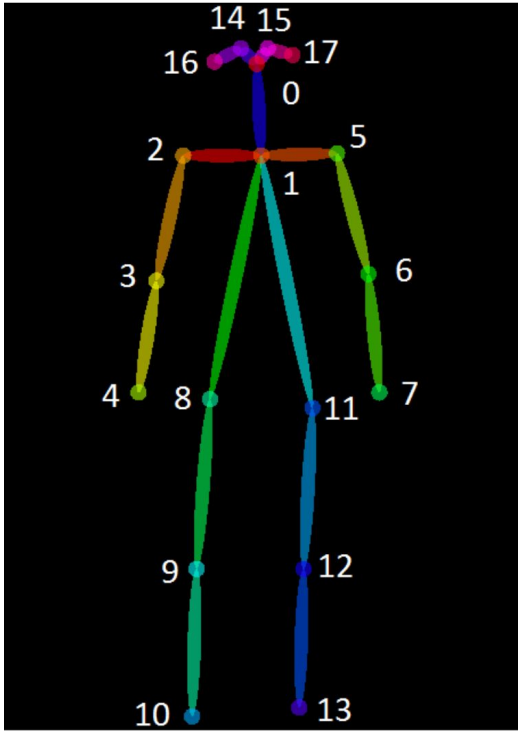
- The first task is **Pose Estimation** (also called in some contexts **Keypoint Detection**):

  Pose Estimation is the task where the goal is to detect the position and orientation of a person or an object in an image or a video.

- In the context of *Human* Pose Estimation (which is usually the most common type of pose), it simply means estimate how a person is physically positioned, such as standing, sitting, or lying down.
- Usually, this is done by predicting the location of specific **keypoints** like hands, head, elbows, etc.

Keypoint Skeleton

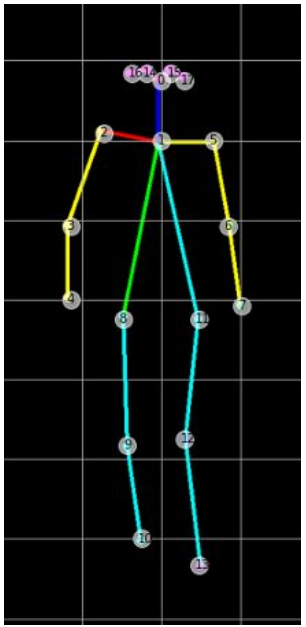
- Human pose (and some other types of pose) estimation can be reframed as finding a **Keypoint Skeleton** that represents the pose we're looking for.
- Each skeleton is formed by:
  - **Parts** or **joints**, which are the keypoints themselves (the dots on the right image).
  - **Pairs** or **limbs** (the lines on the right image), which are the connections between two parts.
- Joints are numbered and have a semantic meaning. For example, on the right, *#14* and *#15* are the left and right eyes, while *#9* and *#10* are the left knee and foot.
- Limbs should also have semantic meaning, therefore not all parts should be connected to each other in the skeleton.



Keypoint Skeleton

Original Image

Estimated Pose

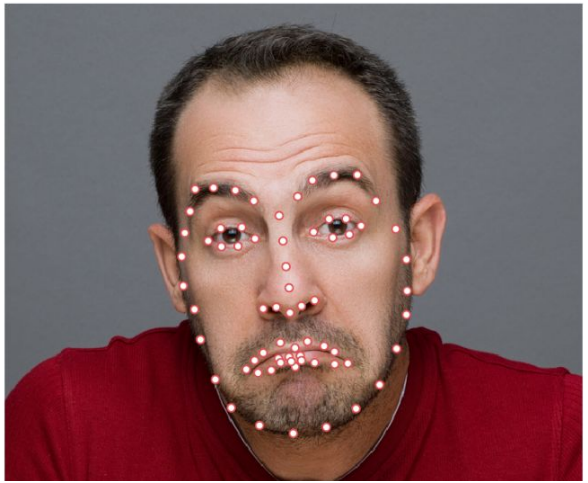


- If we have a set of parts we'd like to detect in an image of a person, we can then reframe the **pose estimation problem as a detection problem**.
- In that case we can also use the knowledge that **certain parts should be found next to others** to form limbs to help detection.
- Before doing pose estimation, one should notice that:
  - The set of parts to be detected **is subjective** and may depend on an application (sometimes one only needs a keepoint for "head" instead of "eyes" and "ears").
  - Limbs connections between parts is also subjective sometimes and **different datasets may have differents sets of limbs.**

Facial Keypoint Detection

- In pose estimation, one can also consider the problem of detecting facial keypoints on images of human faces, i.e., eyebrows, nose mouth, the face contour, etc.
- Since some facial poses are very suttle, usually many keypoints need to be estimated.





