Deep learning Models for Image/Video

Prof. Venki Muthukumar

What is OpenCV

- **OpenCV: Open-Source Computer Vision & Machine Learning software library**
  - Created in 1999 by Intel
  - Supported from 2008 by Willow Garage(a SME dedicated to hardware & open-source software for personal robotics applications)

    www.willowgarage.com/pages/software/overview

    - Willow Garage also supports the Point Cloud Library (PCL)
- **Cross-platform**
  - Windows|Linux|Android|MacOS|iOS*
- **Free under BSD License**
  - Commercial & non-commercial applications

OpenCV

- **Written in C / C++**
- **Extended to Python**
  - Stable source code **opencv.org/releases**
  - Developments
    - Source code **github.com/opencv**
    - Wiki **github.com/opencv/opencv/wiki**
- **APIs available for a variety of programming languages**

python Java MATLAB NVIDIA CUDA OpenCL

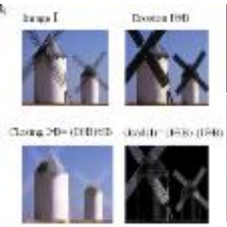
- **Current stable version is 4.13.0**

OpenCV Overview:

OpenCV

opencv.willowgarage.com

> 2000 algorithms



General Image Processing Functions

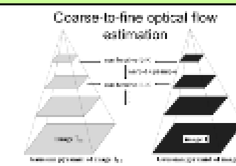
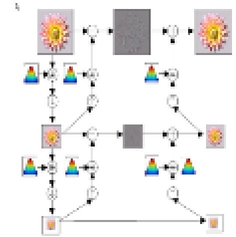
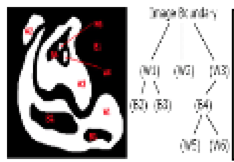
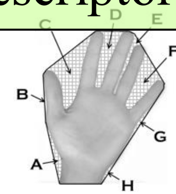


Image Pyramids

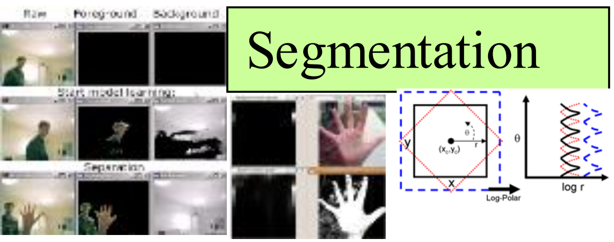


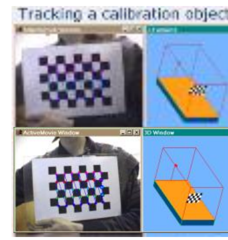


Geometric descriptors

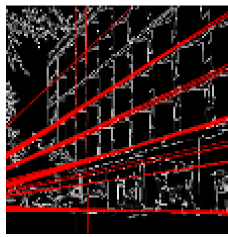


Segmentation

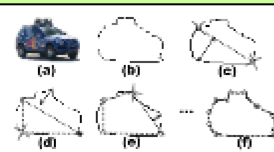


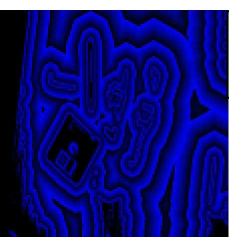


Camera calibration, Stereo, 3D

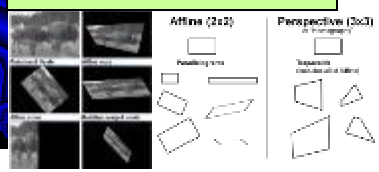


Features

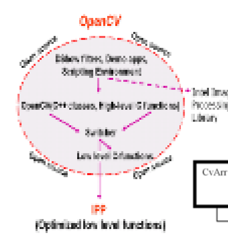


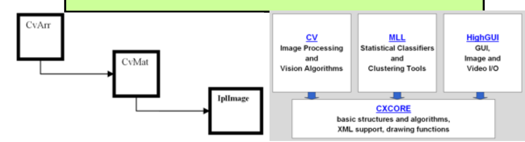


Transforms

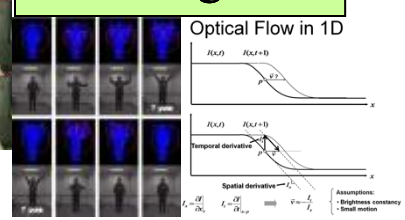


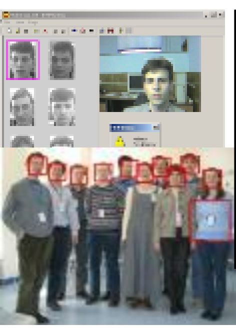
Utilities and Data Structures





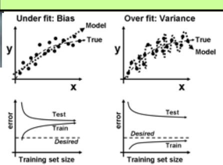
Tracking

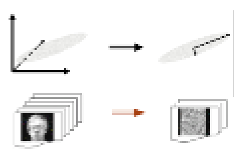




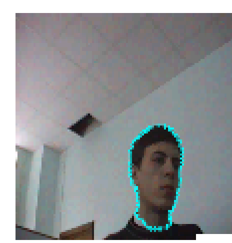
Machine Learning:

- Detection,
- Recognition

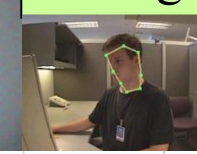


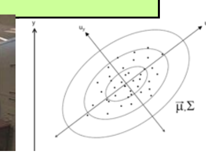


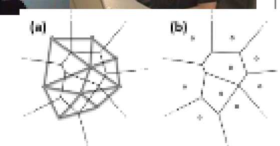
Matrix Math



Fitting

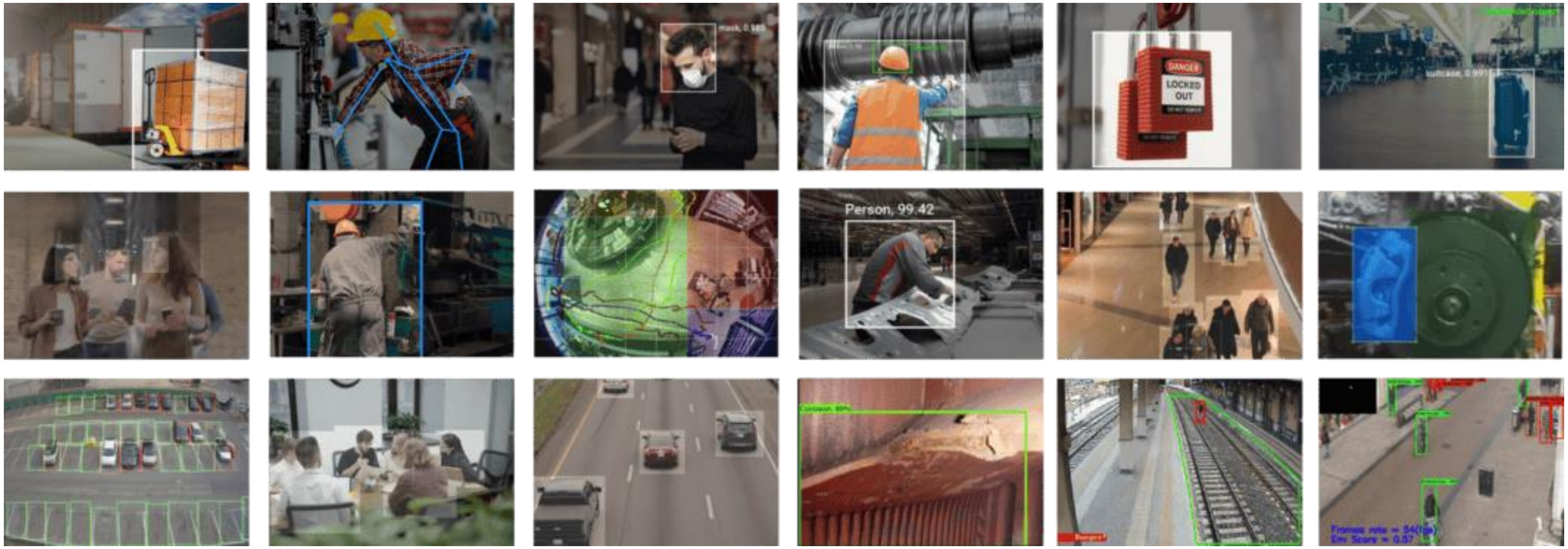






Where is OpenCV Used?

- Google Maps, Google street view, Google Earth, Books
- Academic and Industry Research
- Safety monitoring (Dam sites, mines, swimming pools)
- Security systems
- Image retrieval
- Video search
- Structure from motion in movies
- Machine vision factory production inspection systems
- Robotics



OpenCV & DL

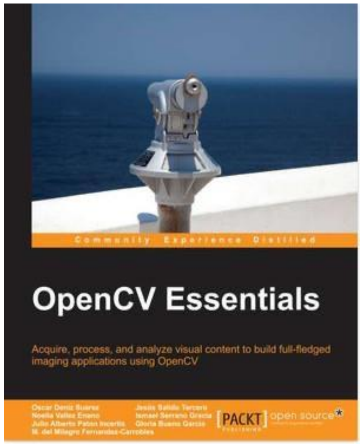
- Tensorflow, Pytorch, Caffe, MxNet, CNTK, Keras etc. are the libraries use OpenCV to build deep learning vision models.

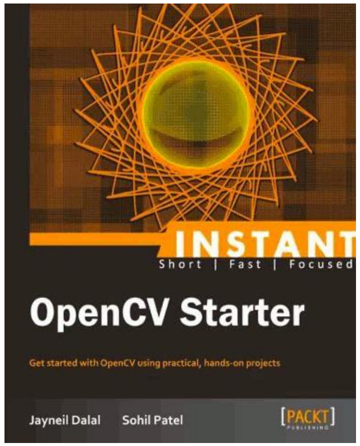
OpenCV Resources

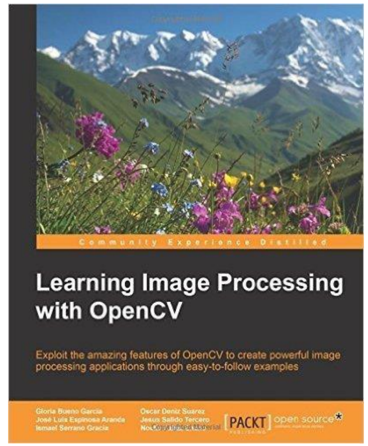
- **Online documentation** docs.opencv.org
- Reference|Tutorials|QuickStart|Examples
- **Online resources** opencv.org>Resources
- Books|Publications|Usefullinks
- **Q&A Forum** answers.opencv.org

