Computer Vision

7/15/2025
Jing Jia

MAYFLOWER
A.S.T.E.M.
ACADEMY

1

Computer Vision

A multidisciplinary field:

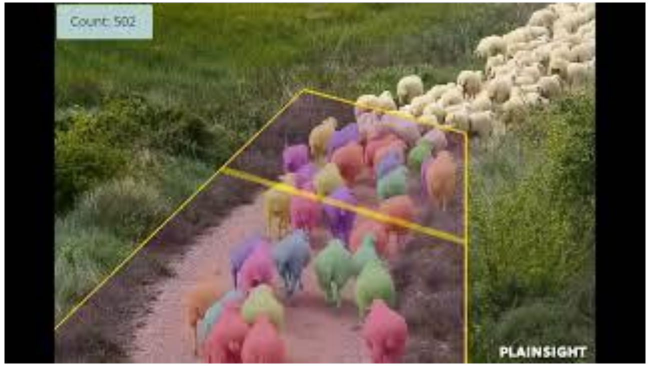
- Physics - light is an EM wave.
- Optics - lenses, reflectance, etc.
- Applied Mathematics (linear algebra, numerical methods, computational
- geometry)
- Digital Signal Processing
- Stochastics/Probability (dealing with uncertainty in estimation)
- Computer Science (algorithm development)

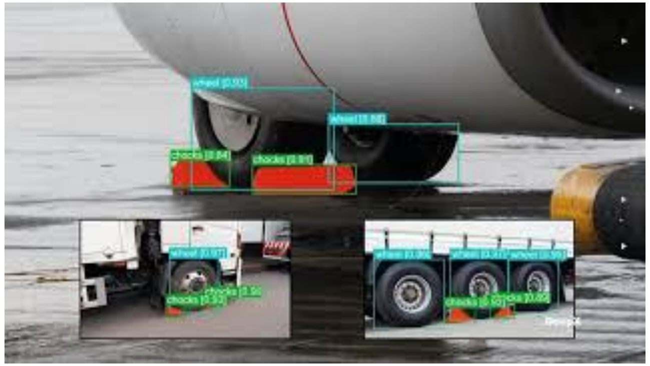
- Computer Vision is a branch of AI
- Computer Vision=Robot Vision
- CV has overlap with Computer Graphics

What is Robot Vision, Computer Vision?

The goal of computer vision is to computationally represent visual imagery in order to enable a computer system to interpret and interact with the surrounding environment. Interpretation of imagery includes image segmentation, object/texture recognition, depth/distance estimation and motion estimation

Computer Vision in Real world





Applications of Computer Vision

- Recognition: Object, Face, Material
- Computer-assisted surgery (e.g. Frameless brain surgery)
- Autonomous navigation for Robotics
- Driverless Cars
- Content-based video indexing (e.g. find frames with a key person from thousands of hours of video)
- Hand Gesture Recognition (e.g. mouseless computers)
- Computer Graphics Rendering/ Movie Effects
- Computer Assisted Radiology/ Diagnosis (finding Tumors in X-rays)
- etc

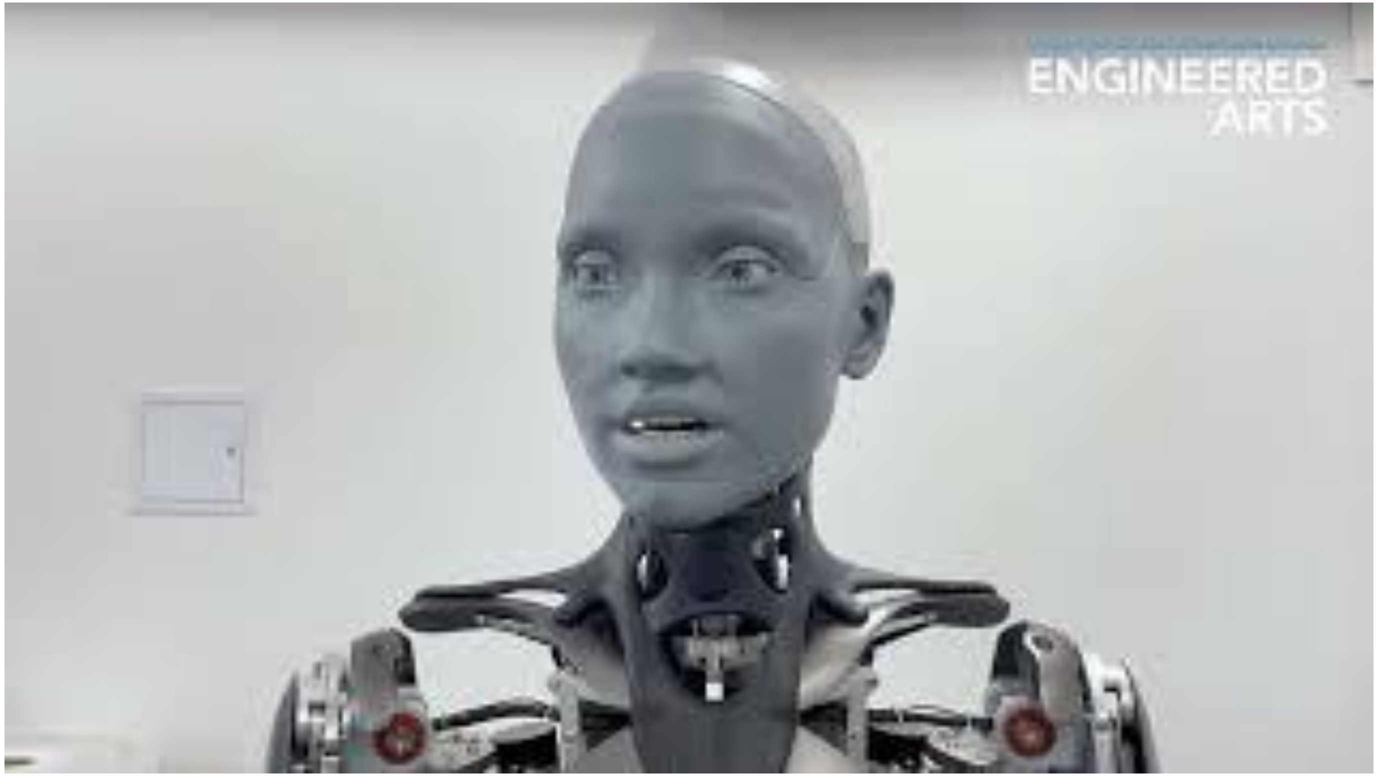
CVPR 2013 – Expressive Visual Text-to-Speech

https://www.youtube.com/watch?v=6QFIWP0ZfPY

Ameca, humanoid robot from Engineered Arts

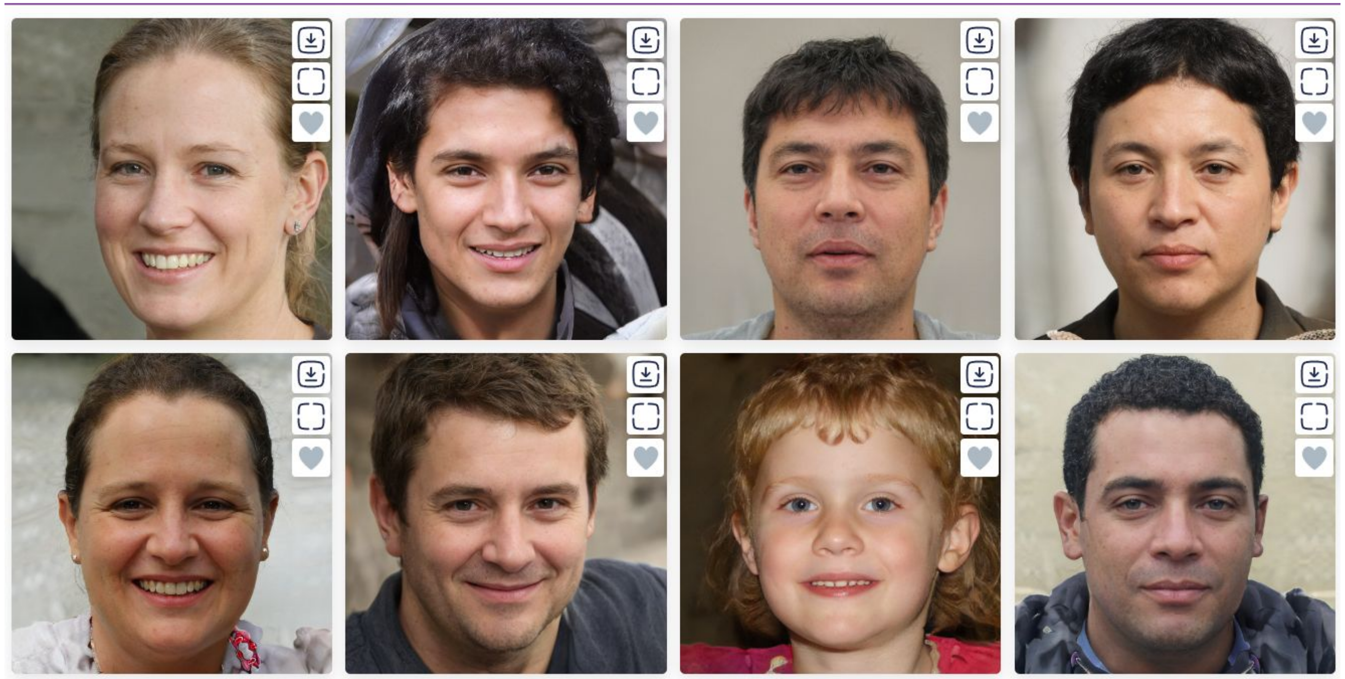
https://www.youtube.com/watch?v=IPukuYb9xWw

Amera and GPT3

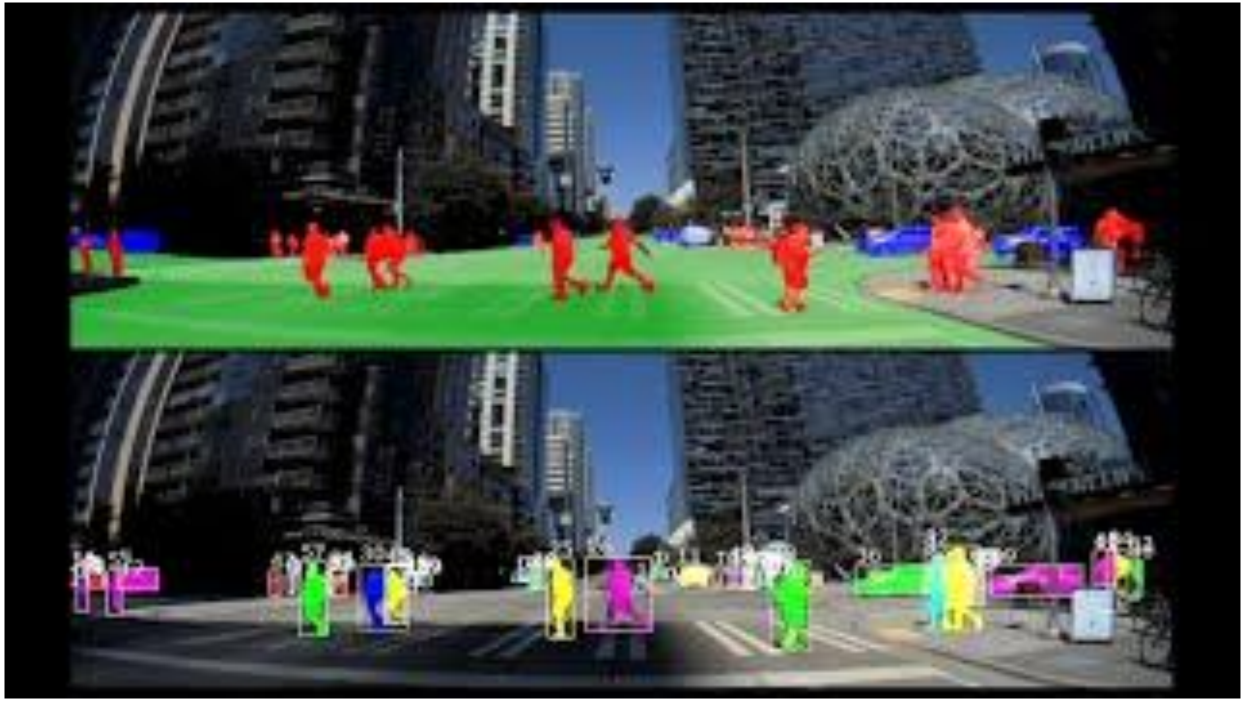


https://www.youtube.com/watch?v=fQ2JpkFBAOs

https://thispersondoesnotexist.com/



Autonomous Driving



https://www.youtube.com/watch?v=HS1wV9NMLr8

Computer Vision

What am I looking at? →	Image Classification/Recognition
What are all the things I see, name every image pixel? →	Semantic Segmentation
Where is the object in the 3D world? →	3D Reconstruction
Where is it and where am I? →	SLAM
What should I do? →	Reinforcement Learning
How do I learn by watching you? →	Imitation Learning

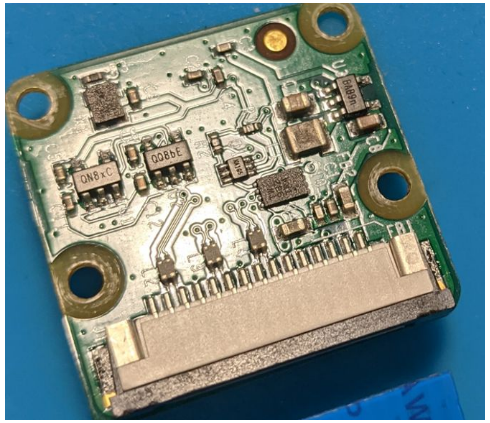
2

Image Formation

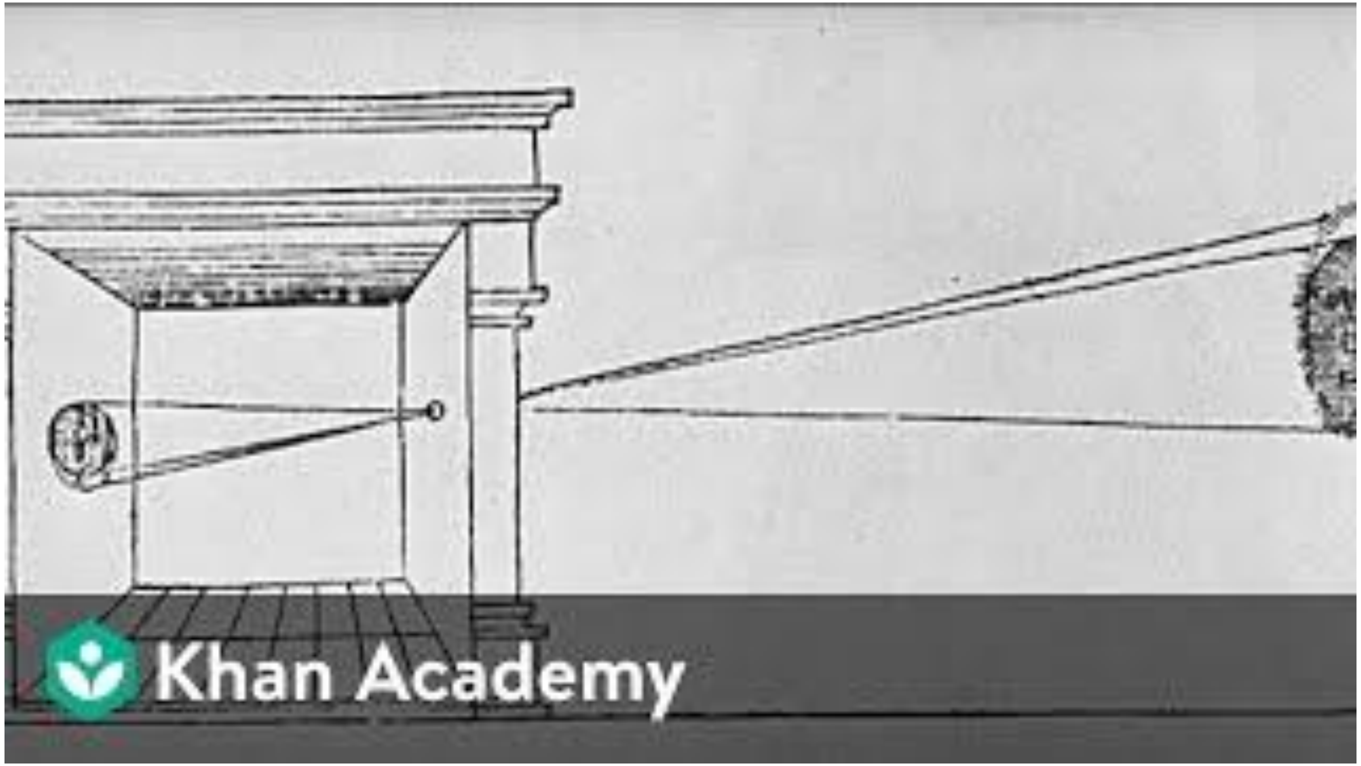
Image Fundamentals

Definition of a Camera:

- A device that converts an optical signal into an electrical signal.