Applications of Pose Estimation

- Pose estimation and keypoint detection has many applications in the industry:

Activity Recognition

Exercise Helper

Applications of Pose Estimation

- Pose estimation and keypoint detection has many applications in the industry:

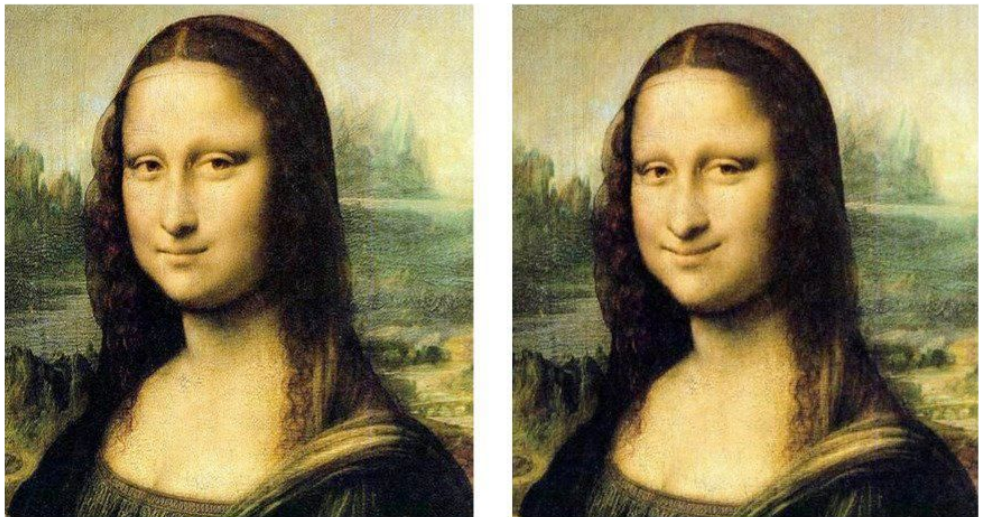
Automatic Motion Tracking for Animated Movies

Face Filters

Applications of Pose Estimation

- Pose estimation and keypoint detection has many applications in the industry:

Change People's Facial Expressions



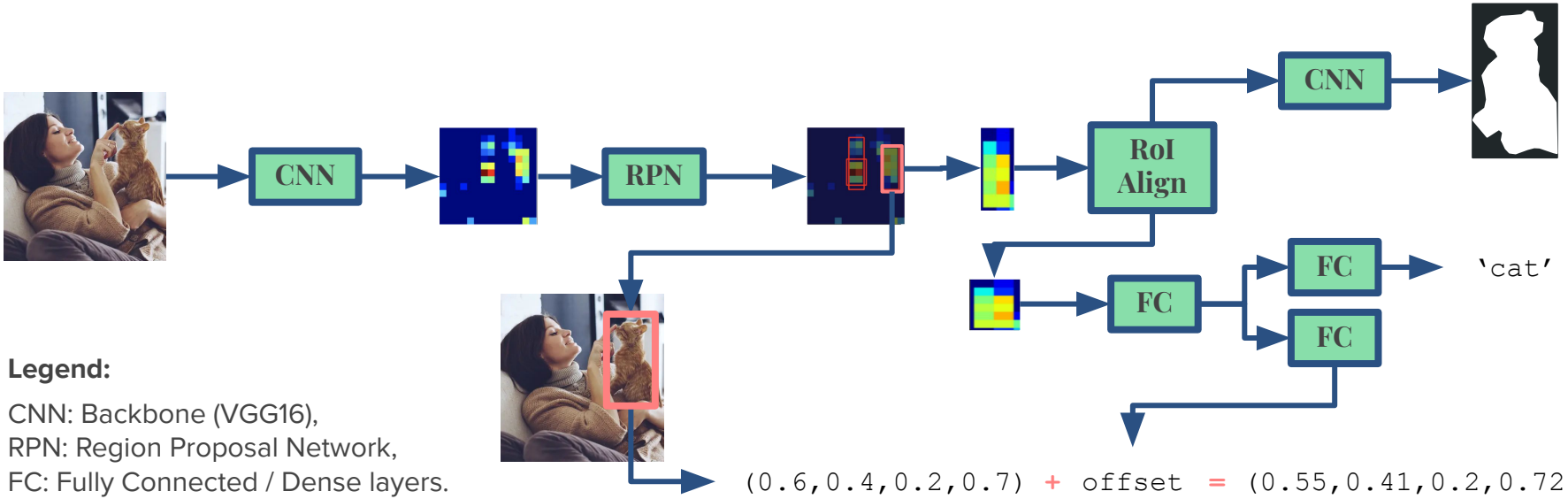
Face Morphing

Pose Estimation, Detection and Segmentation

- If a scene involves many objects, pose estimation **becomes much harder.**
- In this scenario, one has to make sure to **detect and match** only one set of key points to each object.
- One possible way to make this process easier is to do **instance segmentation first** and then search for key points in individual objects.

Revisiting Mask R-CNN

- For that reason, the same creators of Mask R-CNN (shown below), published in the same paper, a method called Keypoint R-CNN for pose estimation



Legend:

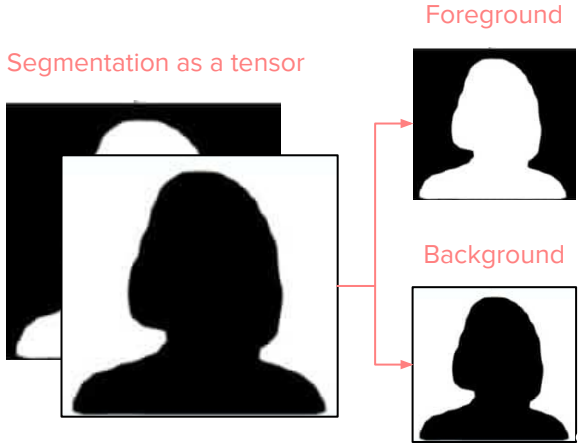
CNN: Backbone (VGG16),
RPN: Region Proposal Network,
FC: Fully Connected / Dense layers.

Keypoint R-CNN

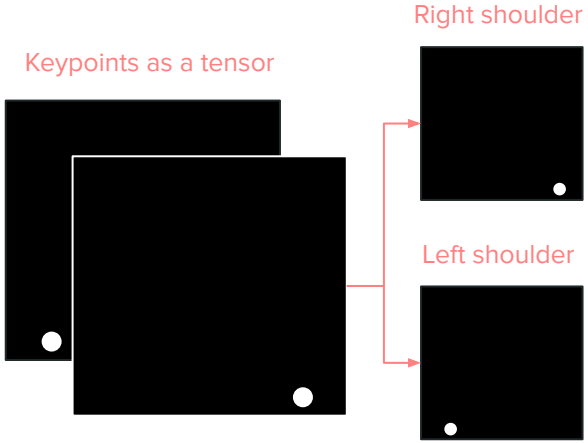
- Mask and Keypoint R-CNN just differ in the way the keypoints are encoded in the mask.
- For Keypoint R-CNN each channel of the output represents a keypoint to be detected in the original image (there will be as many channels as the number of keypoints).

Input Image

Ground Truth for Mask R-CNN



Ground Truth for Keypoint R-CNN



Keypoint R-CNN in Pytorch

- This kind of data is also available in the MS COCO database and a model for Keypoint R-NN trained on that data is available in PyTorch, just like with Mask and Fast R-CNN:

```
from torchvision.models.detection import keypointrcnn_resnet50_fpn
from torchvision.models.detection import KeypointRCNN_ResNet50_FPN_Weights

model_keypoint_rcnn = keypointrcnn_resnet50_fpn(weights=KeypointRCNN_ResNet50_FPN_Weights.COCO_V1)
```

- Again, for a properly processed image called `img`, can do inference* via:

```
results = model_keypoint_rcnn(img)
```

where `results` is a list of dictionaries, one for each object in `img` and each with keys corresponding to the object box, object label, label score, object keypoints and keypoint scores (which measure how certain the network if of those keypoints).