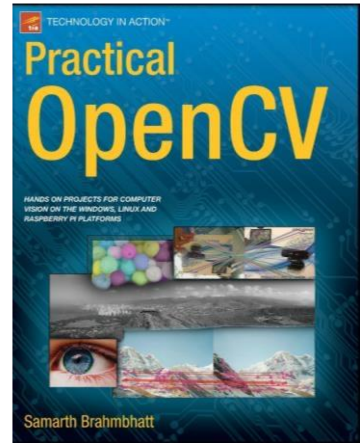
Embedded ML

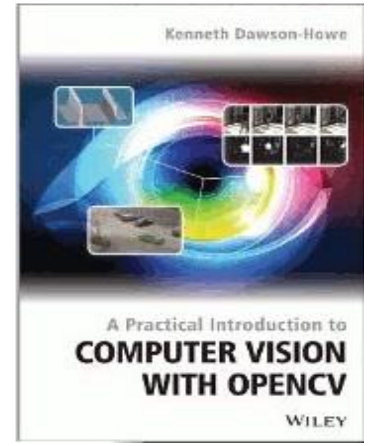
Bibliography

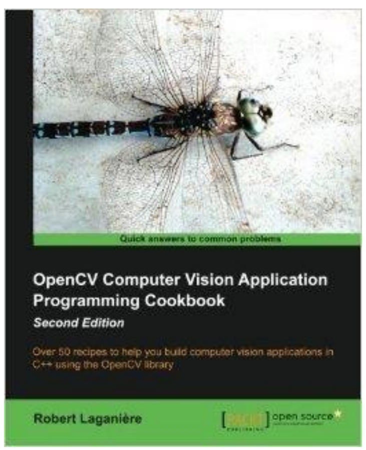


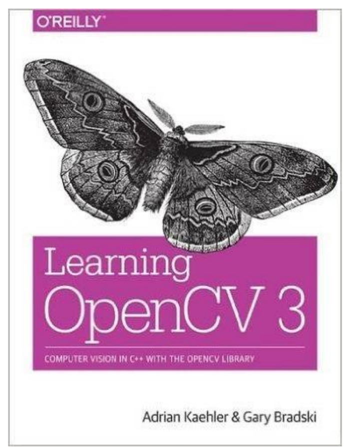


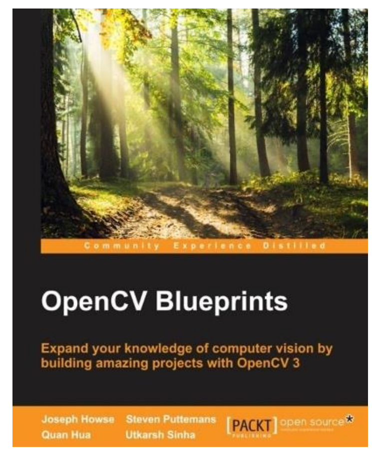






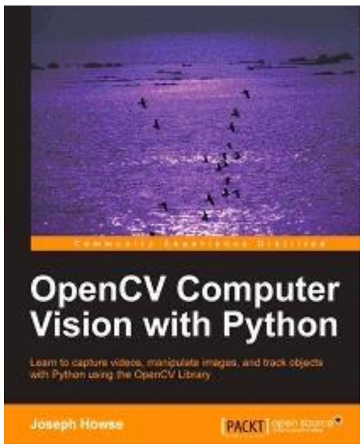


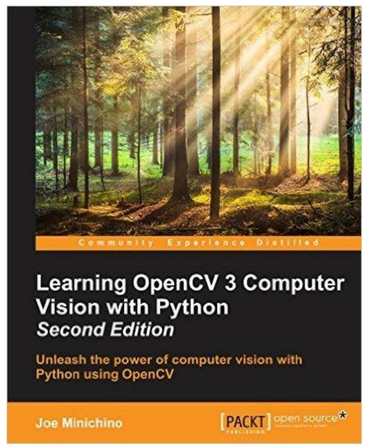


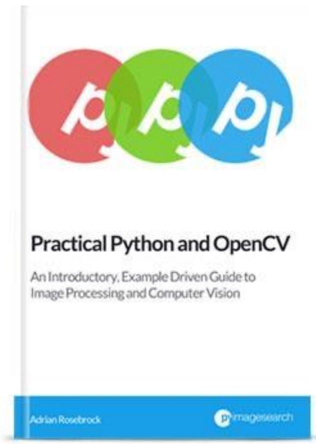


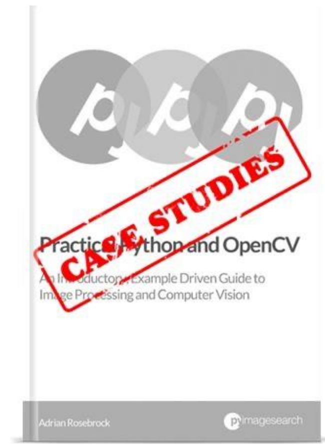
Bibliography

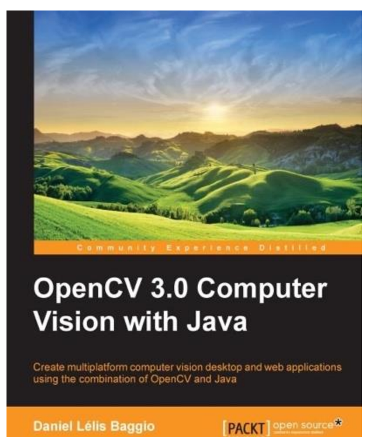
Python | Java | .NET APIs

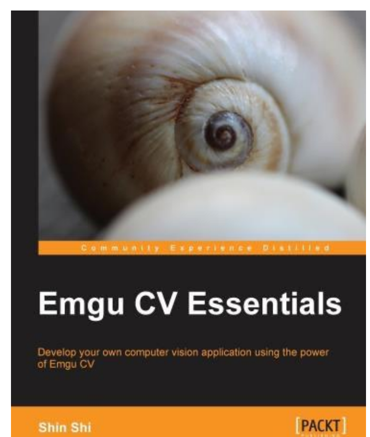








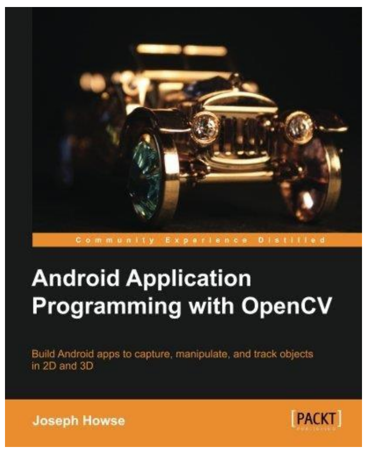


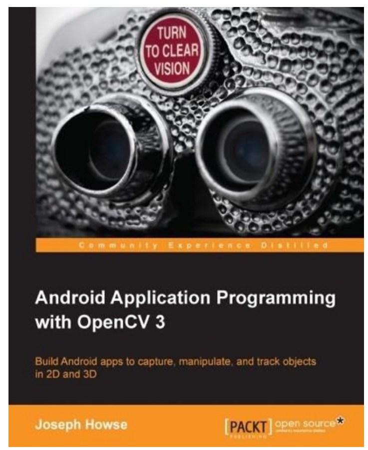


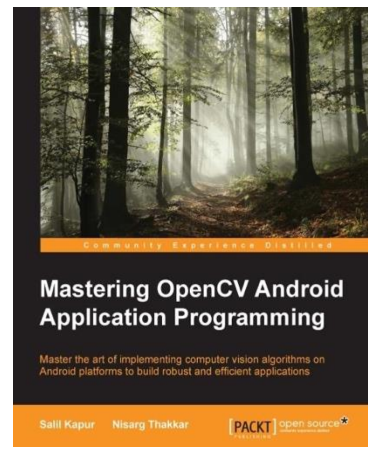
Bibliography

Mobile & Game APIs

Android | iOS| Raspberry Pi | Unity







OpenCV Modules

- **Calib3d**
  - Calibration, stereo, homography, rectify, projection, solvePNP
- **Contrib**
  - Octree, self-similar feature, sparse L-M, bundle adj, chamfer match
- **Core**
  - Data structures, access, matrix ops, basic image operations
- **features2D**
  - Feature detectors, descriptors and matchers in one architecture
- **Flann** (Fast library for approximate nearest neighbors)
- **Gpu** – CUDA speedups
- **Highgui**
  - Gui to read, write, draw, print and interact with images
- **Imgproc** – image processing functions
- **ML** – statistical machine learning, boosting, clustering
- **Objdetect** – PASCAL VOC latent SVM and data reading
- **Traincascade** – boosted rejection cascade