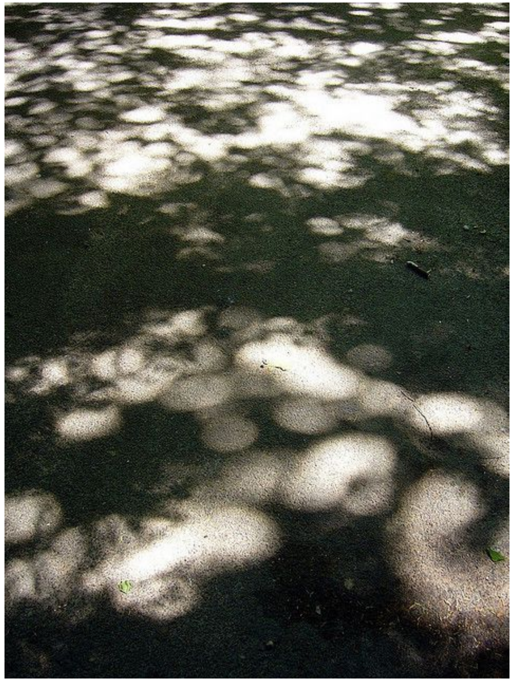


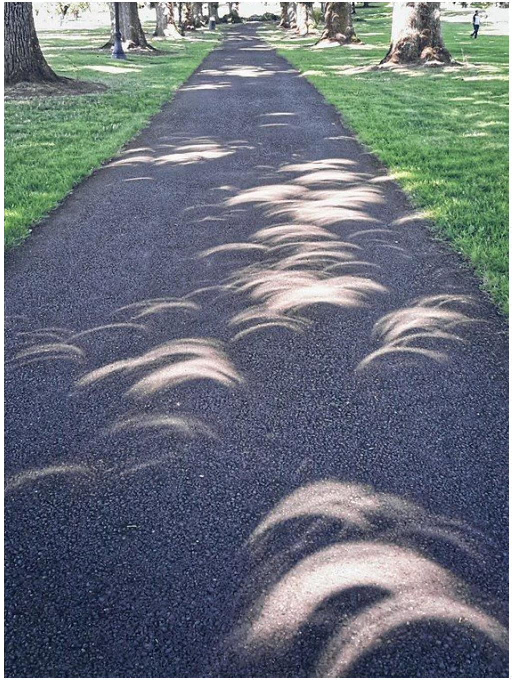
Pinhole Camera



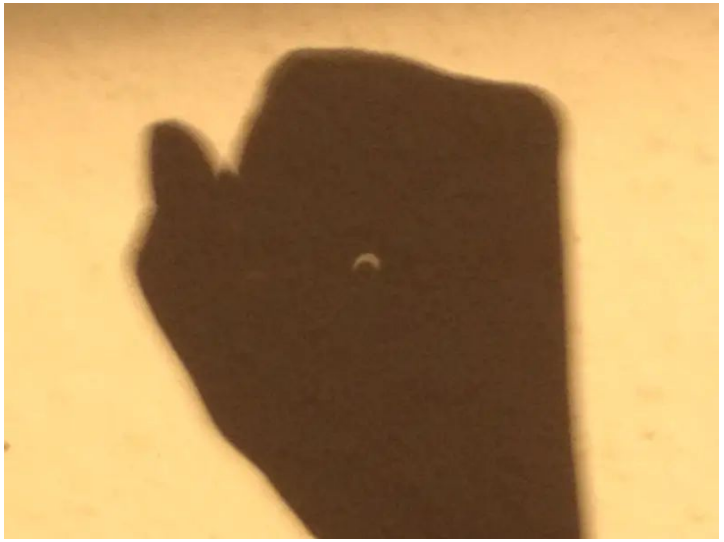
https://www.youtube.com/watch?v=jhBC39xZVnw

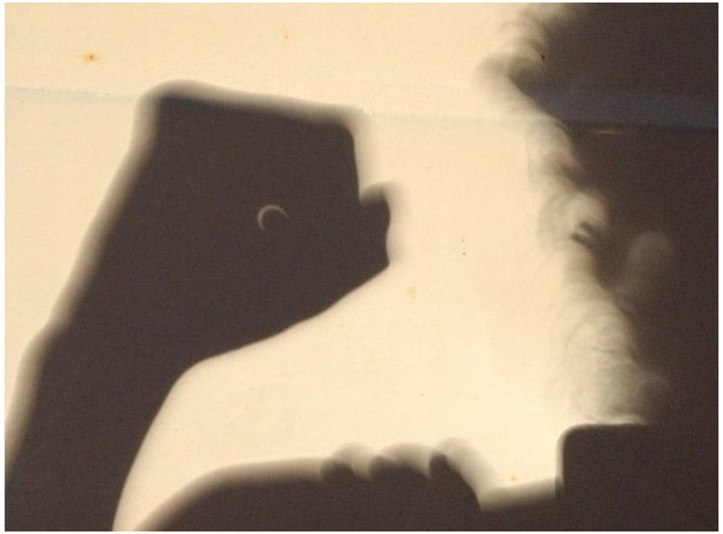
Pinhole Camera in Nature: Dappled Sunlight





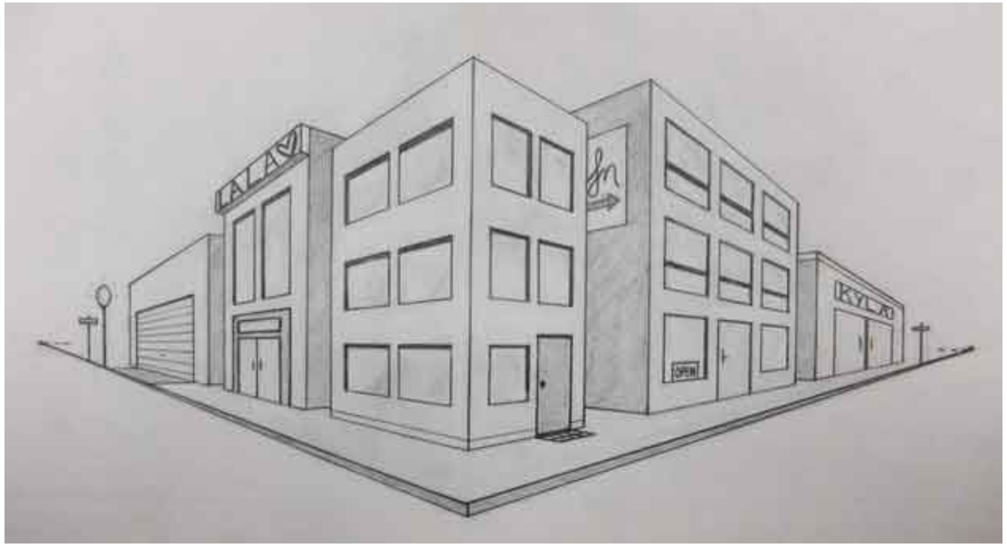
Fist Pinhole Camera





Perspective Projection

Notice that objects appear smaller as they move away from you



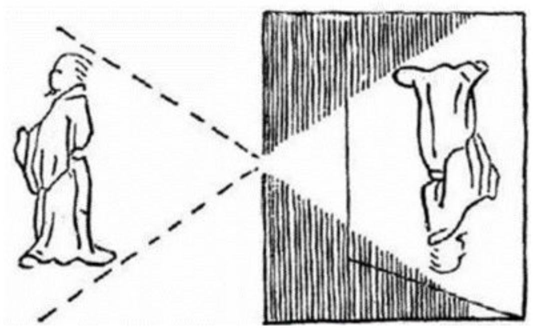
Principle of Image Acquisition

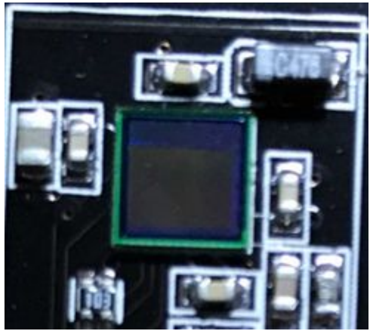
Pinhole Imaging Model:

- Simplest camera model, ignores field curvature and distortions.
- Real cameras use calibration to correct these distortions.
- Widely used due to simplicity and effectiveness.

Image Capture Process:

- Light passes through the camera lens onto a sensor chip.
- The sensor chip converts light information into a digital signal.
- The light-sensitive element on the sensor chip is crucial for this process.





Sensor chip

Principle of Image Acquisition

https://www.youtube.com/watch?v=i85jZ76azZ8

Pixels and Resolution

Pixels:

- Tiny square grids on a light sensor.
- The smaller the grid, the higher the pixel count.
- Higher pixel count results in higher image resolution.

Frame Rate (FPS):

- Frequency at which images are displayed, measured in frames per second.
- Human eye can barely detect lag above 20 fps.
- Higher frame rates are better for machine vision.

An image with a resolution of 31x22 pixels.

Color

Color Perception:

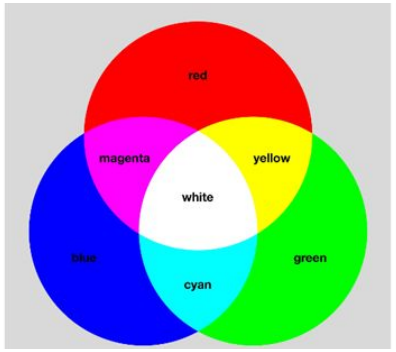
- Visual effect produced by the eye, brain, and individual experiences.
- Different frequencies of electromagnetic waves are seen as different colors.

RGB Color Model:

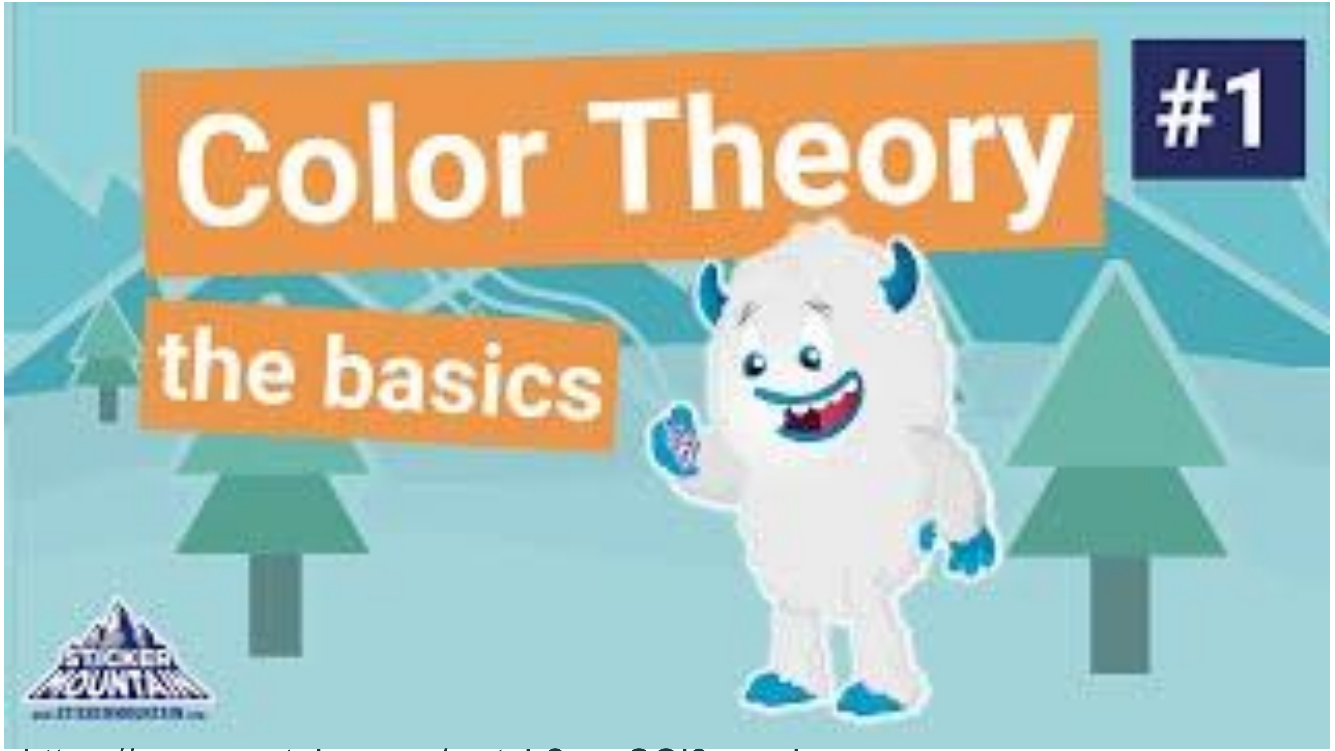
- Based on human eye sensitivity to yellow-green, green, and bluish violet light.
- Used in monitors for color display.

LAB Color Space:

- "L" represents luminance.
- "A" axis: red to green.
- "B" axis: yellow to blue.
- Designed to be close to human vision.

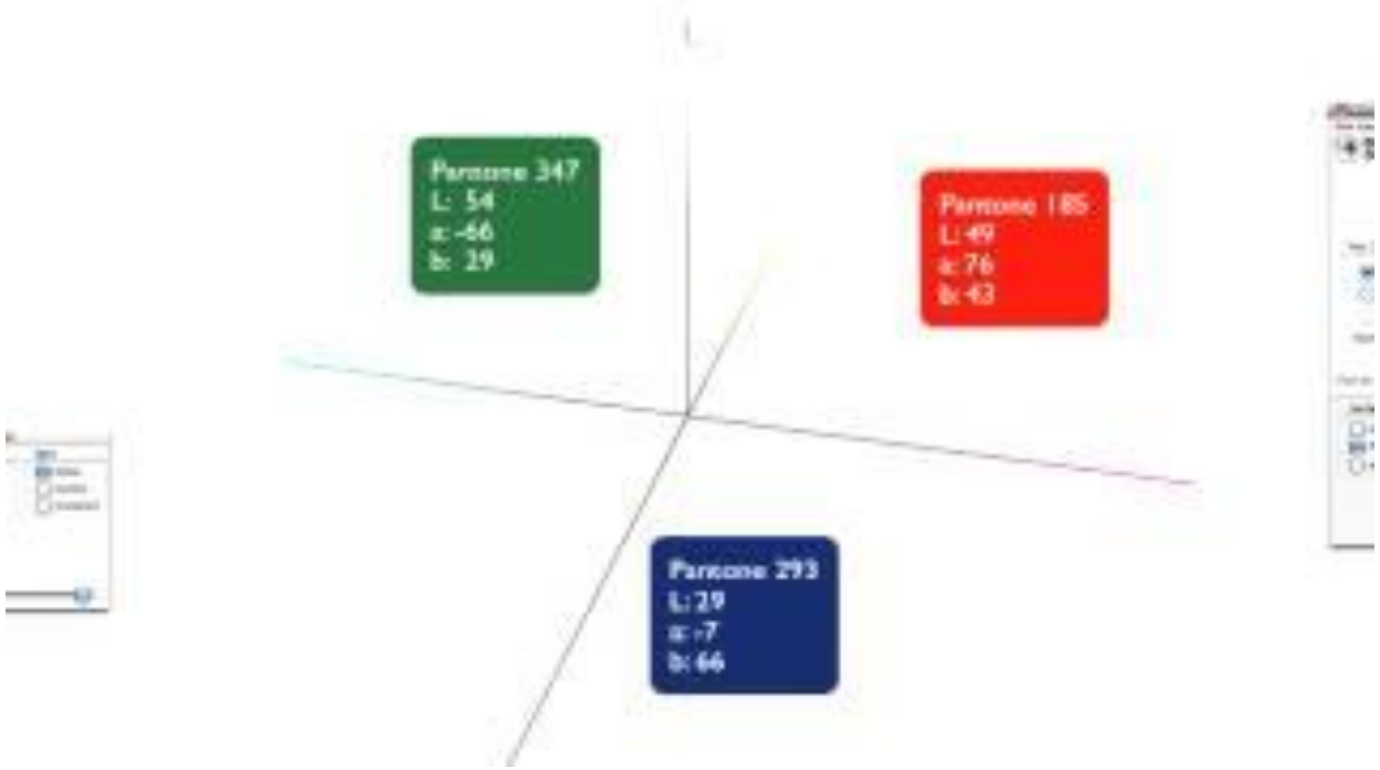


Color Theory



https://www.youtube.com/watch?v=oSQj3meaJcg

LAB Color Space



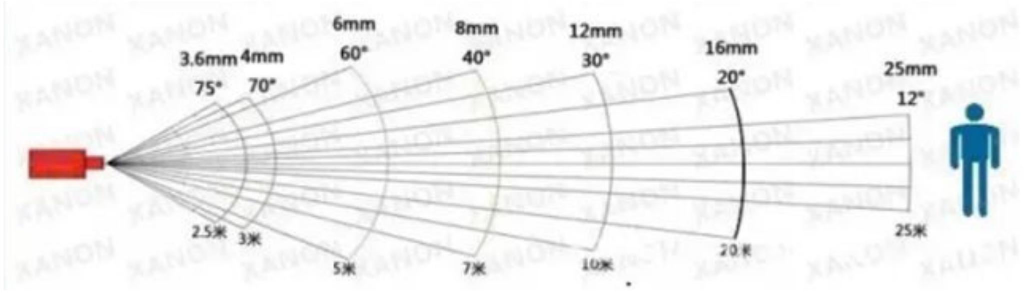
The Lens

Lens Functionality:

- Refracts light to the sensor, determining image size and distance.

