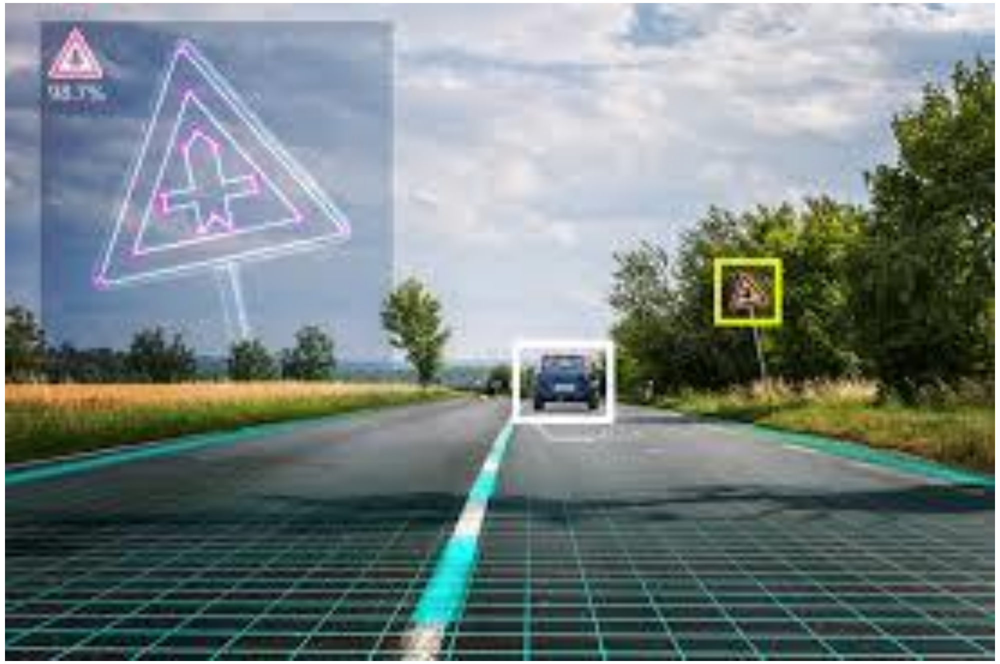
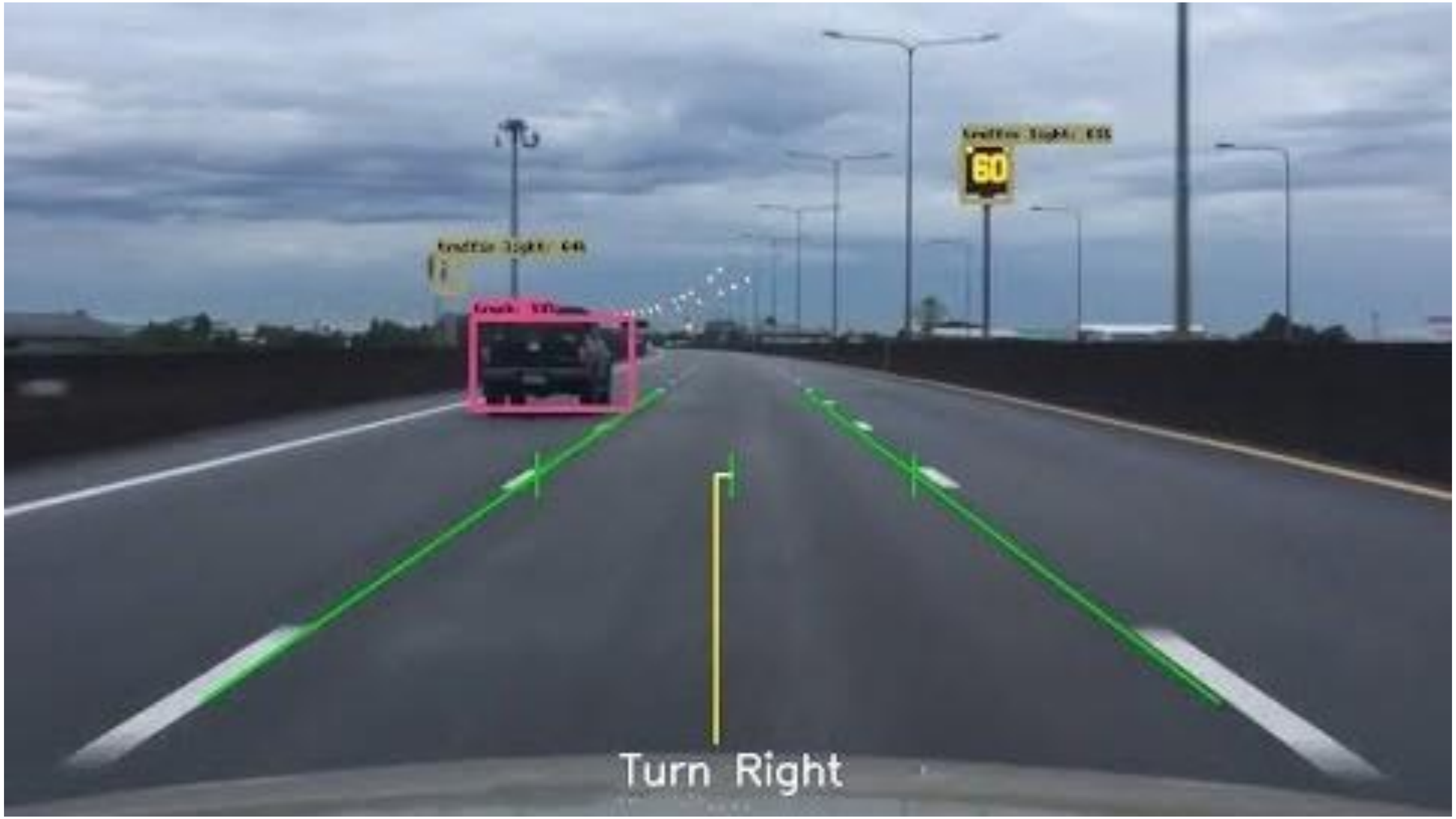