Digital Image Acquisition

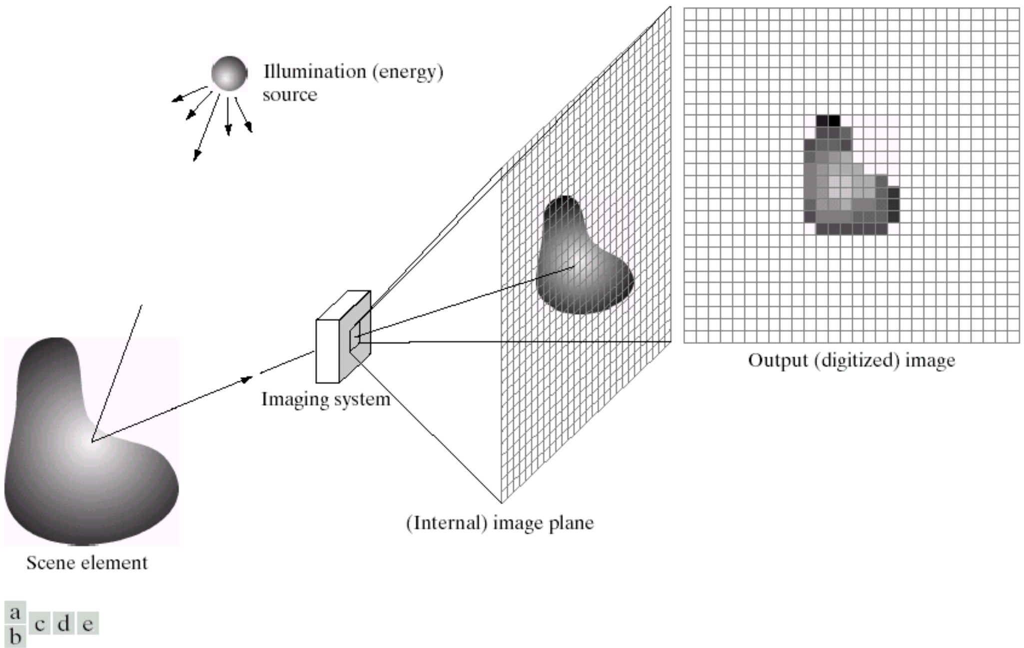


FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Digital Camera Issues

- Noise
  - caused by low light
- Color
  - color fringing (chromatic aberration) artifacts from Bayer patterns
- Blooming
  - charge overflowing into neighboring pixels
- In-camera processing
  - over-sharpening can produce halos
- Compression
  - creates blocking artefacts

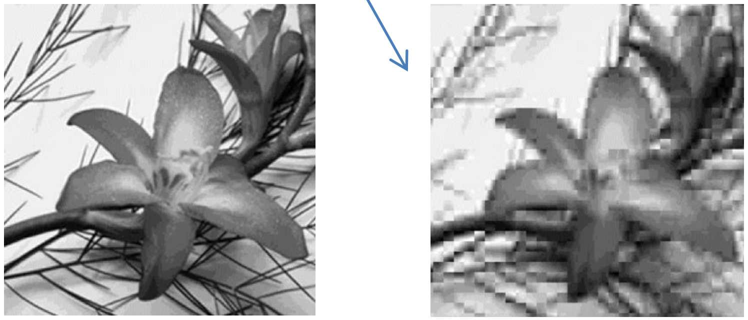
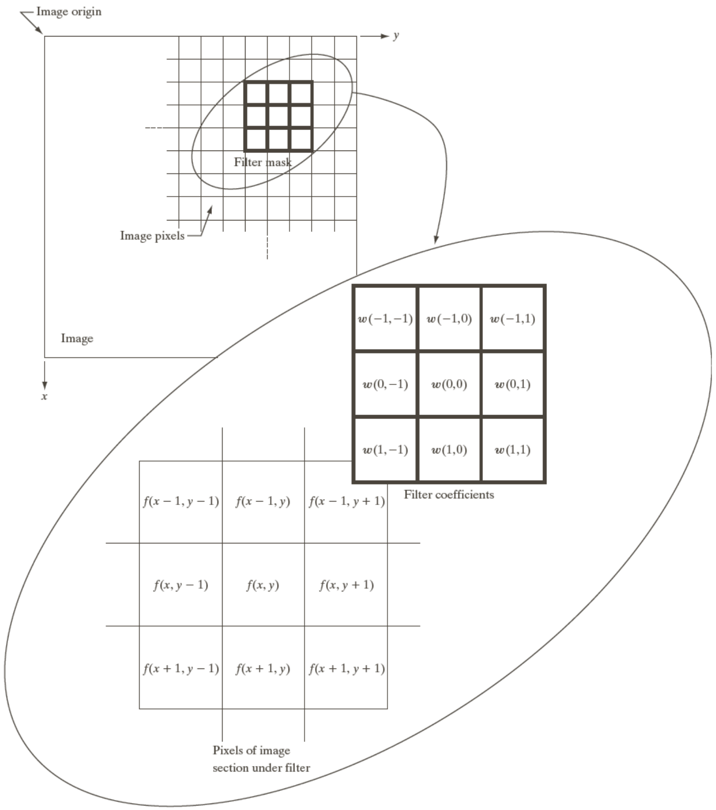