Focal Length:

- Distance from lens's main point to focal point.
- Determines image size, field of view, depth of field, and perspective.
- Longer focal length: larger image, smaller angle of view.
- Shorter focal length: smaller image, larger angle of view.



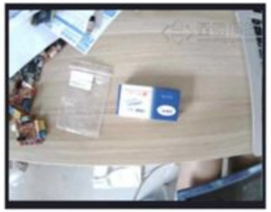
The Lens

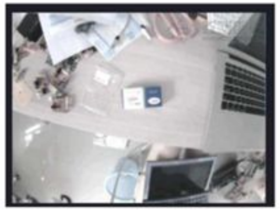
Lens Aberration:

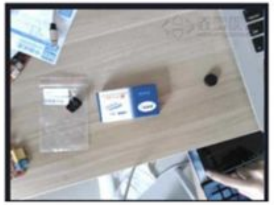
- Distances to the lens vary across the sensor, causing distortion.
- Fisheye effect (barrel distortion) at edges.
- Corrected using algorithms or distortion-free lenses.

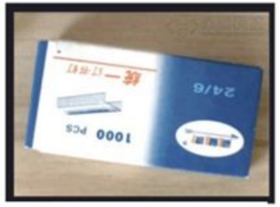
Lens Comparison Example:

- Different focal lengths impact the visual range and image clarity.
- OpenMV example shows differences when 20cm away from the table.

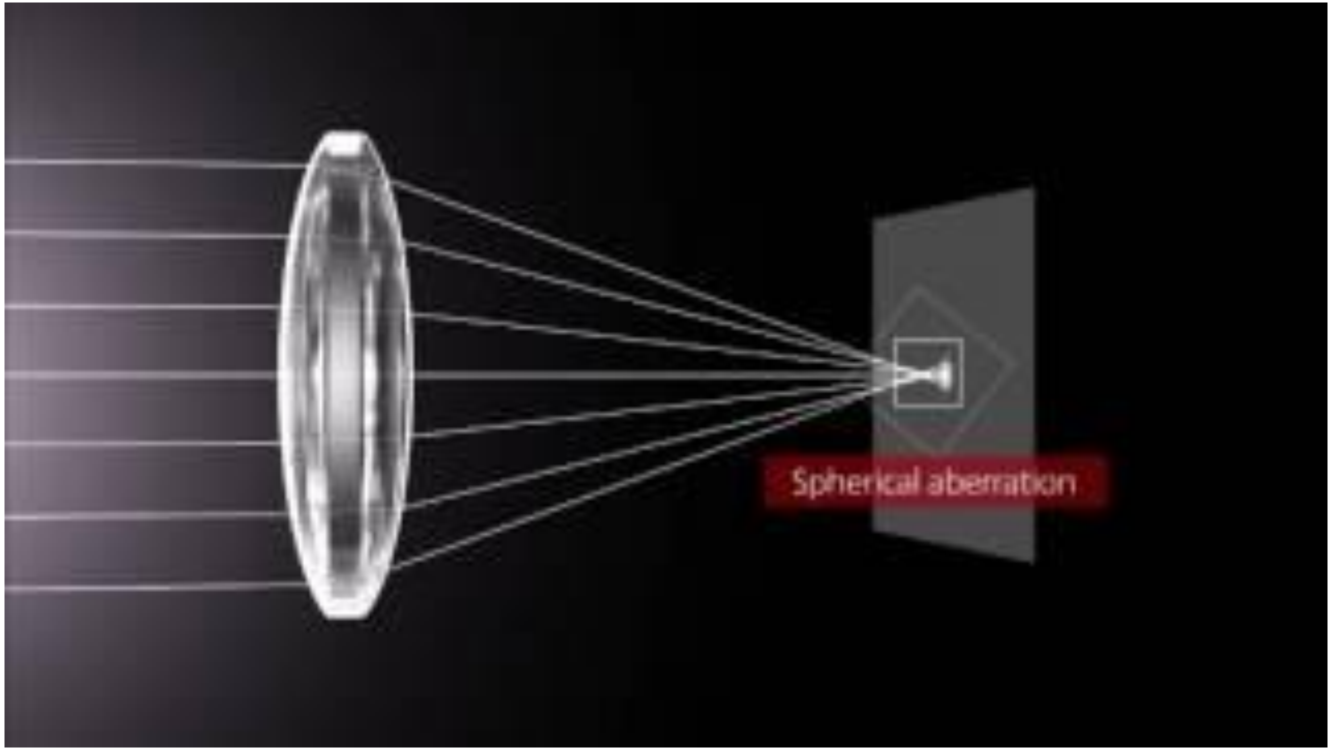

Standard Lens


Wide-Angle Lens


Distortion-Free Lens


Telephoto Lens

Lens



https://www.youtube.com/watch?v=EL9J3Km6wxI

3

Matrix Representation

Vectors, Matrices

Let $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}.$

$$\vec{x}^T \vec{y} = \vec{x} \cdot \vec{y} = x_1 y_1 + x_2 y_2 + x_3 y_3 \text{ (A scalar!)} \tag{1}$$

$$\vec{x}^T = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \tag{2}$$

Matrix $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$

Matrix Multiplication

$$AB = C$$

$$c_{ij} = \sum_t a_{it} b_{tj}$$

c_{ij} is the inner product of the ith row of A and the jth column of B.

Note: Number of cols of A must equal Number of rows of B

Matrix multiplication properties:

- Associative (AB)C = A(BC)
- Distributive A(B + C) = AB + AC
- NOT Commutative AB ≠ BA in general

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$AI = A$$

Transpose

Transpose of A is denoted A^T . The ith row of A is the ith column of A^T .

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

A symmetric matrix is a matrix which equals its transpose

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 8 \end{bmatrix}$$

Linear Equations

Expressing a system of linear equations in matrix form

$$\begin{aligned} 2x_1 + 3x_2 &= 4 \\ 3x_1 + 4x_2 &= 7 \end{aligned} \quad \rightarrow Ax = b$$

$$A = \begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$b = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

Linear Equations

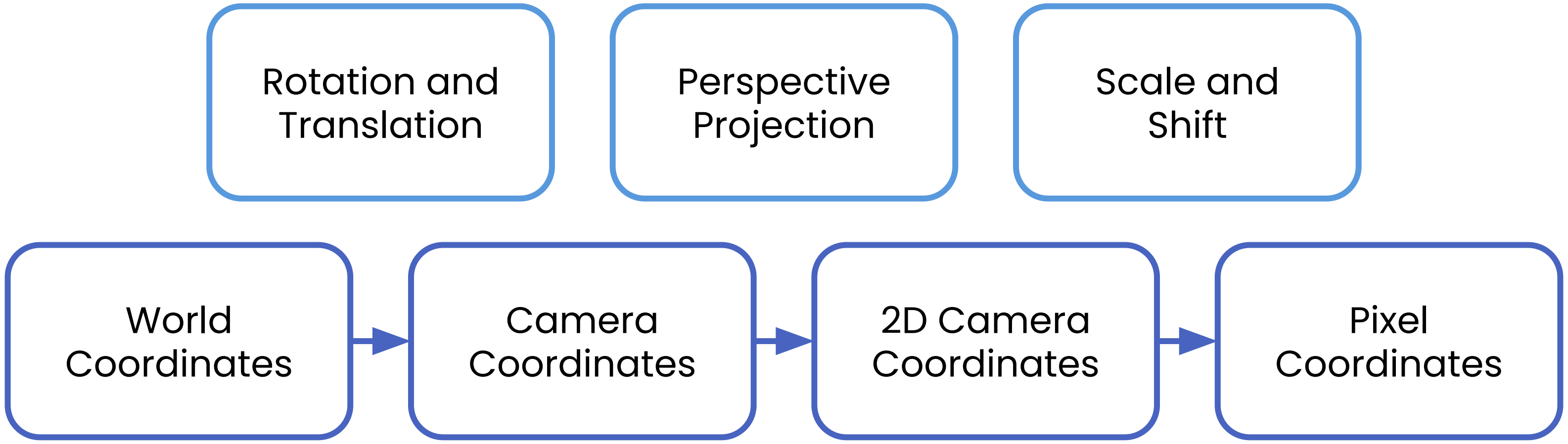
This idea of expressing equations with matrices is fundamentally important.

Ax can be viewed as a weighted sum of the columns of A.

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} x_1 + \begin{bmatrix} 3 \\ 4 \end{bmatrix} x_2 = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

The system Ax = b is solvable if and only if the vector b can be expressed as a linear combination of the columns of A.

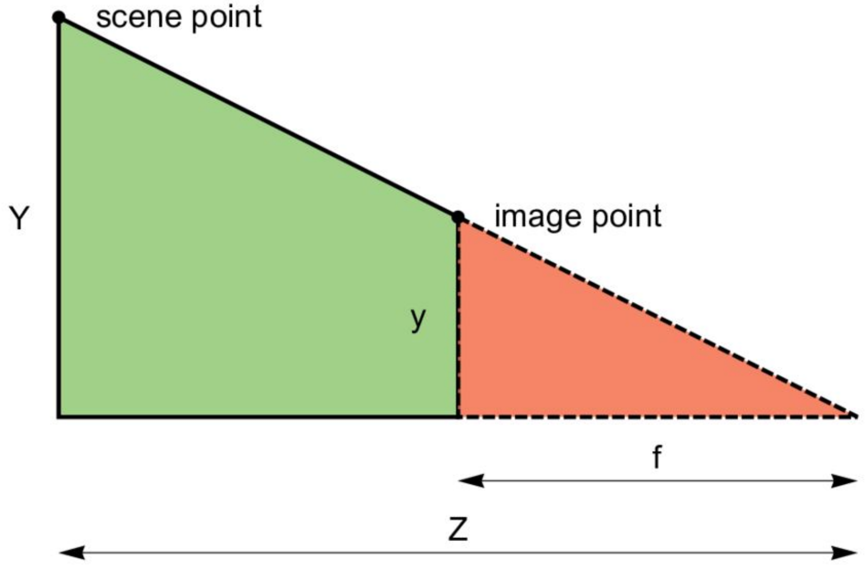
Complete Image Formation Pipeline



The entire imaging pipeline, described by 3 steps, can be described by the product of three matrices... which is a single matrix

Perspective Projection: Similar Triangles

$$x = f\frac{X}{Z}$$
$$y = f\frac{Y}{Z}$$



Perspective Projection as Matrix Multiplication

It is desirable to have all of the mathematical relations in matrix form (easy concatenation of steps in the imaging pipeline, expression as a system of linear equations) The division by Z is a problem. We can’t put the perspective projection in matrix form, these are nonlinear equations. We use homogeneous coordinates or projective coordinates to allow the perspective projection part of the imaging pipeline to be written as a matrix. When we use homogeneous coordinates . . . division is postponed, Z remains in third component.

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Homogeneous Coordinates

But these are homogeneous coordinates, so to get the final answer, divide by the third component.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{\tilde{x}}{\tilde{w}} \\ \frac{\tilde{y}}{\tilde{w}} \end{bmatrix}$$

Shift and Scale (from image to pixel coordinates)

After perspective projection there is a scale and shift that transforms the projected values into pixels.

$$\begin{bmatrix} im_x \\ im_y \end{bmatrix} = \begin{bmatrix} 1/s_x & 0 & o_x \\ 0 & 1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Why is a Shift Needed?

- Image pixels have a different x-y coordinate frame compared to camera coordinates.
- In an image, the origin (0,0) is typically at the upper left corner.
- In projected camera coordinates, the origin (0,0) is at the intersection of the principal axis with the image plane.
- This shift of the origin is accomplished using the O_x and O_y terms in the last column of the transformation matrix.

Why is a Scale Needed?

- The camera coordinate frame uses a unit of measure such as millimeters (mm).
- Pixels are unitless and require a conversion.
- Division by s_x and s_y (the horizontal and vertical size of the pixel in the measurement units, e.g., mm) is necessary to convert camera coordinates to pixel coordinates

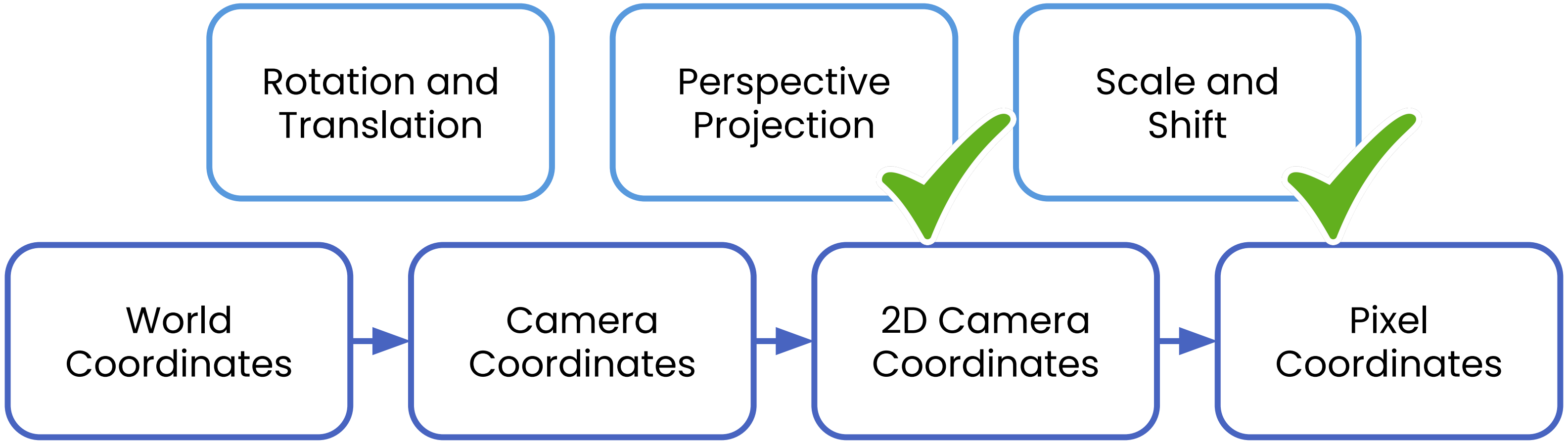
Perspective Project + Scale and Shift

Here we have denoted the pixel coordinates frame **im** by a superscript. We can put together the two matrices as follows:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} 1/s_x & 0 & o_x \\ 0 & 1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f/s_x & 0 & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Image Formation Pipeline now



Coordinate Frame Transformations

- Perspective Projection must be done in a camera coordinate frame
- Object Vertices are in a World Coordinate Frame (a reference frames).
- For example a cube has vertices
  - Bottom Face: (0,0,0); (1,0,0); (1,1,0); (0,1,0)
  - Top Face: (0,0,1); (1,0,1); (1,1,1); (0,1,1)
- These vertices are with respect to a fixed reference coordinate frame.