COMP3431 Robot Software Architectures

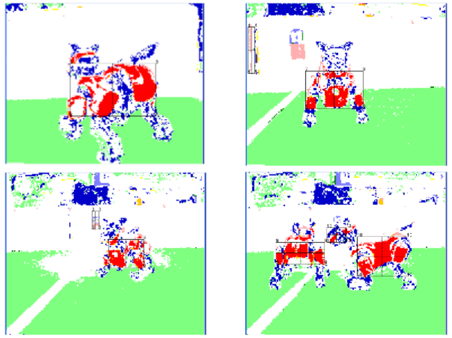
Part 1 - Elements for Robotics Vision

Autonomous Driving



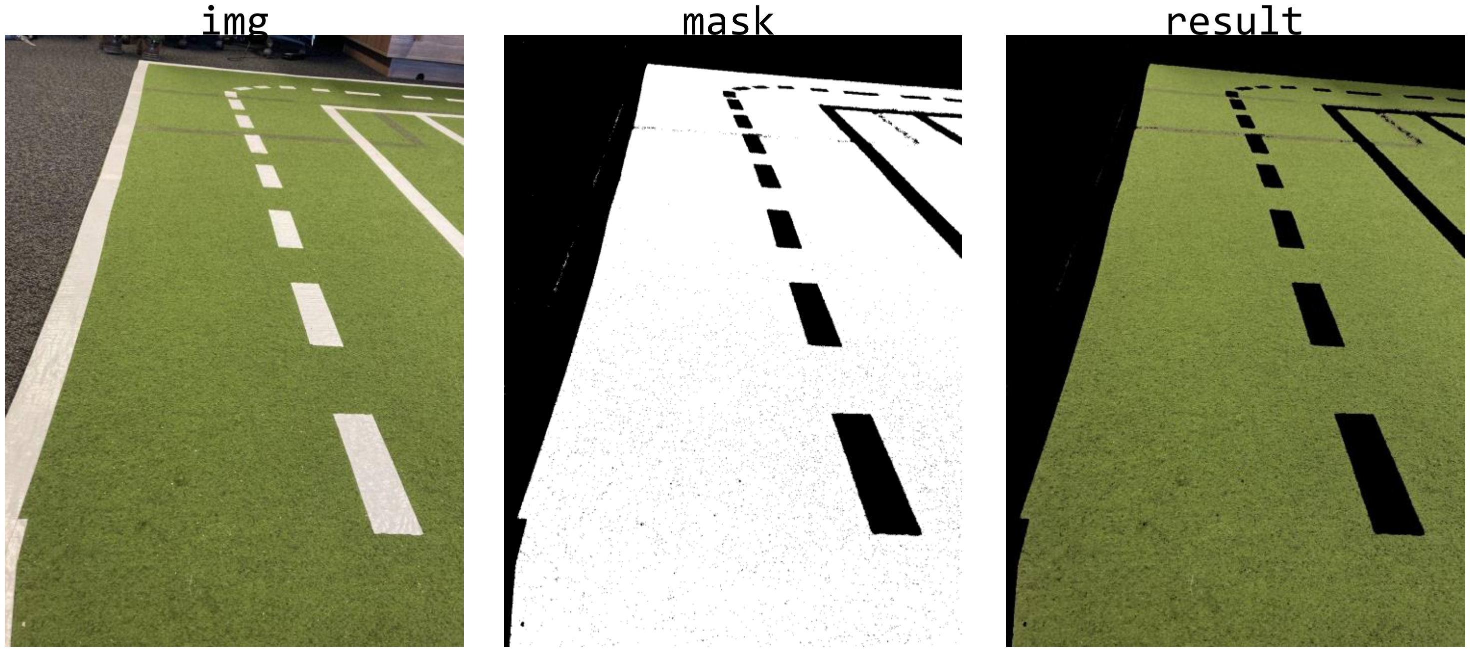
What you've seen so far in robotics vision

- Blob detection / Color Thresholding



Slide 28 – Week 5

```
img = cv2.imread('road_img.jpg')
imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(imgHSV, (20, 80, 70), (50, 255, 255))
result = cv2.bitwise_and(img, img, mask=mask)
```