* For more details details on the necessary preprocessing and the output format of Keypoint R-CNN, check out this link.

Dense pose estimation

- **Dense human pose estimation** (conceived in 2018) is a type of pose estimation that aims at mapping all human pixels of an RGB image to the 3D surface of the human body.
- A dataset for it, called DensePose-COCO, was created in that same year and it consists of images and annotations of how each body point is positioned in a 3D mesh.
- As a result, the **DensePose R-CNN** was also created to solve that task.

Input Image	Annotated Image	Dense Pose estimation
		

Detectron

- In 2018, Meta (the creator of PyTorch and the R-CNN family), launched a library called Detectron, followed by Detectron2 in 2020.
- Detectron2 includes **fast** pretrained models for the following Computer Vision tasks:
  - Object Detection (Fast and Faster R-CNN),
  - Instance Segmentation (Mask R-CNN),
  - Keypoint Detection and Pose estimation (Keypoint R-CNN),
  - DensePose Estimation (DensePose R-CNN),
  - Panoptic Segmentation.
- The library is easy to use* and also provides tools for visualization and evaluation.

Result Example of Detectron2

* Check out its documentation for more details.

Exercise (*in pairs*)

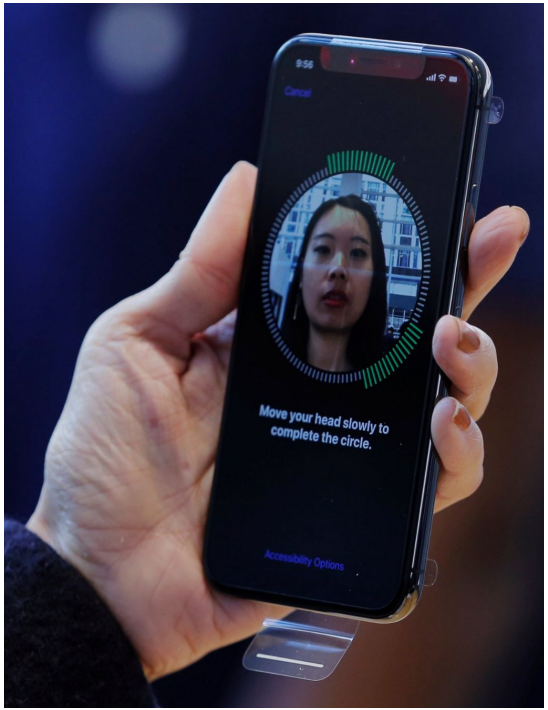
- There are many applications and tasks we can think of that involve Image Segmentation and Detection. Can you think of some?

Face Recognition

- Another very important task related to Object Detection is Face Recognition:

  Face recognition is the task of identifying or verifying the identity of an individual using an image of a video picturing their face.

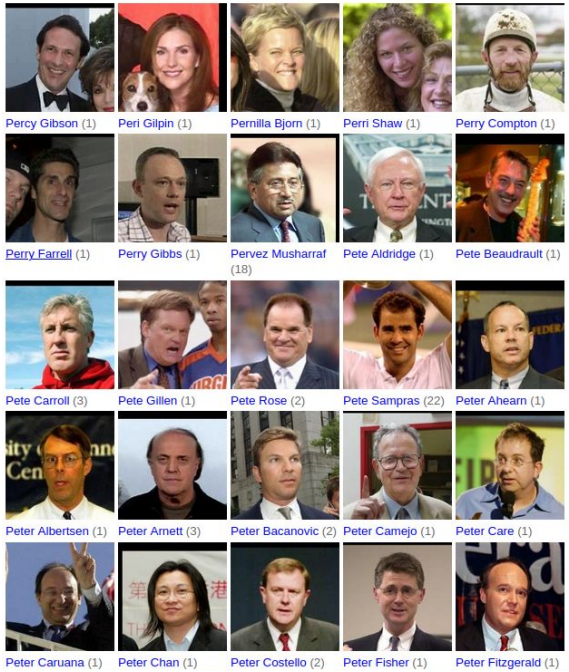
- It is historically one of the most important Computer Vision tasks with great commercial interest, especially in domains of **biometry** and **surveillance**.
- This tasks is usually accompanied by the task of Face Detection (or Localization) which simply aims at finding all face instances in an image.