Image Filtering



Input Image

7	8	4	5	5
5	9	4	3	8
5	2	7	2	2
6	1	9	2	4
3	2	6	9	4

median

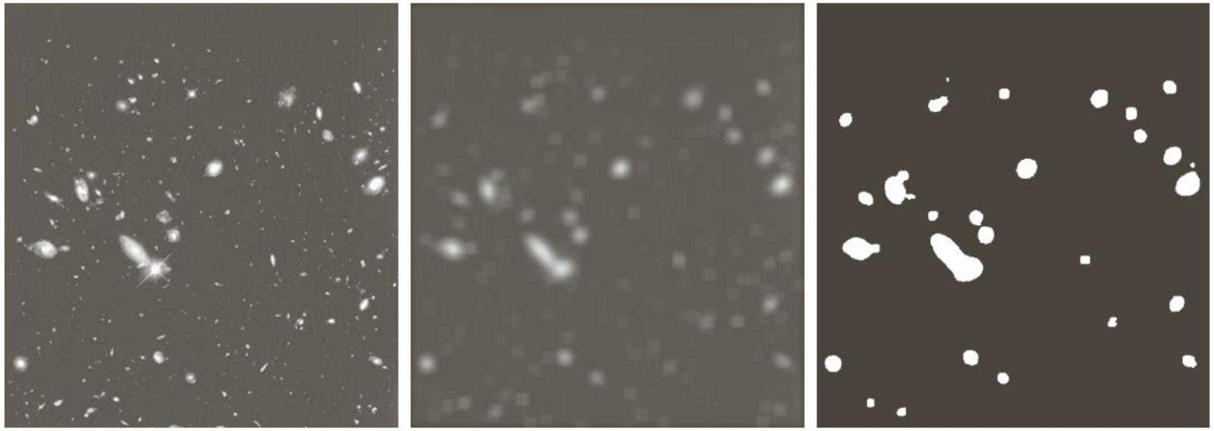
2 2 2 2 4 4 4 9 9

Filtered Img

8	5	4	4	5
7	5	4	4	3
5	5	3	4	2
2	5	2	4	2
2	6	2	4	4

FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

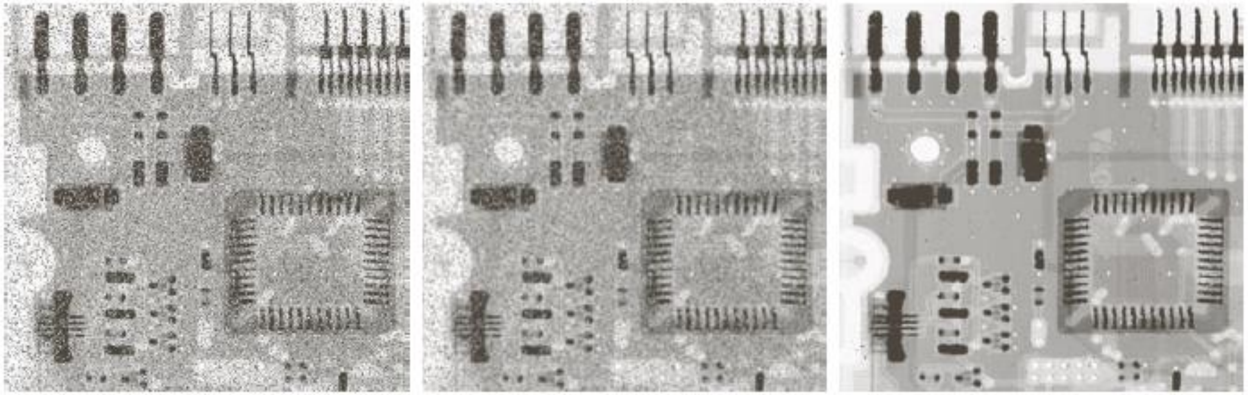
Application of Filtering



a b c

FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

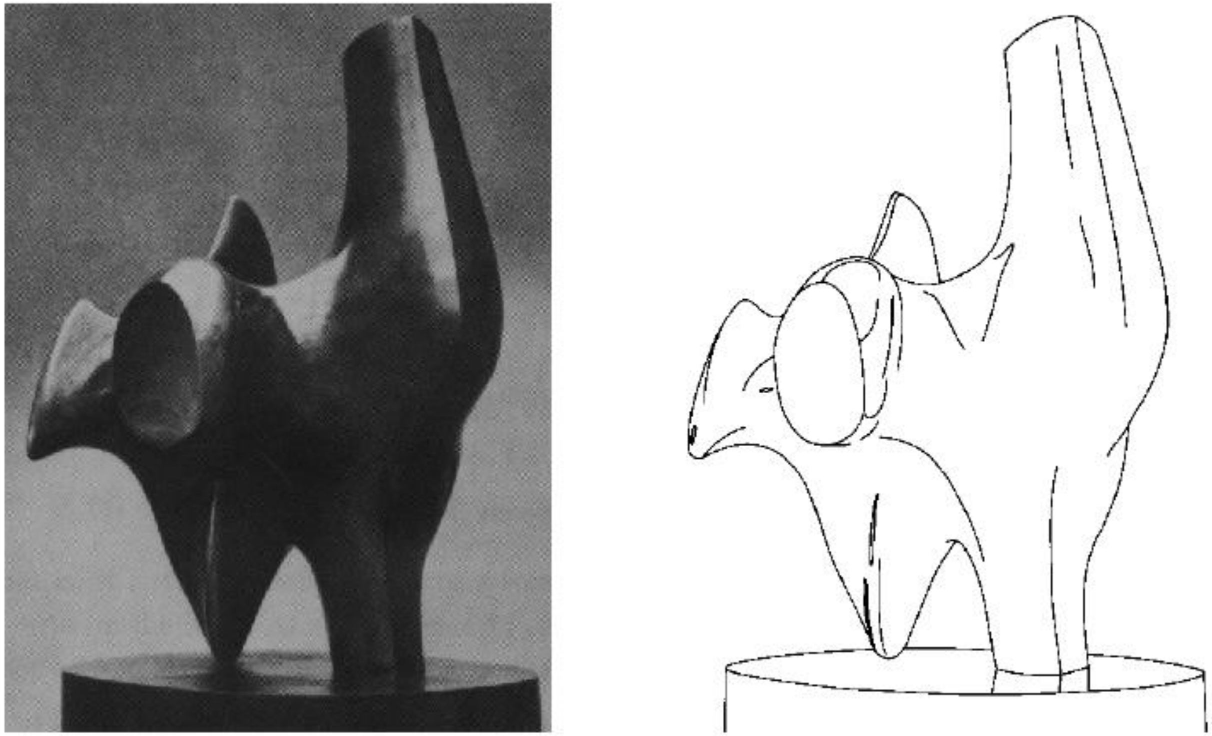
Filtering for Denoising



a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3 × 3 averaging mask. (c) Noise reduction with a 3 × 3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Edge detection



- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Optimal Edge Detection: Canny

- Assume:
  - Linear filtering
  - Additive Gaussian noise
- Edge detector should have:
  - Good Detection. Filter responds to edge, not noise.
  - Good Localization: detected edge near true edge.
  - Single Response: one per edge.
- Detection/Localization trade-off
  - More smoothing improves detection
  - And hurts localization.