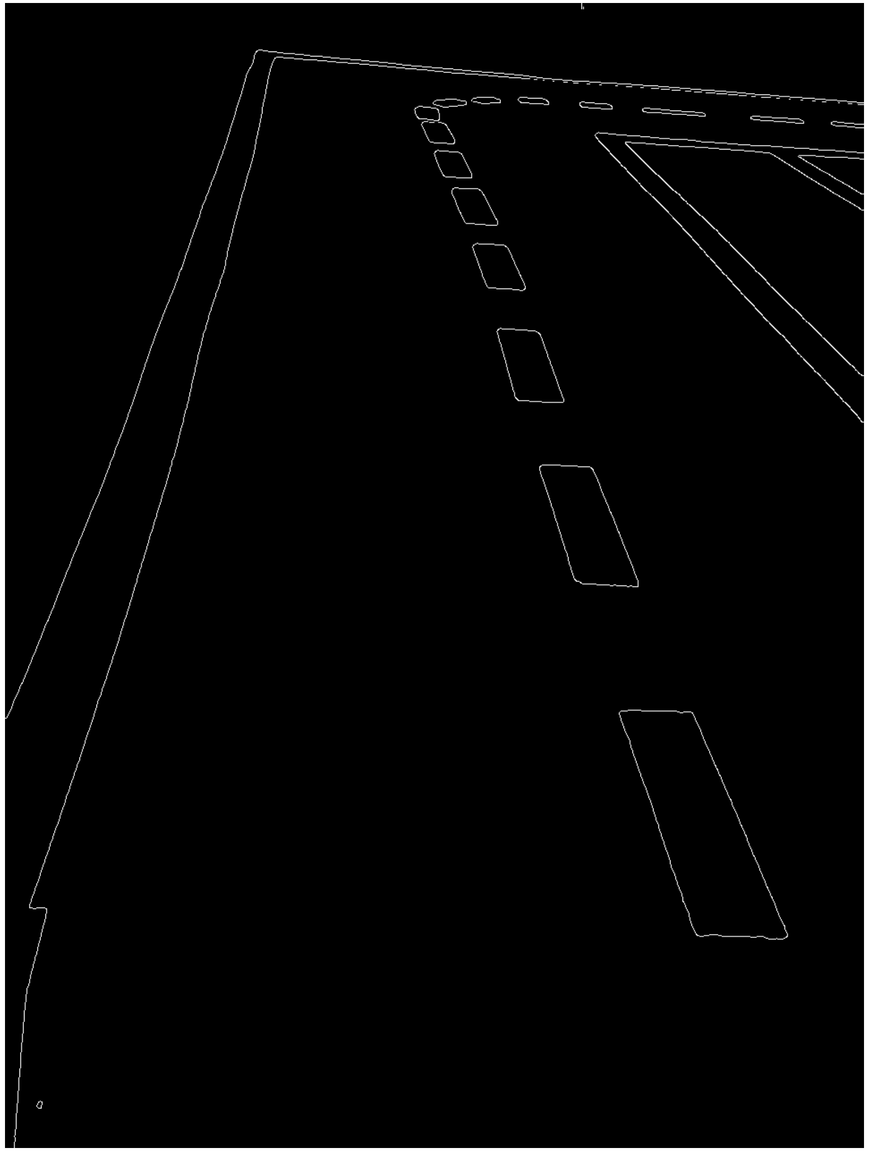


What you've seen so far in robotics vision

- Edge Detection

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.blur(gray,(5,5))
_, th_img = cv2.threshold(blur,160,255,cv2.THRESH_BINARY)