3D Coordinate Frame Transformations

3x4 transformation matrix, rotation followed by a translation

$$T = [\ R \mid t\ ]$$

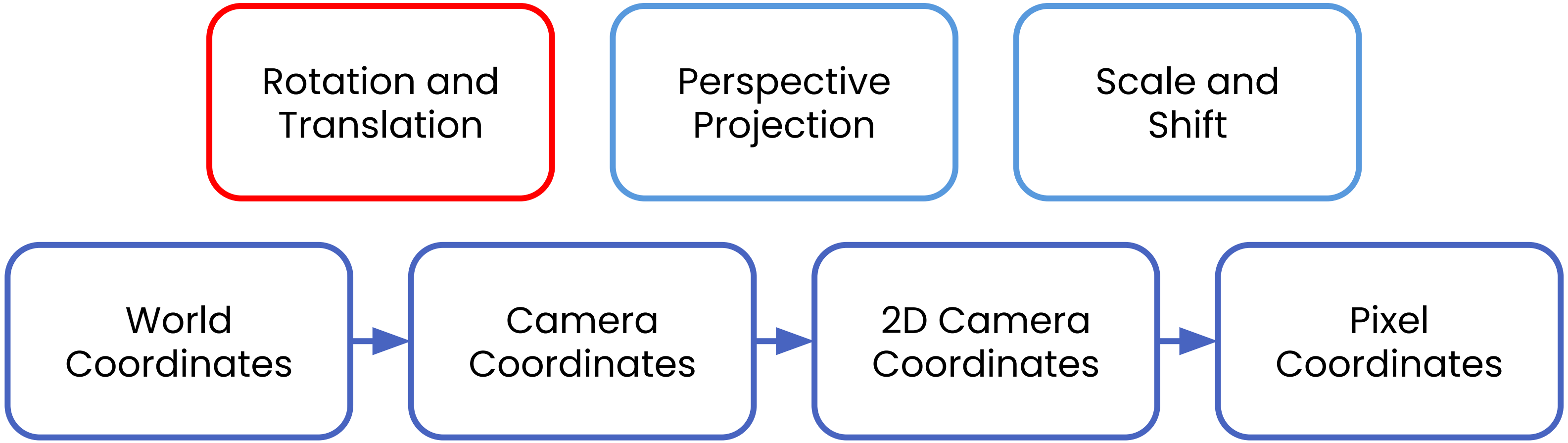


Image Formation Pipeline Complete

The complete image formation pipeline is given by,

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f/s_x & 0 & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^C R & {}^C t \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

Can be simplified to,

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = M \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

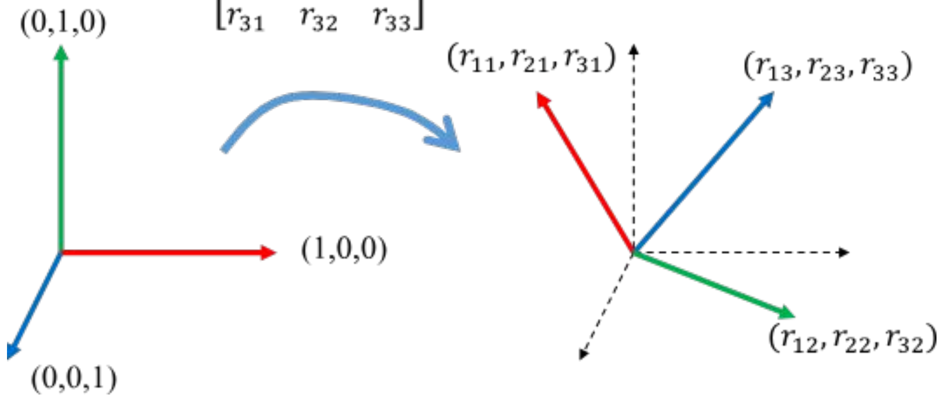
Coordinate Frame Transformation

We will consider two ways to specify the coordinate frame transformation:

1) with a series of rotations followed by translations

2) using the geometric interpretation of the columns of the rotation matrix.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$


$(0,1,0)$
$(1,0,0)$
$(0,0,1)$
(r_{11}, r_{21}, r_{31})
(r_{13}, r_{23}, r_{33})
(r_{12}, r_{22}, r_{32})

Transformation Process

Initial Alignment:

- Let A denote the reference frame and B denote the other frame.
- Coordinate frame B is initially aligned with A.

Rotation Steps:

- We rotate B about x-axis of A by angle α.
- Then rotate B about y-axis of A by angle β.
- Finally, rotate B about z-axis of A by angle θ.

Translation:

- The coordinate frame B is translated by 10 units along x-axis of A .

Coordinate Frame Transformation

Given a point P w.r.t the B coordinate frame we can convert it to that point w.r.t. the A coordinate frame as follows

$$^{A}P = \begin{bmatrix} ^{A}R_B & ^{A}t_B \end{bmatrix} (^{B}P) = ^{A}T_B(^{B}P)$$

$$^{A}R_B = Rot(\hat{z}, \theta) Rot(\hat{y}, \beta) Rot(\hat{x}, \alpha)$$

Note the order of the concatenation of the rotation matrices.

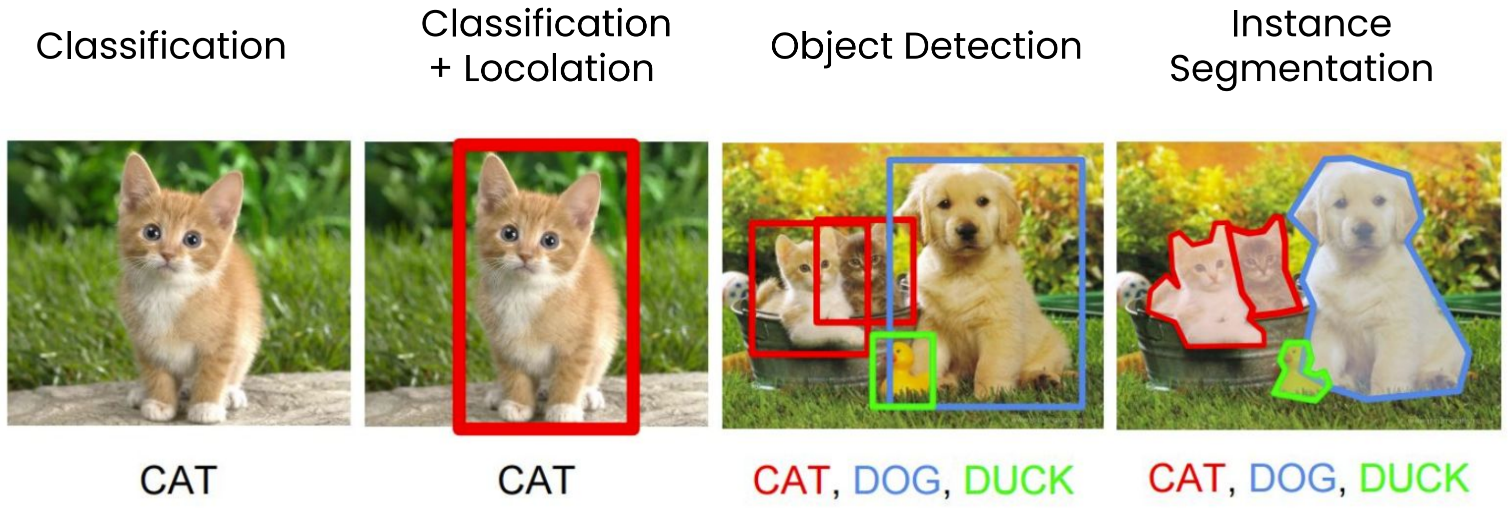
The translation is given by

$$^{A}t_B = \begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix}$$

4

Object Recognition

What is Object Detection?

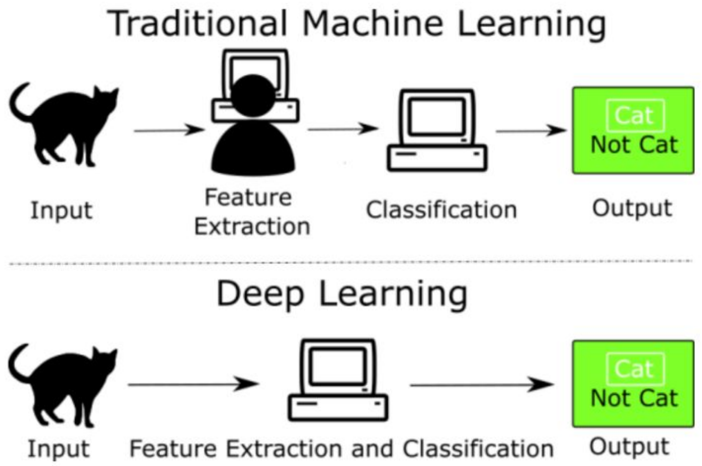


Object Detection act as a combination of image classification and object localization. It takes an image as input and produces one or more bounding boxes with the class label attached to each bounding box.

Traditional Object Detection Era

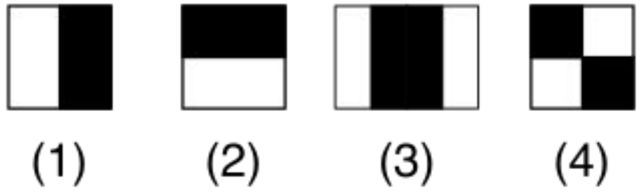
Early Algorithms:

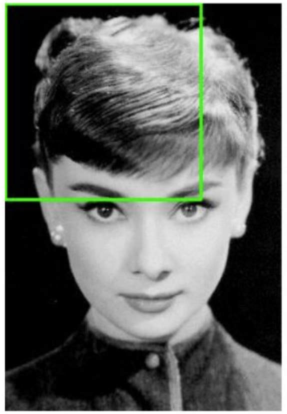
20 years ago, object detection relied on handcrafted features due to the lack of effective image representation.



Viola Jones Detectors

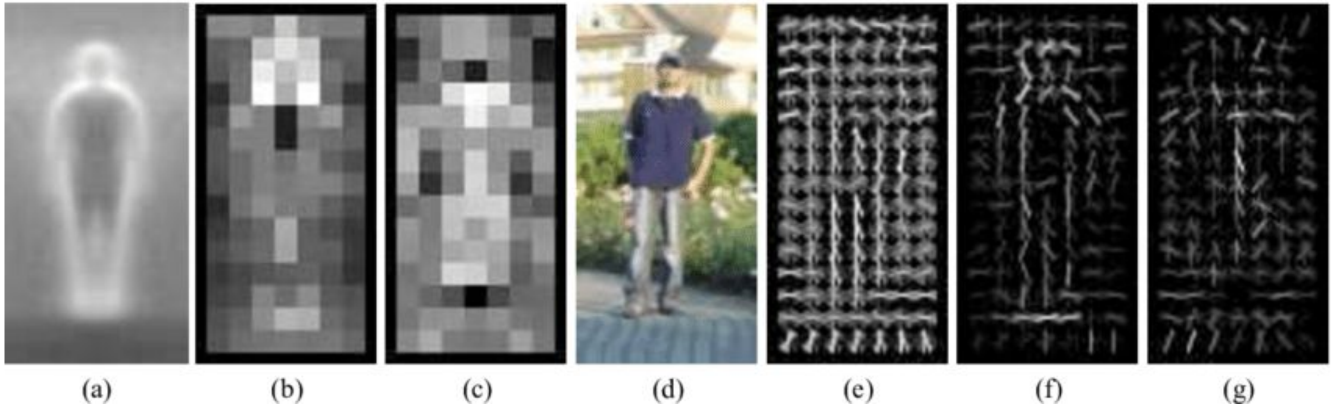
- **Development:** Created in 2001 by Paul Viola and Michael Jones.
- **Function:** Detects human faces in real-time using sliding windows to search for 'haar-like' features.
- **Speed Enhancements:**
  - Integral Image: Reduces computational complexity.
  - Adaboost Algorithm: Selects the most helpful features for face detection.
  - Detection Cascades: Multi-stage detection paradigm to reduce computational overhead.





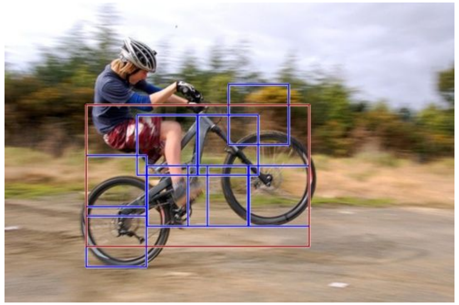
Histogram of Oriented Gradients (HOG) Detector

- **Introduction:** Proposed in 2005 by N. Dalal and B. Triggs.
- **Function:** Uses blocks to constitute gradients from pixel intensity changes.
- **Application:** Widely used for pedestrian detection.
- **Size Adaptation:** Rescales the input image to detect objects of different sizes.



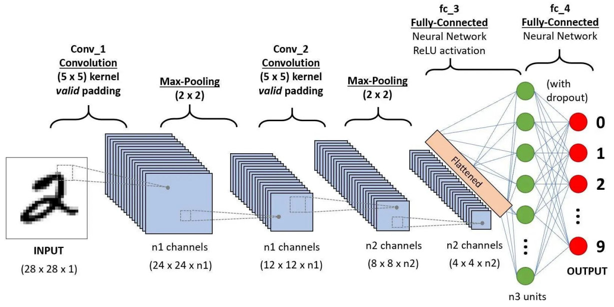
Deformable Part-based Model (DPM)

- **Proposed by:** P. Felzenszwalb in 2008, with improvements by R. Girshick.
- **Strategy:** Divides the detection of an object into its parts (e.g., car's window, body, wheels).
- **Components:** Root-filter and part-filters learned as latent variables.
- **Techniques for Accuracy:**
  - Multi-Instance Learning: Special case used for improving detection accuracy.
  - Additional Techniques: Hard negative mining, bounding box regression, context priming.
  - Cascade Architecture: Achieves acceleration without sacrificing accuracy.



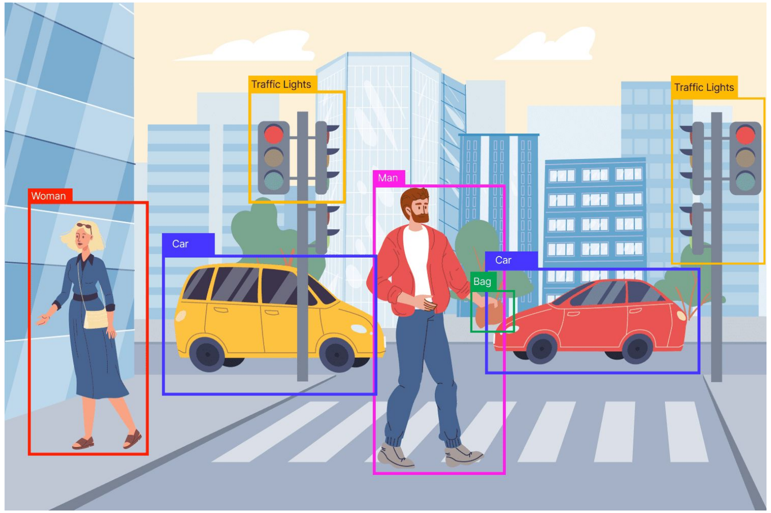
Deep Learning Era

- **Performance Plateau:** Post-2010, performance of handcrafted features plateaued.
- **CNN:** In 2012, convolutional neural networks (CNNs) enabled robust and high-level feature representations.
- **Breakthrough:** 2014 saw the proposal of Regions with CNN features (RCNN) breaking the deadlocks of object detection.
- **Genres:** Object detection is grouped into two-stage detection and one-stage detection.



Why Object Detection?

- Robot navigation
- Medical diagnosis
- Security
- Industrial inspection and automation
- Human-computer interface
- Information retrieval



Applications of Object Recognition

Printing and storage

Creating hardcopy representations of images, for example, to use as illustrations in reports, is important to many users of image processing equipment. It is also usually important to store the images so that they can be retrieved later, for instance to compare with new ones or to transmit to another worker. Both of these activities are necessary because it is rarely possible to reduce an image to a compact verbal description or a series of measurements that will communicate to someone else what we see or believe to be important in the image. In fact, it is often difficult to draw someone else's attention to the particular details or general structure that may be present in an image that we may feel are the significant characteristics present, based on our examination of that image and many more. Faced with the inability to find d
resort to passing a representation of the image on, perhaps with s
scribes this procedure in his song, "Alice's Restaurant," as "twenty
tures with circles and arrows and a paragraph on the back of ea

Printing

This book is printed in color, using high-end printing technolog
gle image processing user. But many everyday jobs can be hand
pensive machines; the quality, speed, and cost of both monoch
proving rapidly. A typical monochrome (black on white) laser
dollars and has become a common accessory to desktop compute
signed primarily to print text, and simple graphics such as line d
used to print images as well. We have come a very long way since
printing Christmas posters using Xs and Os on a teletype to repre
11). In this chapter, we will examine the technology for printing i
top computer-based image processing systems.