Face Recognition

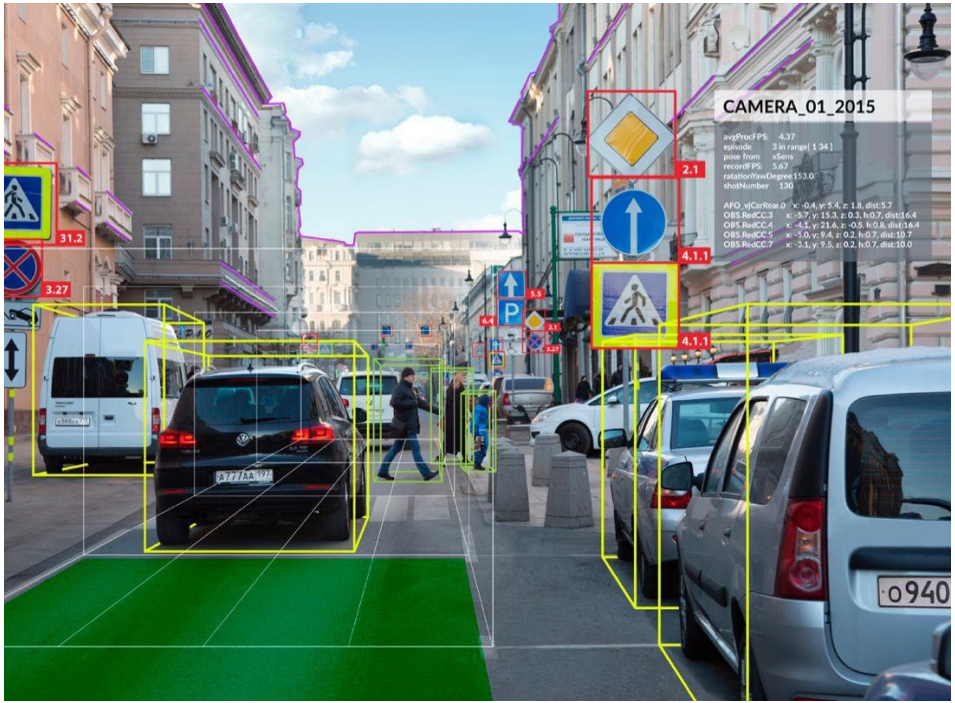
- There are two main modes for face recognition, as:
  - **Face Verification:** A one-to-one mapping of a given face against a known identity (e.g. *is this the person?*).
  - **Face Identification:** A one-to-many mapping for a given face against a database of known faces (e.g. *who is this person?*).
- To train a network for Face Recognition, one can use the Labeled Faces in the Wild (LFW) dataset, composed of 132k images of 5749 people.
- The first major Deep Learning method to attempt this problem was called DeepFace, published in 2014 also by Facebook research, which achieved a 97.35% accuracy in the LFW dataset.

Sample faces from LFW (Letter "P")



3D Object Detection

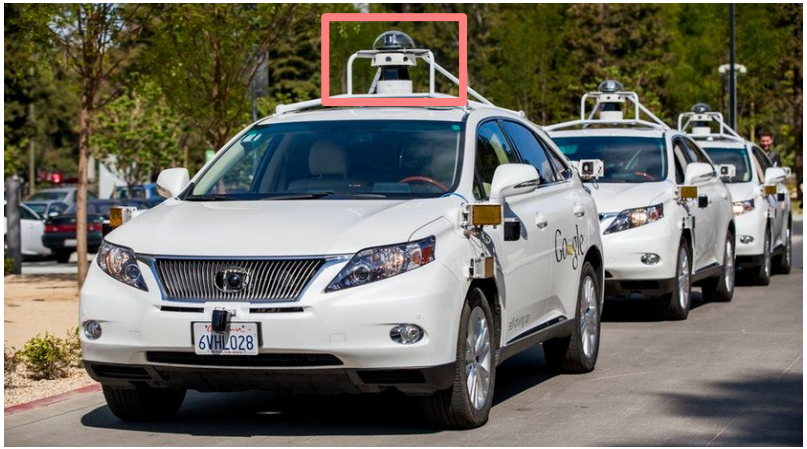
- A simple generalization of Object detection is **3D object detection**, where one looks for 3D objects in a scene or video, which has a clear application to **Autonomous Vehicles**.
- To solve this task, images (which are 2D) are not enough, as one needs knowledge of **Visual Depth** (or distance from the objects in a scene to the observer).
- For that, a typical solution involves using **LIDAR data**.