The Canny edge detector

original image (Lena)

Norm of gradient

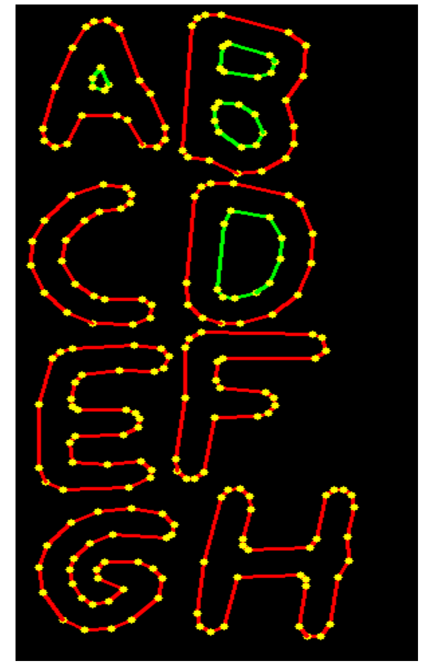
thresholding

thining

Contours Examples

Source Picture
(300x600 = 180000 pts total)

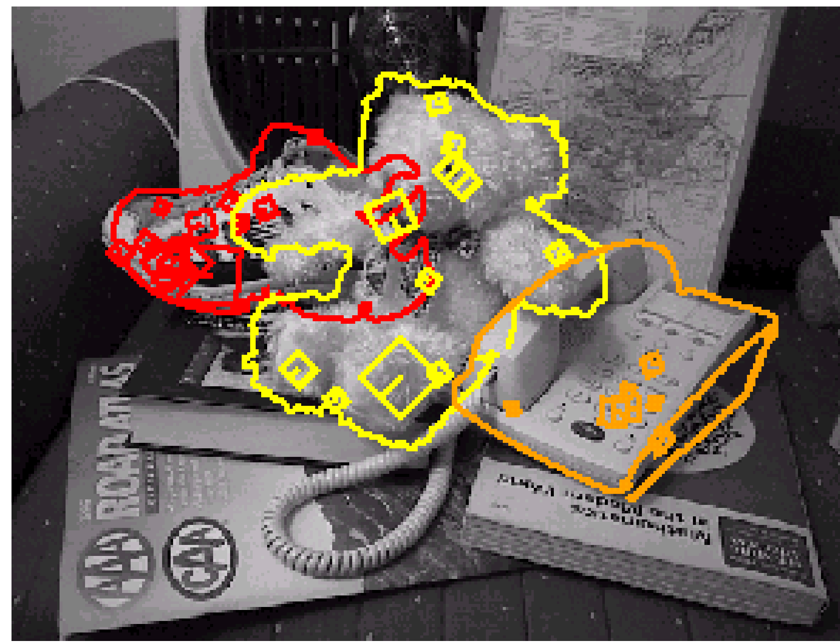
Retrieved Contours
(<1800 pts total)



After Approximation
(<180 pts total)

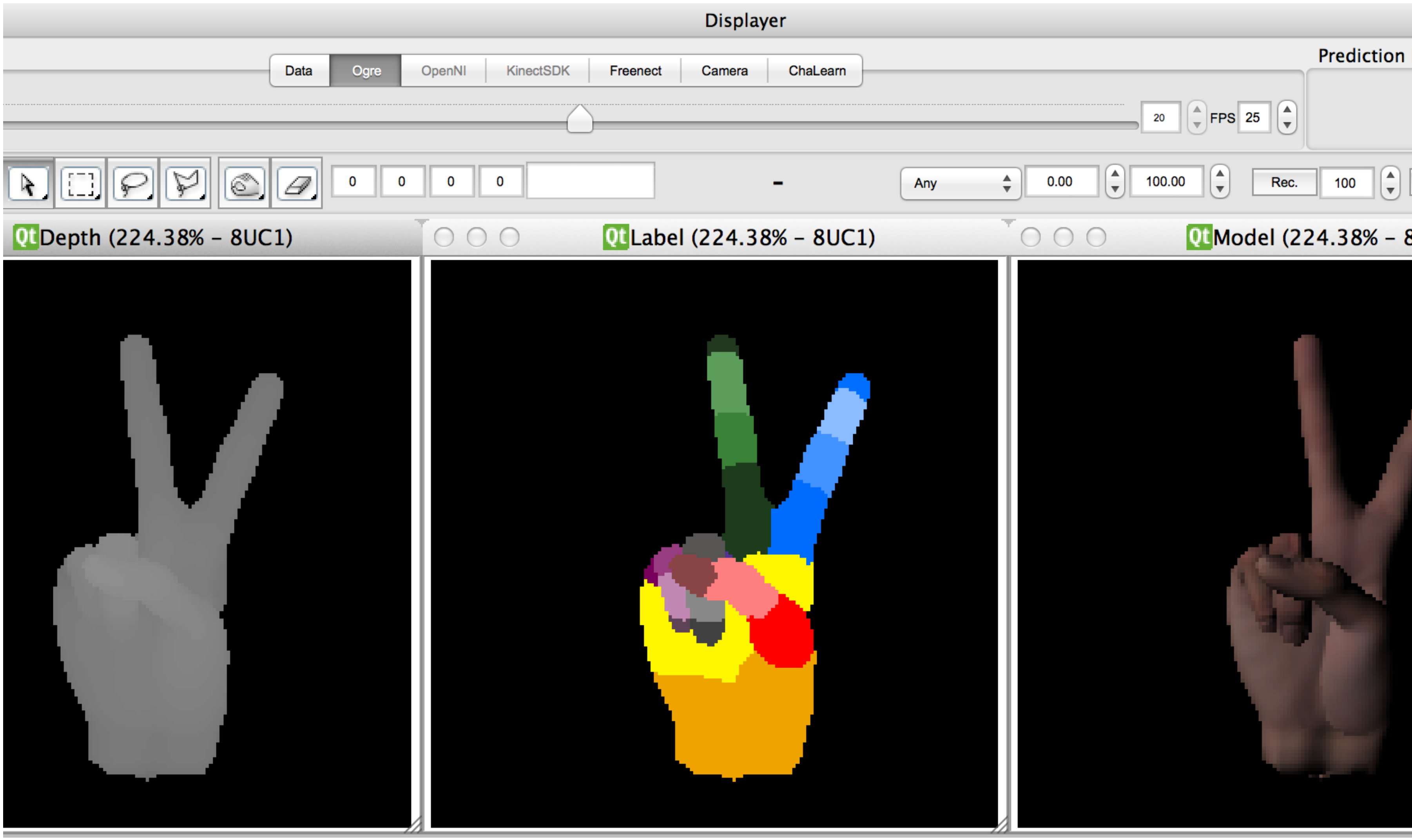
And it is rather fast: ~70 FPS for 640x480 on complex scenes