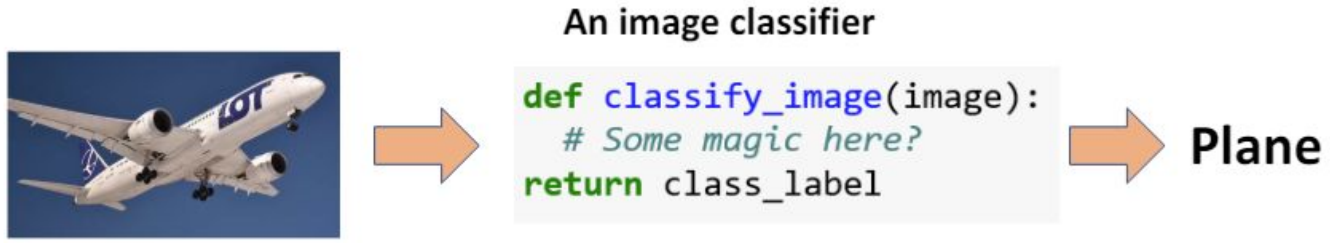
For this purpose, it does not matter whether or not the printers u
language such as PostScript®, which is used to produce smooth
mum printer resolution, so long as they allow the computer to tr

Some images from http://www.cs.utexas.edu/~grauman/research/research.html

5

K-Nearest Neighbors

Image Classification



```
def classify_image(image):
  # Some magic here?
return class_label
```

We usually take image classification in computer vision as example

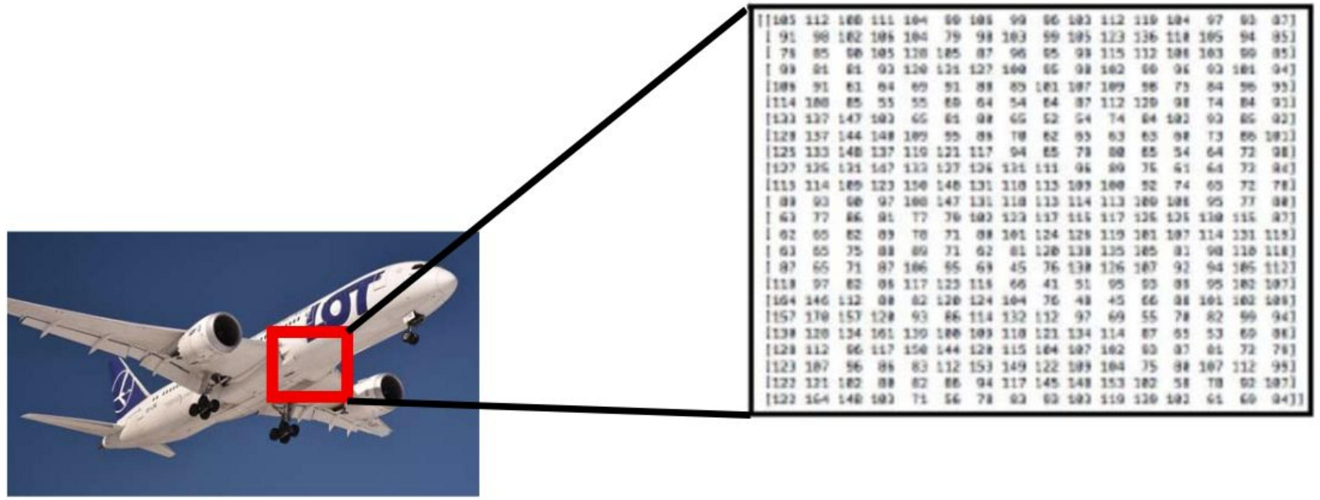
- Computer vision is the most successful field for DNNs

- Image classification is the core task of computer vision

Challenge: Semantic Gap

In computer, each pixel is represented with a number

- Computer views an image as a big grid of numbers



```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

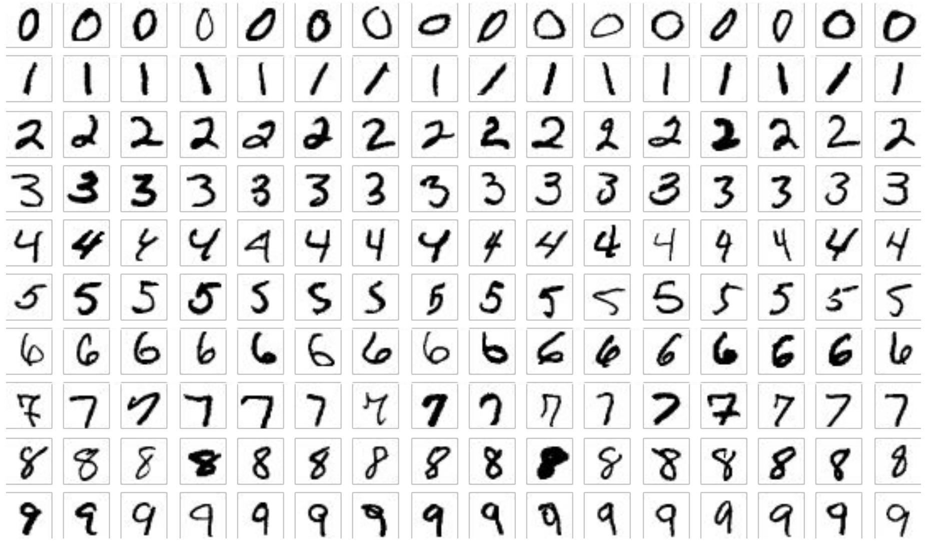
Challenge: Different Sizes, Shapes, Angles...

Machine Learning: Data-driven Approach

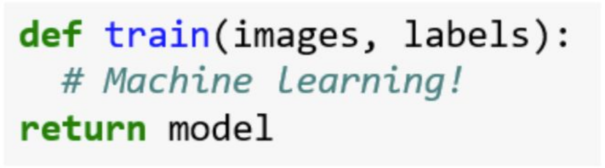
1. Collect a dataset of images and labels

2. Use Machine Learning to train a classifier

3. Evaluate the classifier on new images

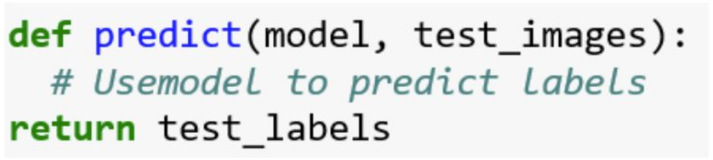


A Naive Idea: Find Similar Images