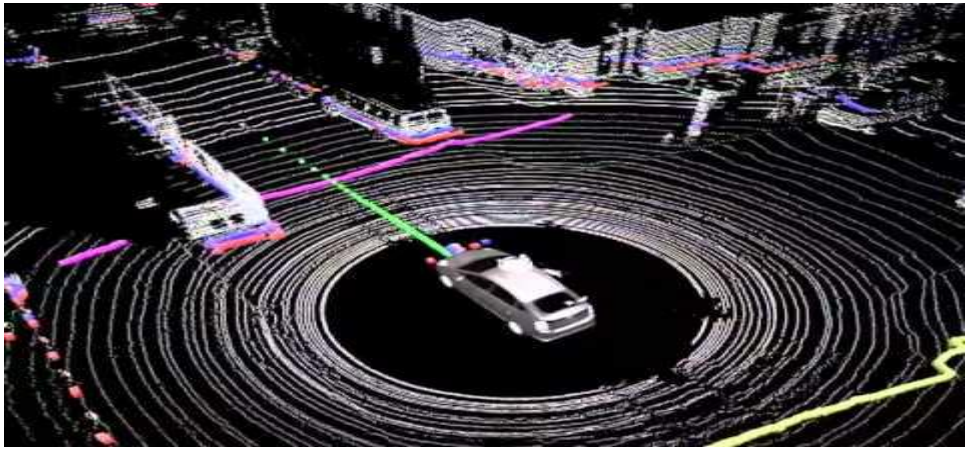
3D Object Detection

- LIDAR (an acronym of “light detection and ranging”) is a light sensing technology that creates a 3-D map of a car’s surroundings using a laser and receiver.
- If annotated, that data is used in 3D versions of YOLO to solve the detection problem.

LIDAR camera placed above self driving cars



Point Cloud detected by the LIDAR Camera



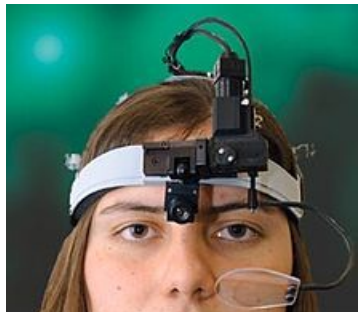
Gaze Detection

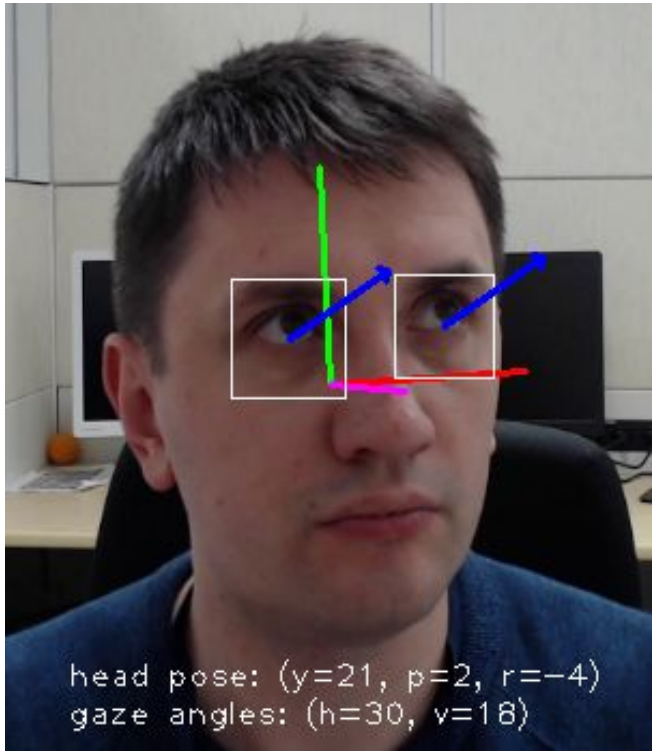
- Another interesting task in (Eye) Gaze Detection:

  Eye gaze detection aims at tracking the direction a human eye is gazing at in an image.