Object Recognition



Human Pose/Sign Estimation

Embedded ML

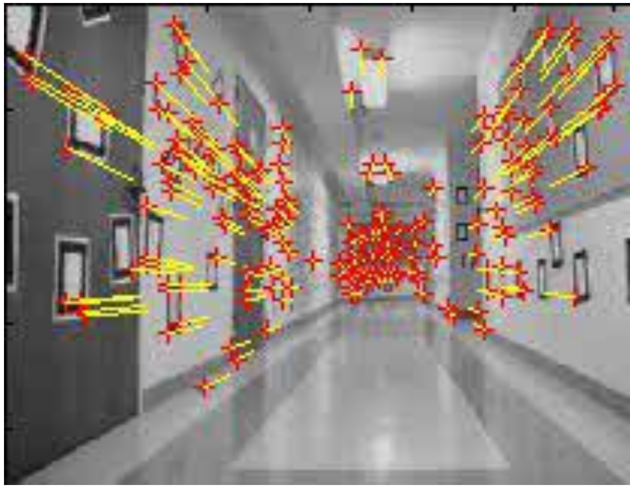


Motion Detection

- **A. Frame Differencing (Basic)**
  - *Logic:* Subtracts the current frame from the previous one (|Frame_t - Frame_{t-1}|).
  - *Pros/Cons:* Extremely fast, but sensitive to noise and fails if the object stops moving.
  - *Key Function:* cv2.absdiff(frame1, frame2)
- **B. Background Subtraction (Intermediate)**
  - *Logic:* The system "learns" a static background model over time and isolates moving foreground objects. Adapts to lighting changes.
  - *Pros/Cons:* Robust for surveillance; handles shadows.
  - *Key Function:* cv2.createBackgroundSubtractorMOG2()
- **C. Optical Flow (Advanced)**
  - *Logic:* Tracks the apparent motion of brightness patterns to calculate velocity and direction vectors for every pixel.
  - *Pros/Cons:* Provides speed and direction data, but is computationally expensive.
  - *Key Function:* cv2.calcOpticalFlowFarneback()

Optical Flow

```
// opencv/samples/c/lkdemo.c
int main(…){
…
CvCapture* capture = <…> ?
    cvCaptureFromCAM(camera_id) :
    cvCaptureFromFile(path);
if( !capture ) return -1;
for(;;) {
  IplImage* frame=cvQueryFrame(capture);
  if(!frame) break;
  // … copy and process image
cvCalcOpticalFlowPyrLK( …)
  cvShowImage( “LkDemo”, result );
```

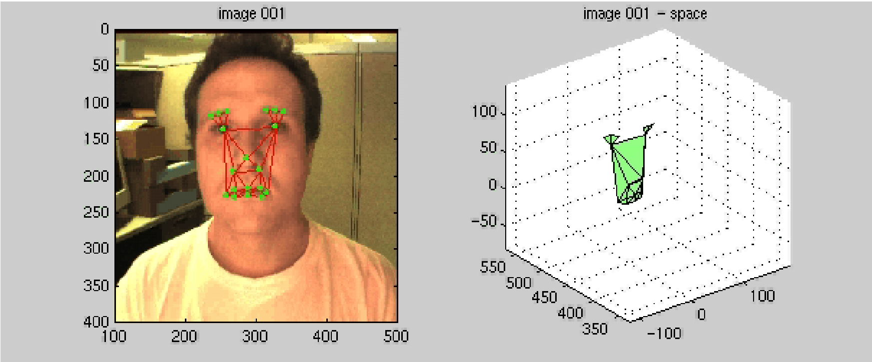


$$I(x+dx, y+dy, t+dt) = I(x, y, t);$$
$$-\partial I/\partial t = \partial I/\partial x \cdot (dx/dt) + \partial I/\partial y \cdot (dy/dt);$$

$$G \cdot \partial X = b,$$

$$\partial X = (\partial x, \partial y), G = \sum \begin{bmatrix} I_x^2, & I_x I_y \\ I_x I_y, & I_y^2 \end{bmatrix}, b = \sum I_t \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

calcOpticalFlowPyrLK()
Also see dense optical flow:
calcOpticalFlowFarneback()



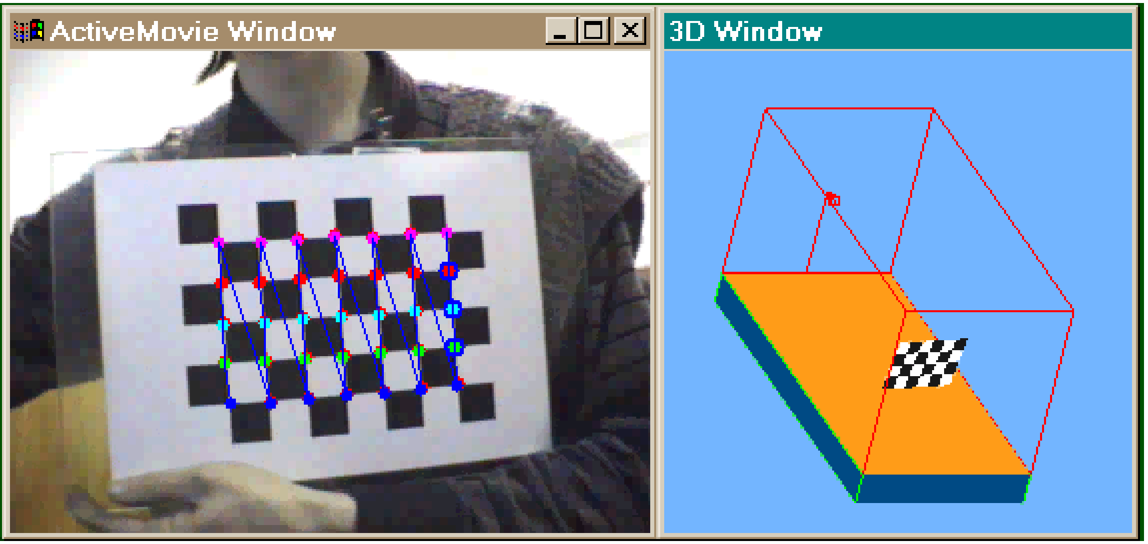
Single Camera Calibration

See samples/cpp/calibration.cpp

Now, camera calibration can be done by holding checkerboard in front of the camera for a few seconds.