```python
def train(images, labels):
  # Machine learning!
return model
```

Memorize all data and labels



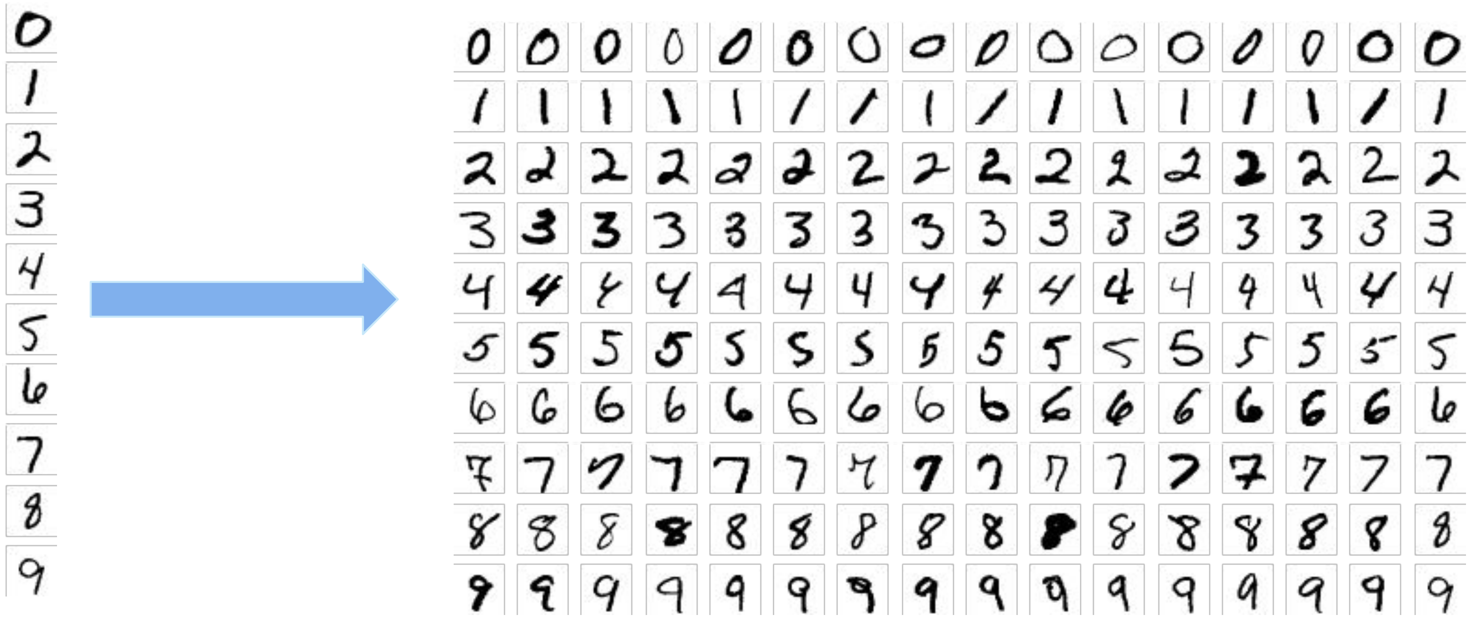
```python
def predict(model, test_images):
  # Usemodel to predict labels
return test_labels
```

Predict the label of the most similar training images

Nearest Neighbors in Training Image

Test Image

Training Image



How to Determine the Nearest?

P-norm is good metric:

p: Dimension (or pixel position)

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

10	20	24	17
8	80	109	12
12	16	178	170
4	32	233	112

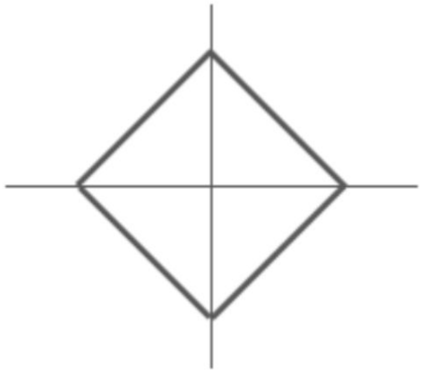
$\rightarrow$

46	12	14	1
82	43	19	121
12	10	0	30
2	32	22	108

Distance Metric

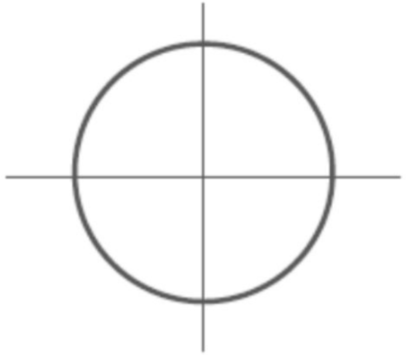
L1(Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

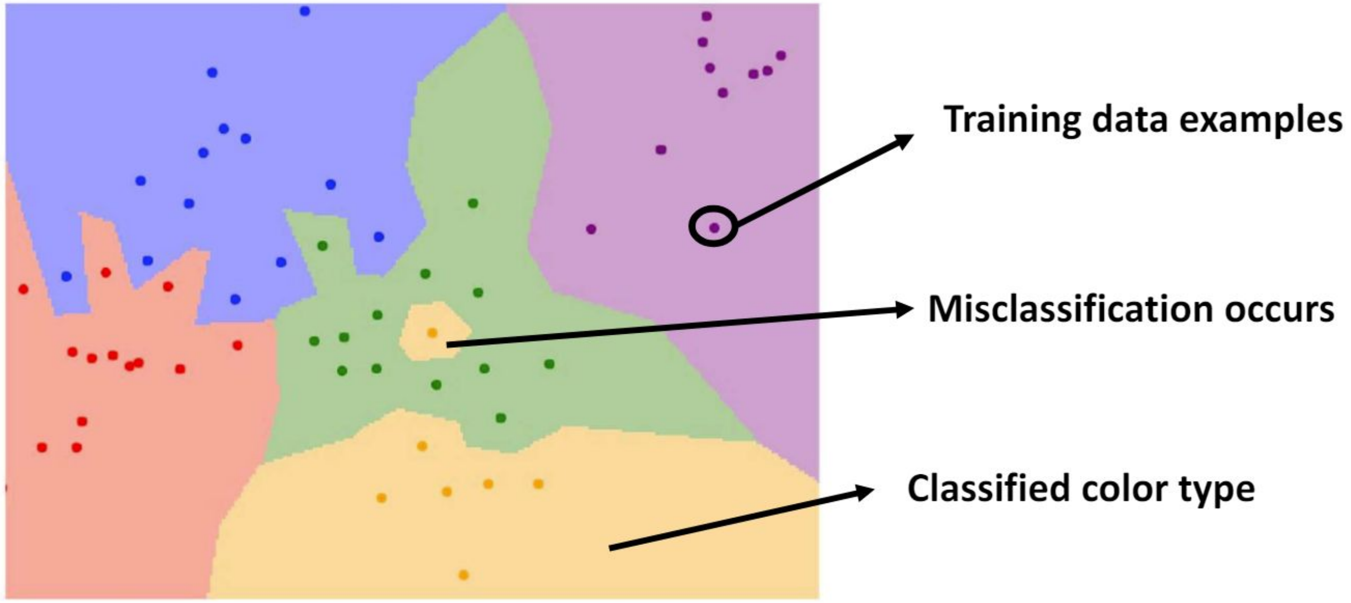


L2(Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

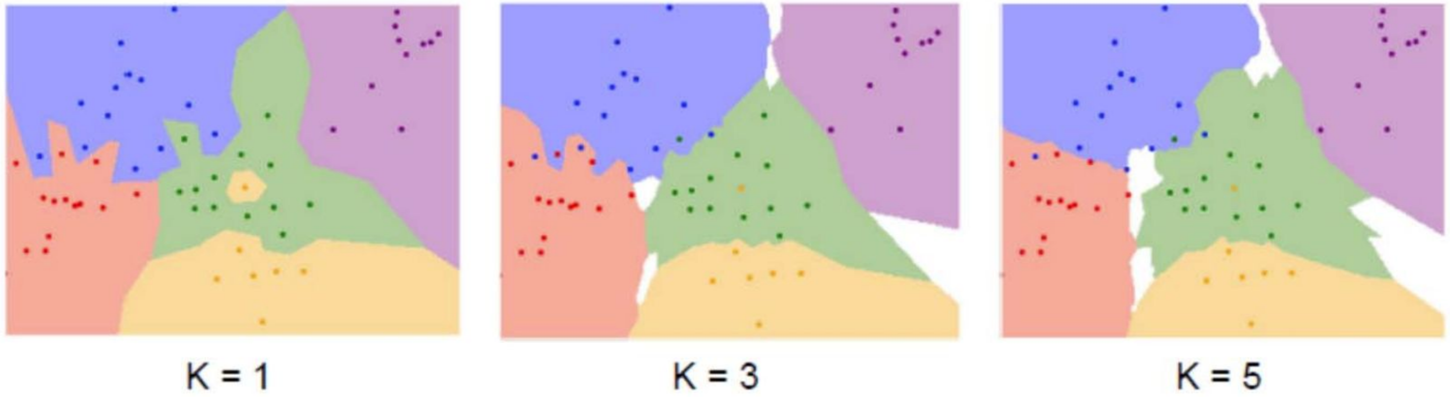


Disadvantage If Only Considering Nearest



Improvement: Using K Nearest Neighbors (KNN)

KNN: Take majority vote from K closest points



Hyperparameter

When setting up KNN, you can choose two parameters:

- What is the best value of k to use for voting?
- What is the best distance to use for measuring?

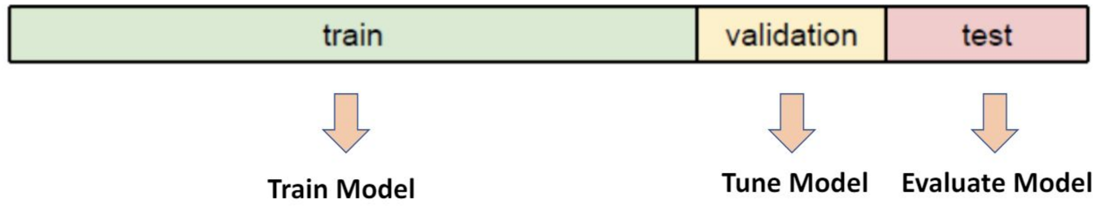
These are hyperparameters, which are not adapted by the machine learning algorithm itself.

In DNN, many hyperparameters

- # of layers/neuron, learning rate, weight decay……

Train, Validation and Test Dataset

- **Training set**
  - a set of data used to discover predictive relationships
  - simple words: help to learn the model
- **Test set**
  - a set of data used to assess the strength and utility of a predictive relationship
  - simple words: help to evaluate a learned model
- **Validation set**
  - a set of examples used to tune the hyperparameters



Summary of Non - parametric KNN

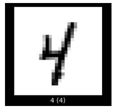
Training Data

→ ML Model f(x_train)

Pros: Fast Training
Easy to implement

Cons: Slow Testing
Store all training data

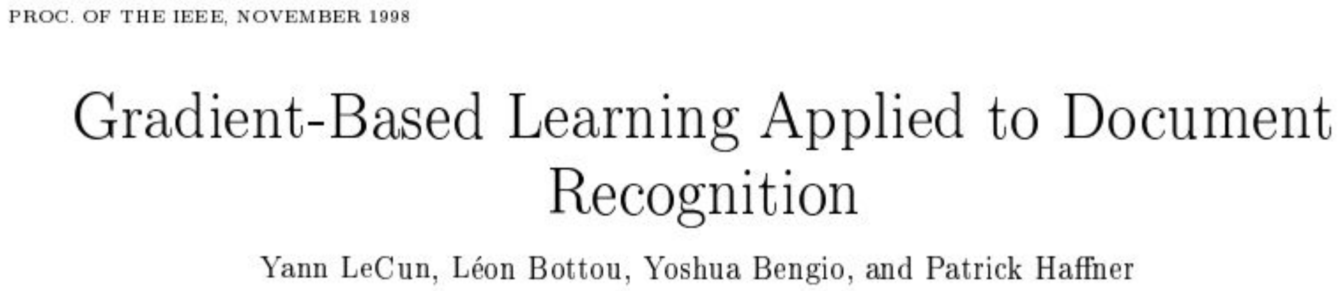
Test Data



→ ML Model f(x_train, x_test) → classification result

6 Convolutional Neural Networks

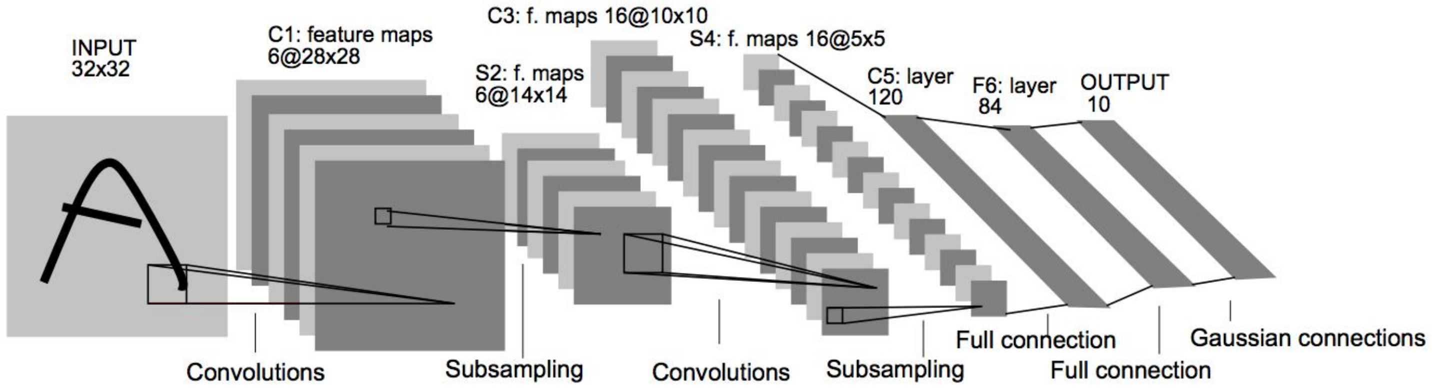
History



PROC. OF THE IEEE, NOVEMBER 1998

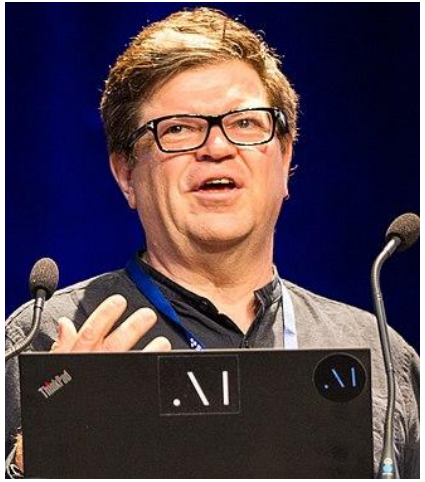
Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner



Yann LeCun

- Machine Learning
- Computer Vision
- Mobile Robotics
- Computational Neuroscience



Key Contributions:

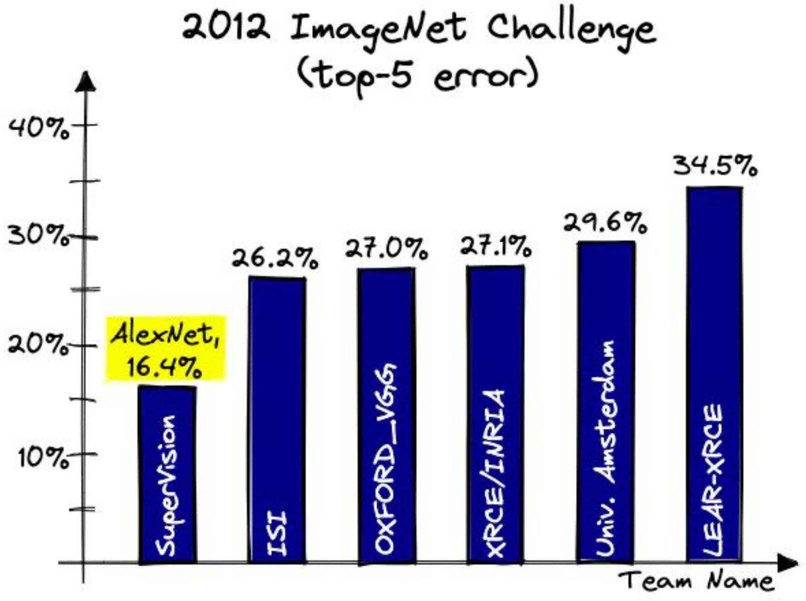
- Optical Character Recognition & Computer Vision: Known for pioneering work using convolutional neural networks (CNNs).
- Often referred to as one of the "Godfathers of AI" and "Godfathers of Deep Learning."

Convolutional Network Demo from 1989

https://www.youtube.com/watch?v=FwFduRA_L6Q

ImageNet Challenge in 2012

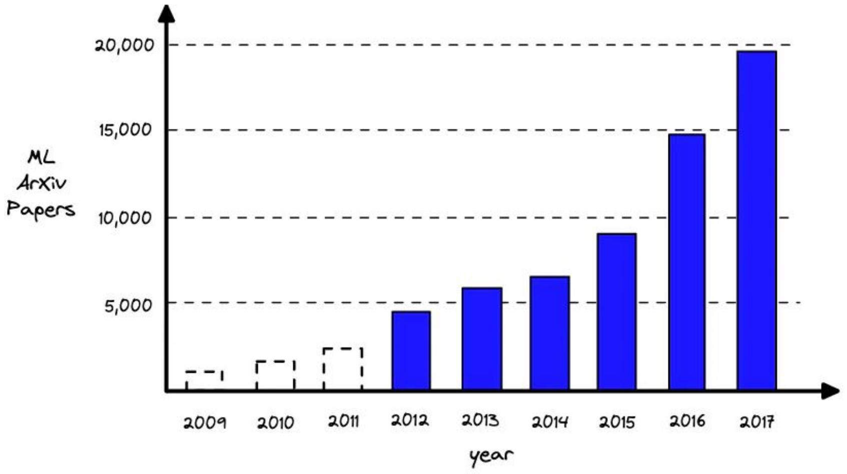
ImageNet team managed to organize the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Competitors had to correctly classify images and detect different objects and scenes across a trimmed list of 1,000 ImageNet categories. Every year, the team that produced the model with the lowest error rate won



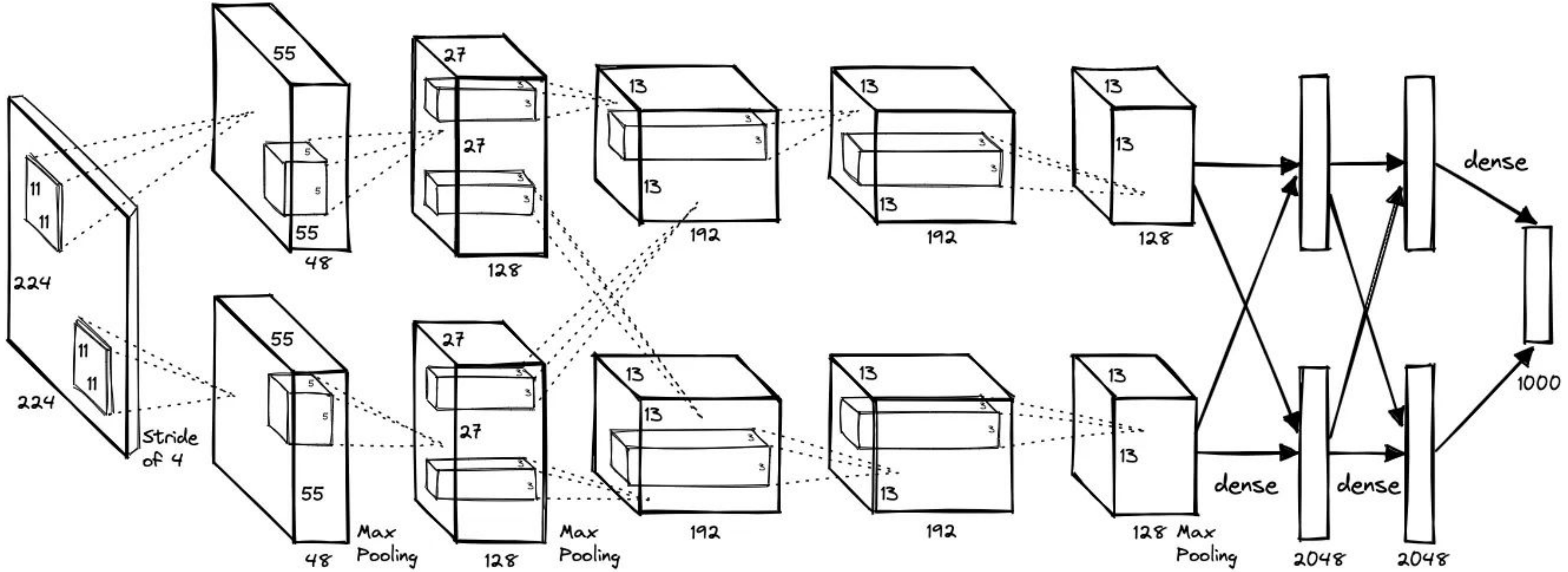
AlexNet

AlexNet didn't just win; it dominated.

AlexNet was unlike the other competitors. This new model demonstrated unparalleled performance on the largest image dataset of the time, ImageNet. This event made AlexNet the first widely acknowledged, successful application of deep learning. It caught people's attention with a 9.8 percentage point advantage over the nearest competitor

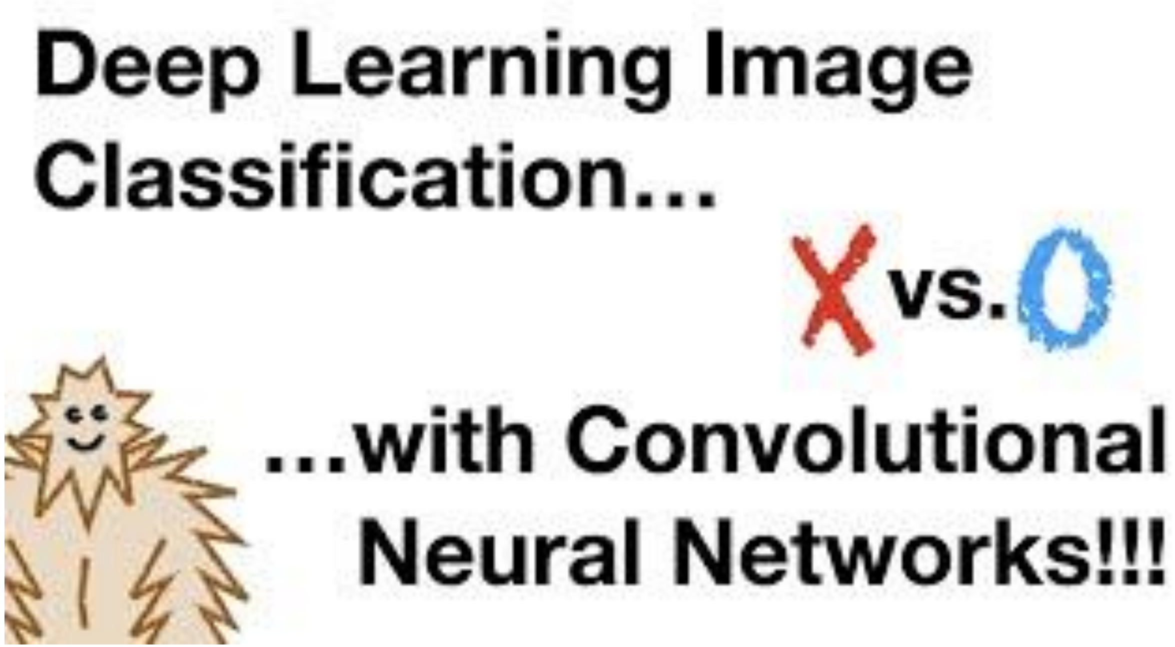


AlexNet



224
224
11
11
11
11
Stride of 4
55
55
48
55
55
48
Max Pooling
27
27
128
27
27
128
Max Pooling
5
5
3
13
13
192
13
13
192
13
13
192
13
13
192
13
13
128
13
13
128
Max Pooling
dense
dense
dense
2048
2048
1000

CNN


Deep Learning Image Classification...
X vs. O
...with Convolutional Neural Networks!!!

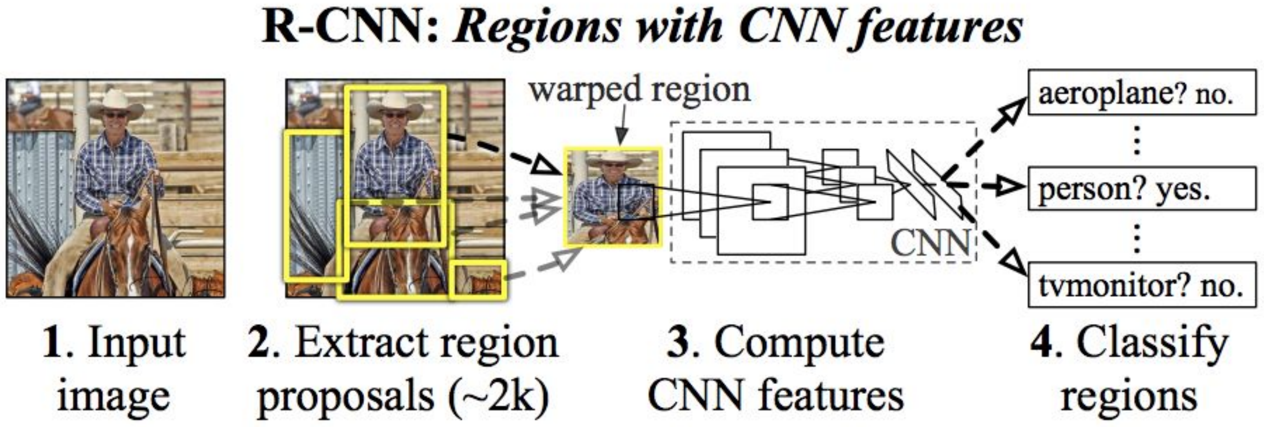
https://www.youtube.com/watch?v=HGwBXDKFk9I

7

R-CNN

RCNN : Region Proposal + CNN

Region-based Convolutional Neural Network (R-CNN) is a type of deep learning architecture used for object detection in computer vision tasks. RCNN was one of the pioneering models that helped advance the object detection field by combining the power of convolutional neural networks and region-based approaches.



Selective Search

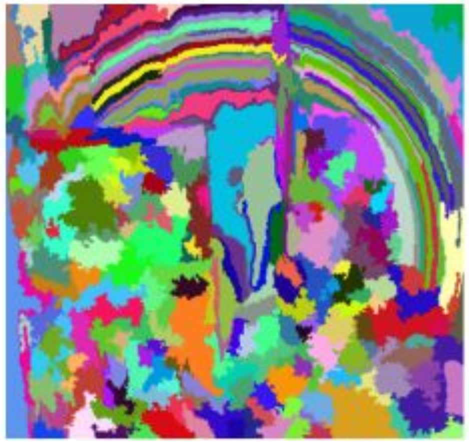
Selective Search is a **region proposal algorithm** for object detection tasks. It starts by over-segmenting the image based on intensity of the pixels. Selective Search then takes these oversegments as initial input and performs the following steps

1. Add all bounding boxes corresponding to segmented parts to the list of regional proposals
2. Group adjacent segments based on similarity
3. Go to step 1

At each iteration, larger segments are formed and added to the list of region proposals. Hence we create region proposals from smaller segments to larger segments in a bottom-up approach.

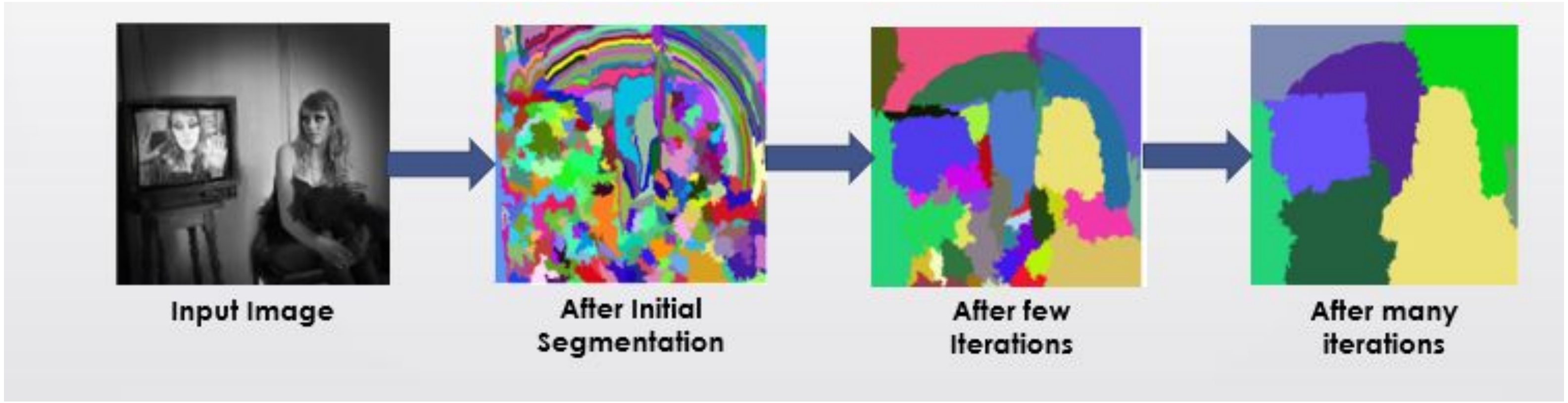
Selective Search

1. Generate initial sub-segmentation of input image using the method describe by Felzenszwalb et al in his paper “Efficient Graph-Based Image Segmentation “.



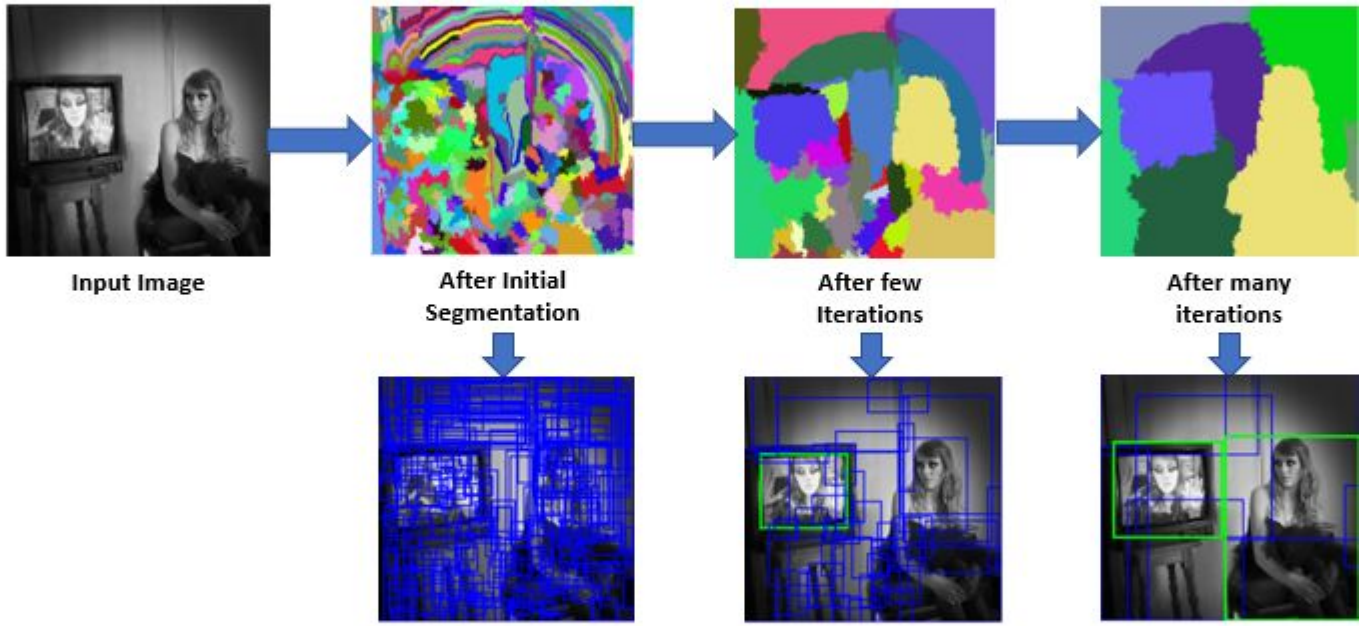
Selective Search

2. Recursively combine the smaller similar regions into larger ones. We use Greedy algorithm to combine similar regions to make larger regions.



Selective Search

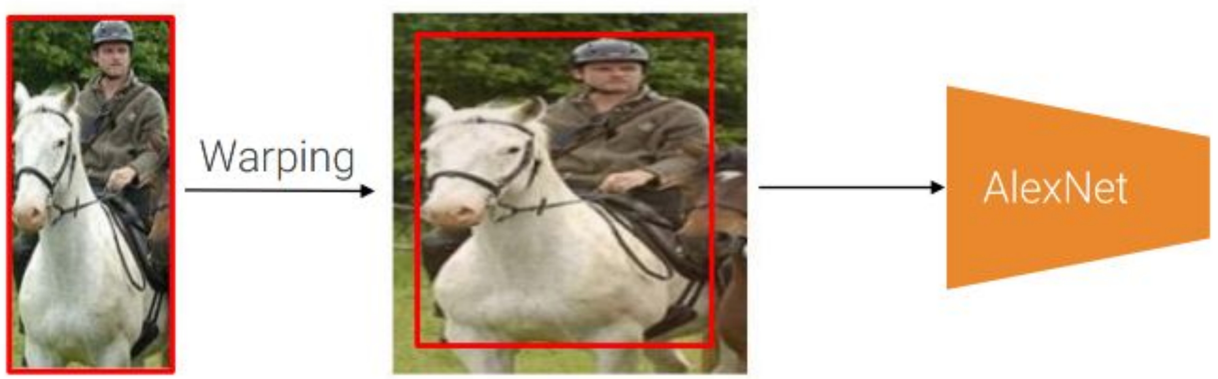
3. Use the segmented region proposals to generate candidate object locations



Feature Extraction

Once the region proposals are generated, approximately 2,000 regions are extracted and anisotropically warped to a consistent input size that the CNN expects (e.g., 224x224 pixels) and then it is passed through the CNN to extract features.

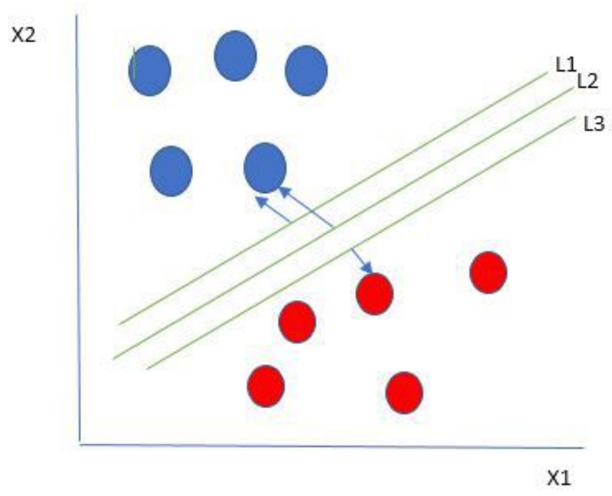
The output of the CNN is a high-dimensional feature vector representing the content of the region proposal.



Object Classification

The extracted feature vectors from the region proposals are fed into a separate machine learning classifier for each object class of interest. R-CNN typically uses Support Vector Machines (SVMs) for classification. For each class, a unique SVM is trained to determine whether or not the region proposal contains an instance of that class.

During training, positive samples are regions that contain an instance of the class. Negative samples are regions that do not.

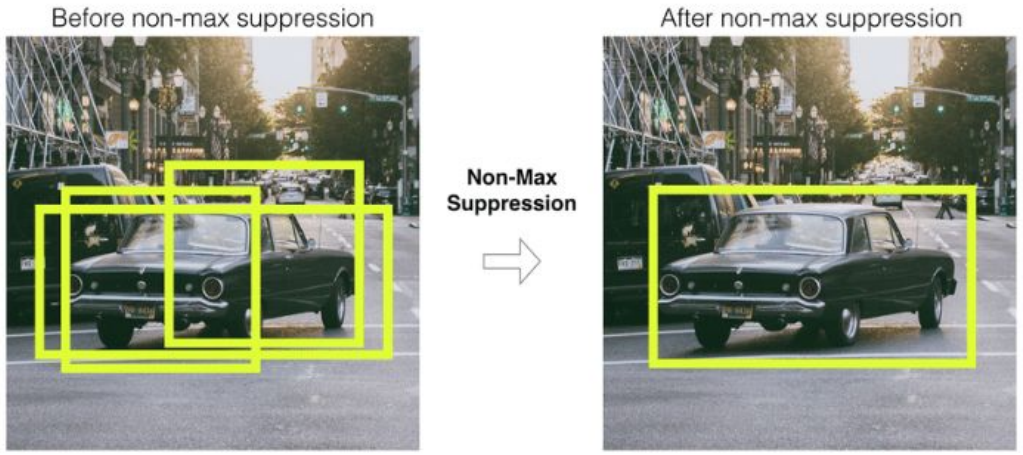


Support Vector Machines (SVMs)

https://www.youtube.com/watch?v=_YPScrckx28

Non-Maximum Suppression (NMS)

After classifying and getting bounding boxes for each region proposal, R-CNN applies non-maximum suppression to eliminate duplicate or highly overlapping bounding boxes. NMS ensures that only the most confident and non-overlapping bounding boxes are retained as final object detections.