- While this could be solved using an expensive **Eye Tracking device**, the goal here is to not need it.
- For Deep Learning-based Gaze Detection, we use images of faces annotated using the data coming from a tracking device to train a network that also detects where the human eyes are.

Eye Tracking Device

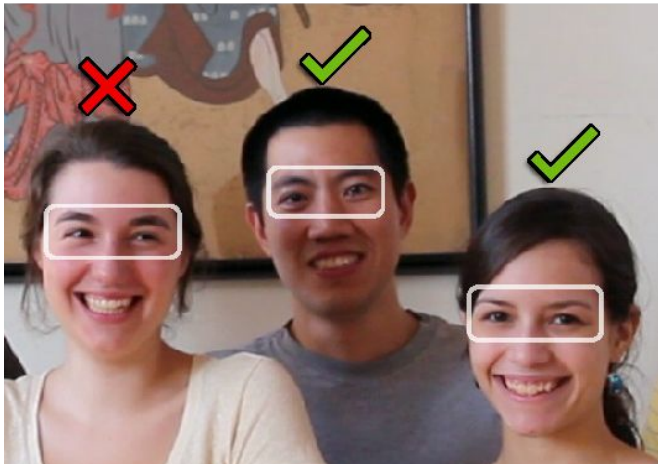




Gaze Detection

- Although it is a recent computer vision task, it has already found many applications in the industry:

Smart Photo Locking Mode



Fatigue Detection

Future of Virtual Reality

Other Tasks

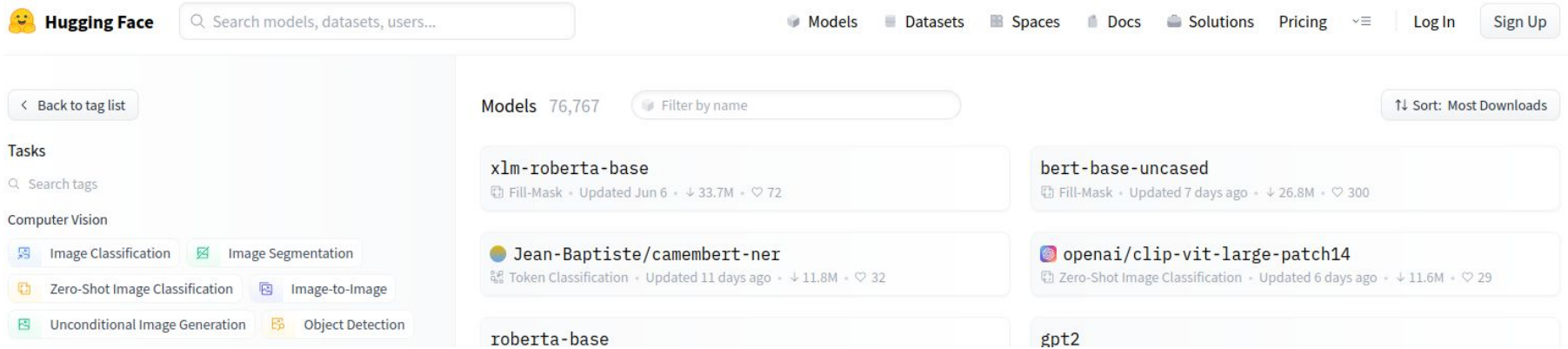
- Some other interesting tasks related to Object Detection and Segmentation are:
  - **Object Counting**: we first detect the desired object and then count (or estimate the count) how many of then are in there. This is very useful in **crowd control, for example.**
  - **Optical Character Recognition**: in order to create an automatic way to read texts from images, we first need to detect it. After that, we can segment each word and classify that word according to its meaning
- There numerous more tasks, however! In fact, every once in a while a new one is created!

Crowd Counting

IOS Live Text

Hugging Face

- Hugging Face is a very good resource to find pre-trained models written in PyTorch for many Computer Vision (like Detectron2) and NLP tasks.
- There you'll find dozens of CV tasks / DL models we won't cover in this course.



- In it you find also a **hosted api** where you can try out the inference of these models directly from your browser! It is pretty cool.

Video: The Road to Autonomous Driving