And after that you'll get:

3D view of checkerboard



Un-distorted image

Machine Learning Library (MLL)

Embedded ML

CLASSIFICATION / REGRESSION

(new) Fast Approximate NN (FLANN)

(new) Extremely Random Trees

(coming) LSH

CART

Naïve Bayes

MLP (Back propagation)

Statistical Boosting, 4 flavors

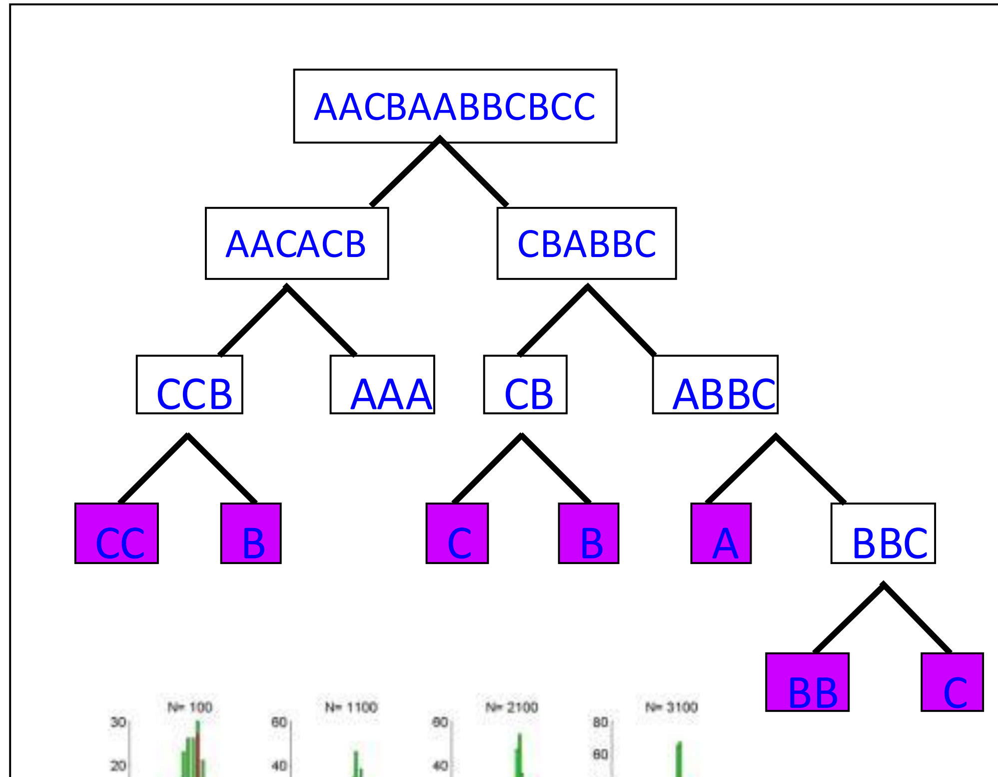
Random Forests

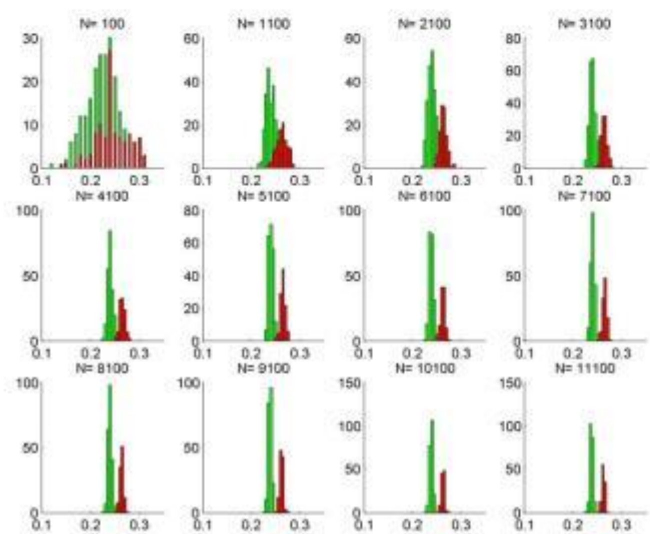
SVM

Face Detector

(Histogram matching)

(Correlation)



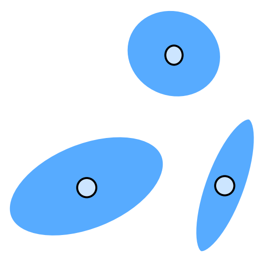


CLUSTERING

K-Means

EM

(Mahalanobis distance)



TUNING/VALIDATION

Cross validation

Bootstrapping

Variable importance

Sampling methods

http://opencv.willowgarage.com

YOLO

- **Definition:** YOLO is a state-of-the-art, real-time object detection system.
- **The Core Concept:** Unlike traditional classifiers that scan images step-by-step, YOLO treats object detection as a single regression problem.
- "**You Only Look Once**"**:** The model looks at the entire image exactly once to predict:
- What objects are present (Classification).
- Where they are located (Localization/Bounding Boxes).
- **Key Characteristic:** Extremely fast inference speed, making it the standard for video processing and live streams.

Evolution of YOLO

- **YOLOv1 (2015):** The breakthrough. Fast but struggled with small objects.
- **YOLOv2 & v3:** Introduced "Anchor Boxes" and multi-scale detection (detecting small & big objects simultaneously).
- **YOLOv4 & v5:** Focused on speed optimizations (Mosaic augmentation) and usability (PyTorch implementation).
- **YOLOv8 (2023) & Beyond:** The current standard. Introduced "Anchor-Free" detection, simpler architecture, and native support for segmentation and pose estimation.
- **YOLO-NAS / YOLO-World:** New variants using Neural Architecture Search and open-vocabulary detection.
-

Pros & Cons of YOLO

- **Advantages:**
- **Speed:** Capable of 140+ FPS (frames per second) on modern GPUs.
- **Global Context:** Because it sees the whole image at once, it makes fewer "background errors" compared to sliding-window models.
- **Versatility:** Can perform Detection, Segmentation (masks), and Pose Estimation (skeletons) with the same API.
- **Limitations:**
- Can struggle with very small objects grouped tightly together (e.g., a flock of birds) compared to slower, two-stage detectors like Faster R-CNN.

YOLO Real-World Applications

- **Autonomous Driving:** Detecting pedestrians, traffic lights, and other cars in milliseconds.
- **Security & Surveillance:** Intruder detection, face recognition, and crowd counting.
- **Retail:** Automated checkout (identifying products) and people counting in stores.
- **Healthcare:** Detecting tumors in X-rays or counting cells in microscopy.
- **Agriculture:** Weed detection and crop monitoring from drones.