R-CNN Pro and Cons

Pros

- Accurate Object Detection: Leverages region-based convolutional features for precise localization and recognition.
- Robustness to Object Variations: Handles different sizes, orientations, and scales, suitable for diverse objects and complex backgrounds.
- Flexibility: Adaptable to tasks, including instance segmentation and object tracking, by modifying the final network layers.

Cons

- Slow Inference: Sequential processing of region proposals results in latency, unsuitable for real-time applications.
- Overlapping Region Proposals: Generates multiple overlapping proposals, leading to redundant computation.
- Not End-to-End: Separate modules for region proposal and classification can result in suboptimal performance

8

Top Models

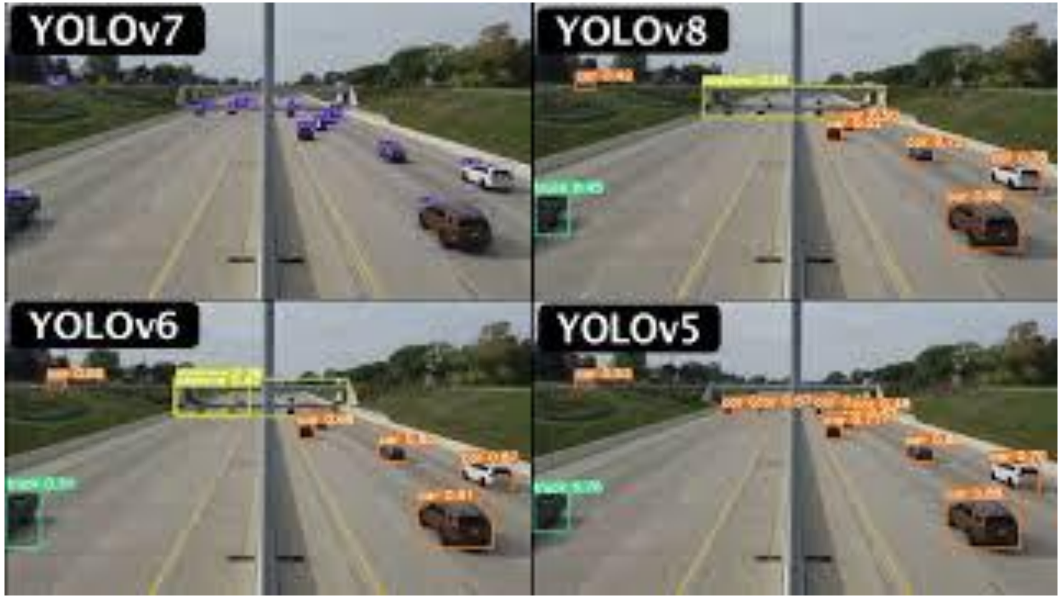
YOLO (You Only Look Once) Series

- **Architecture:** YOLO’s architecture is inherently different from the region’s proposal-based methods. Instead of proposing potential object locations and then classifying them, YOLO performs both tasks simultaneously. This concurrent approach is what gives YOLO its name and its speed.
- **Analysis:** Its standout feature is its speed. YOLO can process images in real-time, making it one of the fastest object detection models available. However, this design also means that YOLO might struggle with small objects or objects that are close together, as they might fall into the same grid cell.

YOLO Application

- **Traffic Management:** YOLO’s real-time processing capability is invaluable in traffic management systems. For instance, in smart cities, YOLO is used to monitor intersections, detect traffic violations, and predict congestion. Its ability to swiftly detect vehicles, pedestrians, and even traffic signs aids in real-time decision-making, optimizing traffic flow and enhancing safety.
- **Real-time Surveillance:** Security systems leverage YOLO for intruder detection, unauthorized access, and monitoring restricted areas. The speed of YOLO ensures that security systems detect and address threats immediately.
- **Sports Analytics:** In sports, YOLO aids in player tracking, ball trajectory prediction, and game analysis. For example, in football, YOLO can track players’ movements, analyze team formations, and even predict potential goal-scoring opportunities.

YOLO

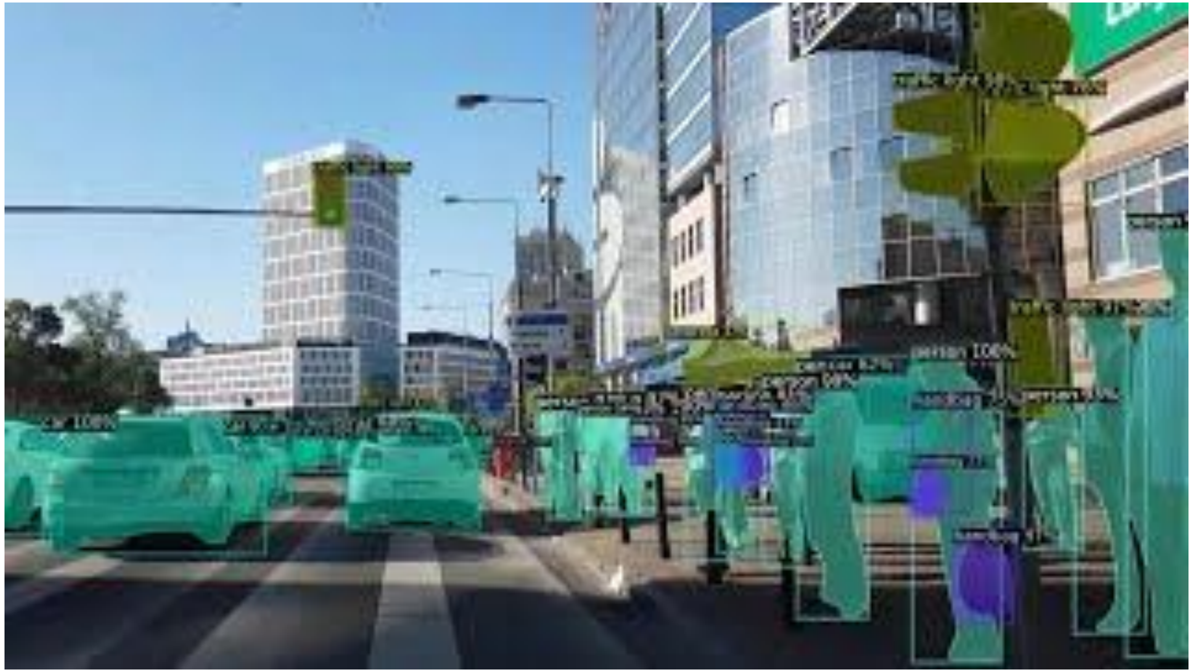


https://www.youtube.com/watch?v=QOC6vgnWnYo

Detectron2

- **Architecture:** Detectron2 is the next-generation library developed by Facebook AI Research, succeeding Detectron and maskrcnn-benchmark. Built on the PyTorch framework, it offers a modular library rich with a variety of pre-trained models. Detectron2 provides state-of-the-art detection and segmentation algorithms, supporting numerous computer vision research projects and production applications.
- **Analysis:** Detectron2 is not just an incremental improvement over its predecessors, but a complete overhaul. It introduces new capabilities such as panoptic segmentation, Densepose, Cascade R-CNN, PointRend, DeepLab, ViTDet, MViTv2, and more. The library is both a research tool and a production-ready system. Its modular nature allows for easy customization, making it a preferred choice for researchers and developers alike.

Detectron2



https://www.youtube.com/watch?v=GwQxaE8teRc

EfficientDet

- **Architecture:** EfficientDet is an innovative object detection model that emphasizes efficiency without compromising on performance. It’s built on the foundation of compound scaling – a method that uniformly scales the resolution, depth, and width of the network. This ensures that the model remains balanced and optimized across various device capabilities. One of the architectural innovations in EfficientDet is the introduction of a weighted bi-directional feature pyramid network (BiFPN). This BiFPN facilitates easy and rapid multiscale feature fusion, enhancing the model’s ability to detect objects of varying sizes.
- **Analysis:** The BiFPN allows for faster feature fusion across different scales. Additionally, the compound scaling method ensures that all components of the model (backbone, feature network, and prediction networks) scale uniformly, maintaining a harmonious balance.

EfficientDet

https://www.youtube.com/watch?v=AEgkLRZnm3I

9

AprilTags

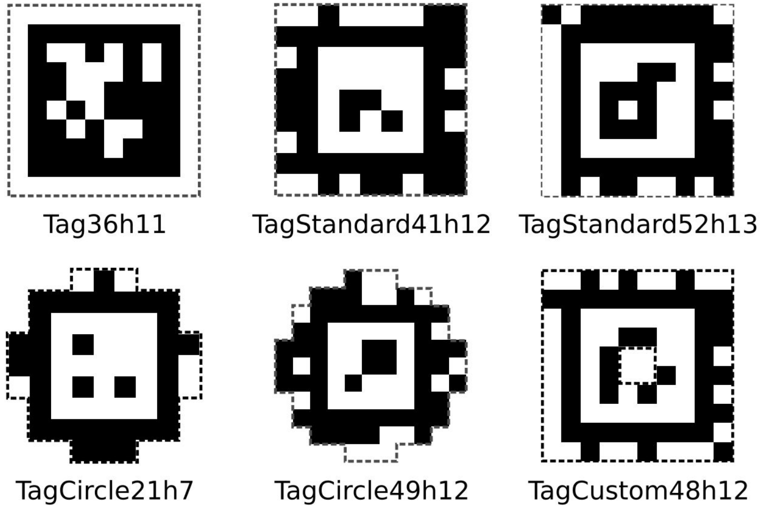
AprilTags

AprilTag is a visual fiducial system, useful for a wide variety of tasks including augmented reality, robotics, and camera calibration. Targets can be created from an ordinary printer, and the AprilTag detection software computes the precise 3D position, orientation, and identity of the tags relative to the camera. The AprilTag library is implemented in C with no external dependencies. It is designed to be easily included in other applications, as well as be portable to embedded devices. Real-time performance can be achieved even on cell-phone grade processors.

https://april.eecs.umich.edu/software/apriltag

AprilTags

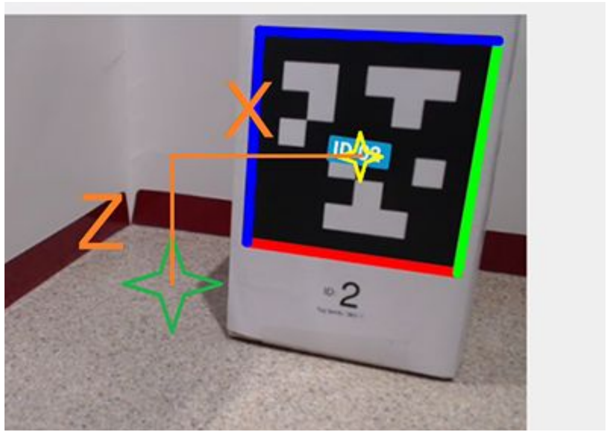
AprilTags are conceptually similar to QR Codes, in that they are a type of two-dimensional bar code. However, they are designed to encode far smaller data payloads (between 4 and 12 bits), allowing them to be detected more robustly and from longer ranges. Further, they are designed for high localization accuracy—you can compute the precise 3D position of the AprilTag with respect to the camera.



VisionPortal from FIRST Tech Challenge

VisionPortal is a comprehensive new interface for vision processing. VisionPortal offers key capabilities of AprilTag and EasyOpenCV, along with TensorFlow Object Detection (TFOD)

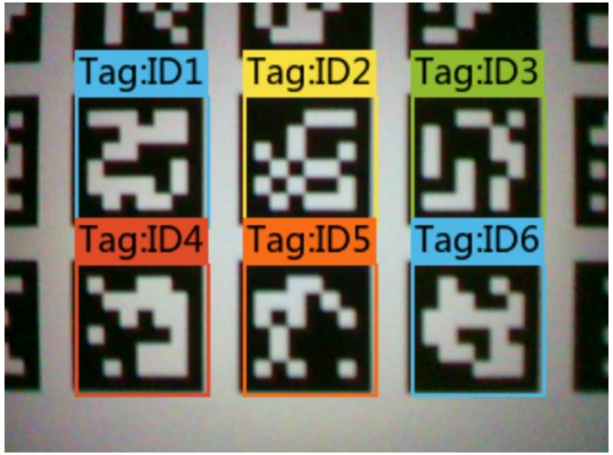
- **AprilTag**
  - Detects an ID code.
  - Estimates pose (position and rotation).
- **TensorFlow Object Detection (TFOD)**
  - Recognizes 3D physical objects
  - Defines a "bounding box" and estimates heading.



HuskyLens

Huskylens AI Vision Sensor can be used in the FIRST Tech Challenge Game. It offers Tag Recognition function which can detect tags, and learn, recognize, track specified tags.

https://www.youtube.com/watch?v=Kv1HiN7nmg4



10

OpenCV

What Is OpenCV?

- OpenCV: An open-source computer vision library available at SourceForge.net.
- Languages: Written in C and C++, with active development for interfaces in Python, Ruby, Matlab, and others.
- Platforms: Runs on Linux, Windows, and Mac OS X.

Design and Efficiency

- Computational Efficiency: Designed for real-time applications.
- Optimized C Code: Takes advantage of multicore processors.

OpenCV Goals

- Vision Infrastructure: Provides a simple-to-use framework for building sophisticated vision applications quickly.
- Functionality: Contains over 500 functions for various vision tasks:
    - Factory product inspection
    - Medical imaging
    - Security
    - User interface
    - Camera calibration
    - Stereo vision
    - Robotics

Core Operations

Function	Purpose
imread(image_path, flag)	This method is used to read an image from its path
imshow(window_name, image)	It is used to show the image in the window.
imwrite(filename, image)	This method is used to write or save an image using OpenCV.

Reading Images

Image Reading Modes	Description
cv2.IMREAD_COLOR	Read colored images
cv2.IMREAD_GRAYSCALE	Read Grayscale images
cv2.IMREAD_UNCHANGED	Read images with alpha channel

Drawing Shapes and Text on Images

Syntax	Purpose
line(image, start_coordinates, end_coordinates, color_in_bgr, line_thickness)	By using this function we can create a line in the image from start coordinates to end coordinates with a certain thickness which can be mentioned specifically as well.
rectangle(image,top_left_vertex_coordinates, lower_right_vertex_coordinates, color_in_bgr, thickness)	This function is used to create a box with a certain thickness and color which can be specified as well.
circle(image, center_coordinates, radius, color, thickness)	It is used to draw a circle whose centre and radius length is given with a certain thickness and the colour of the strokes of the circle.

Drawing Shapes and Text on Images

Syntax	Purpose
polylines(image, [pts], isClosed, color, thickness)	It is used to draw a polygon on any image whose vertex coordinates are provided.
putText(image, ‘TextContent’, ‘text_starting_point_coordinates’, ‘fontToBeUsed’, ‘font_size’, ‘text_color’, ‘text_thickness’, ‘line_type’)	It is used to write some text on the image loaded.

Arithmetic Operations on Images

Function	Uses
add(image1, image2)	This function is used to add two images.
subtract(image1, image2)	This function is used to subtract two images.
addWeighted(image1, weight1, image2, weight2, gammaValue)	This is also known as Alpha Blending. This is nothing but a weighted blending process of two images.

Arithmetic Operations on Images

Function	Uses
bitwise_and(image1, image2, destination, mask)	This performs bitwise and logical operations between two images.
bitwise_or(image1, image2, destination, mask)	This performs bitwise or logical operations between two images.
bitwise_not(image, destination, mask)	This performs bitwise not logical operations between an image and a mask.
bitwise_xor(image1, image2, destination, mask)	This performs bitwise xor logical operations between two images.

Geometric Transformations on Image

Function	Purpose
resize(image, width, height, interpolation_method)	By using this method we can resize the image at hand for specific width and height. Some of the methods that can be used for interpolation are as below: ● cv2.INTER_AREA ● cv2.INTER_CUBIC ● cv2.INTER_LINEAR

Geometric Transformations on Image

Function	Purpose
M = cv2.getRotationMatrix2D(center, angle, scale) warpAffine(image, M, (width, height))	Image rotation is a common image processing routine with applications in matching, alignment, and other image-based algorithms, in image rotation the image is rotated by a definite angle. We can also use cv2.rotate() function for image rotation.

Geometric Transformations on Image

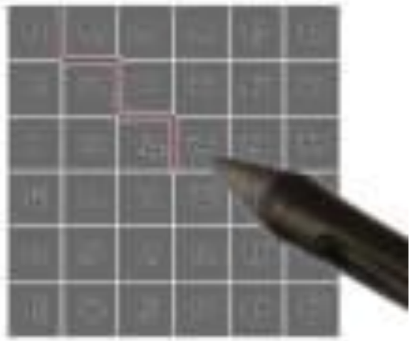
Function	Purpose
matrix = cv2.getPerspectiveTransform(src, dst) warpPerspective(image, matrix, dsize)	In Perspective Transformation, we can change the perspective of a given image or video for getting better insights into the required information. We need to provide the points on the image from which want to gather information by changing the perspective. We also need to provide the points inside which we want to display our image.

Edge Detection

Edge Detection

Basic Idea

* Look for a neighborhood with strong signs of change



Edge/Line Detection (Features)

Operation	Uses
cv2.Canny(image, T_lower, T_upper, aperture_size, L2Gradient)	The Canny Edge Detection is an algorithm used for edge detection. It reduces noise using Gaussian Smoothing and computes image gradient using The Sobel filter.
cv2.HoughLines(edges, 1, np.pi/180, 200)	The Hough Transform is a method that is used in image processing to detect any shape if that shape can be represented in mathematical form. It can detect the shape even if it is broken or distorted a little bit.

Edge/Line Detection (Features)

Operation	Uses
cv2.HoughCircles(image, cv2.HOUGH_GRADIENT, 1, 20, param1 = 50, param2 = 30, minRadius = 1, maxRadius = 40)	Circle detection finds a variety of uses in biomedical applications, ranging from iris detection to white blood cell segmentation.
cv2.cornerHarris(src, dest, blockSize, kSize, freeParameter, borderType)	Harris Corner detection algorithm was developed to identify the internal corners of an image. The corners of an image are basically identified as the regions in which there are variations in the large intensity of the gradient in all possible dimensions and directions.

Changing the Colorspace of Images

cv2.cvtColor(image, conversion_scale)	Some of the commonly used ways in which color coding of the images are changed is as shown below: ● cv2.COLOR_BGR2GRAY ● cv2.COLOR_BGR2HSV ● cv2.COLOR_BGR2LAB ● cv2.COLOR_BGR2YCrCb

hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
lower_blue = np.array([110,50,50])
upper_blue = np.array([130,255,255])
cv2.inRange(hsv, lower_blue, upper_blue)
	OpenCV provides functions for the
	detection of a specific color
	in live-stream video content. A video is composed of infinite frames at different time instants.

Working With Videos

cv2.VideoCapture(“file_name.mp4”)	To capture a video using OpenCV, we need to create a VideoCapture object. VideoCapture has the device index or the name of a video file. The device index is just the number to specify which camera. If we pass 0 then it is for the first camera, 1 for the second camera.

cv2.VideoCapture(File_path)
 	
cv2.read()
 	
cv2.imwrite(filename, img[, params])
	Using OpenCV, techniques such as image scanning, and face recognition can be accomplished quite easily. We can
	
	extract images from videos as well using the OpenCV module along with the os module.

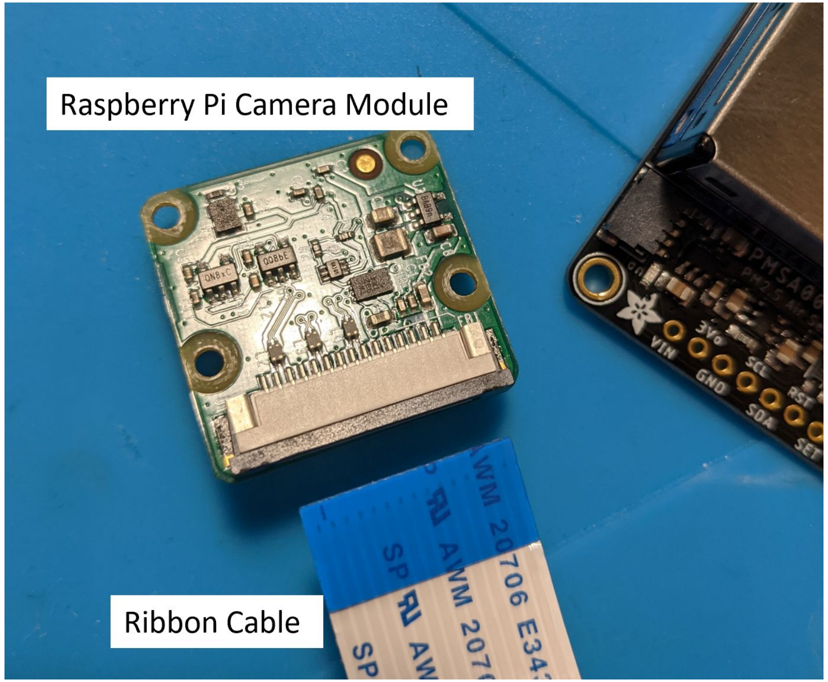
11

Raspberry Pi Camera

Raspberry Pi Camera Module

Raspberry Pi Camera Module

- Connect the Camera
- Please power off your Pi before plugin the Camera
- Please it's at the correct orientation



Test Camera

To test camera, use following command in terminal:

```
rpicam-hello
```

Python Packages for PiCamera

Create a new virtual environment with default packages including picamera2:

```
python -m venv --system-site-packages ~/env/opencv
```

Install opencv:

```
source ~/env/opencv/bin/activate

pip install opencv-python
```

Capture & Display Image

```
from picamera2 import Picamera2

import cv2

picam2 = Picamera2()

picam2.start()

image = picam2.capture_array()    # Capture an image with RGB encoding

image_bgr = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)        # Convert from RGB to BGR

cv2.imshow("Captured Image", image_bgr)

cv2.waitKey(0)
```