edges = cv2.Canny(th_img,100,200)
```



Lines Detection

- Least Square
- RANSAC
  - Voting system, using inliners
  - Each potential line gets voted on by each data point, best wins
  - Might-endup with very similar lines
  - Need post-processing

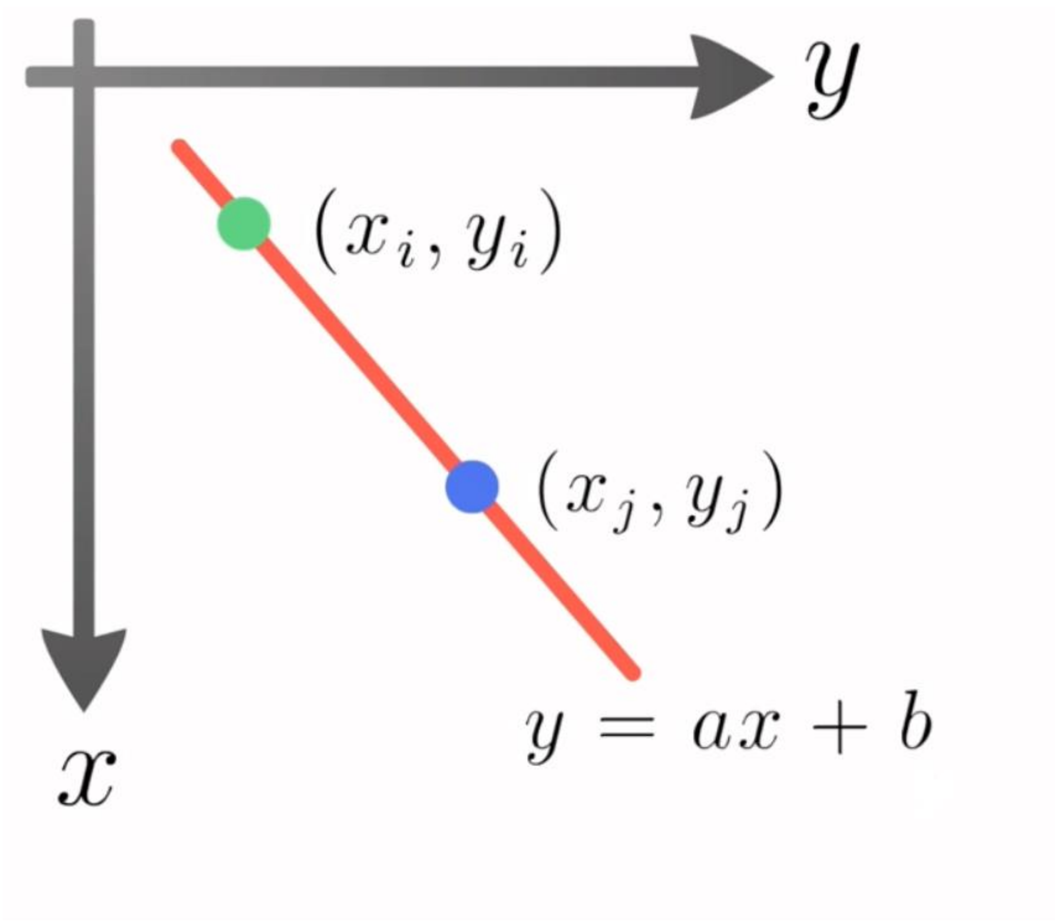
- Other voting methods

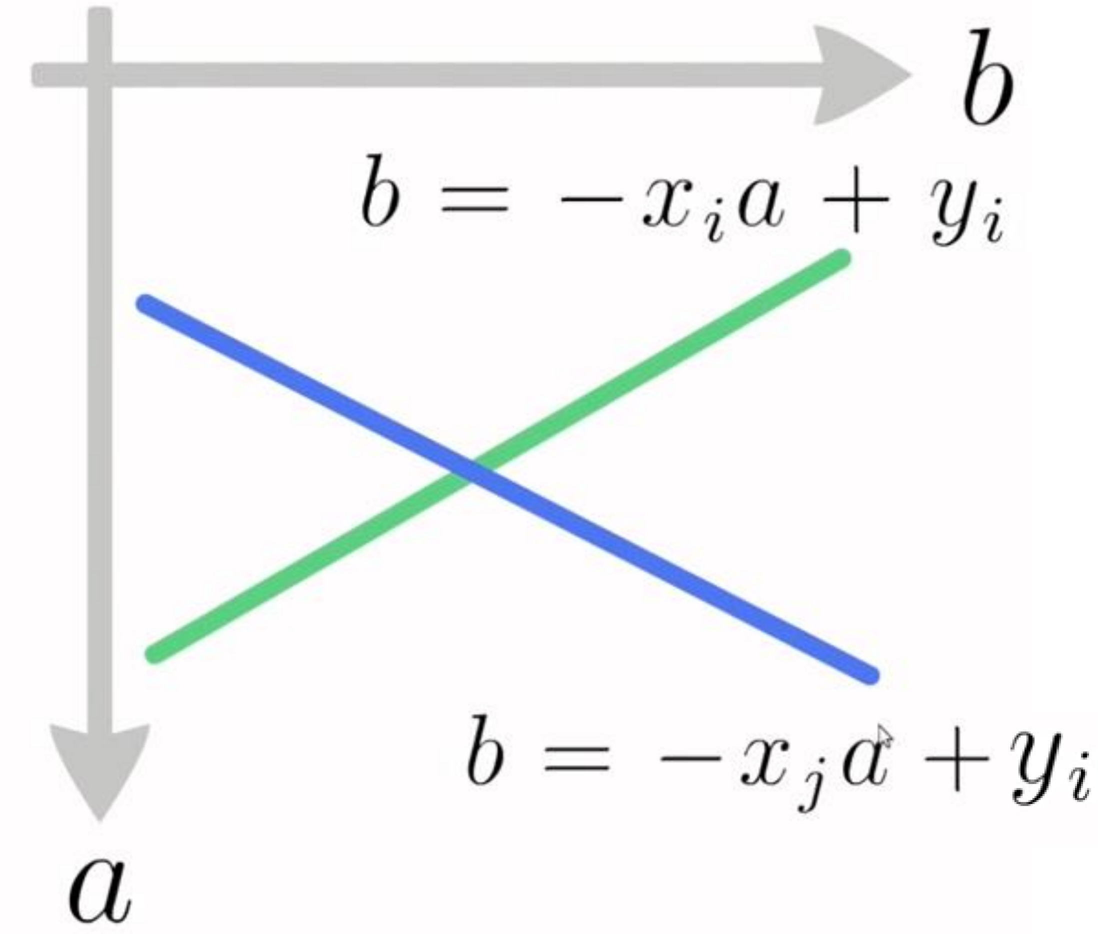
Hough Transform

- Algorithm used to find straight lines or any geometrically parametrized shapes (ellipses)

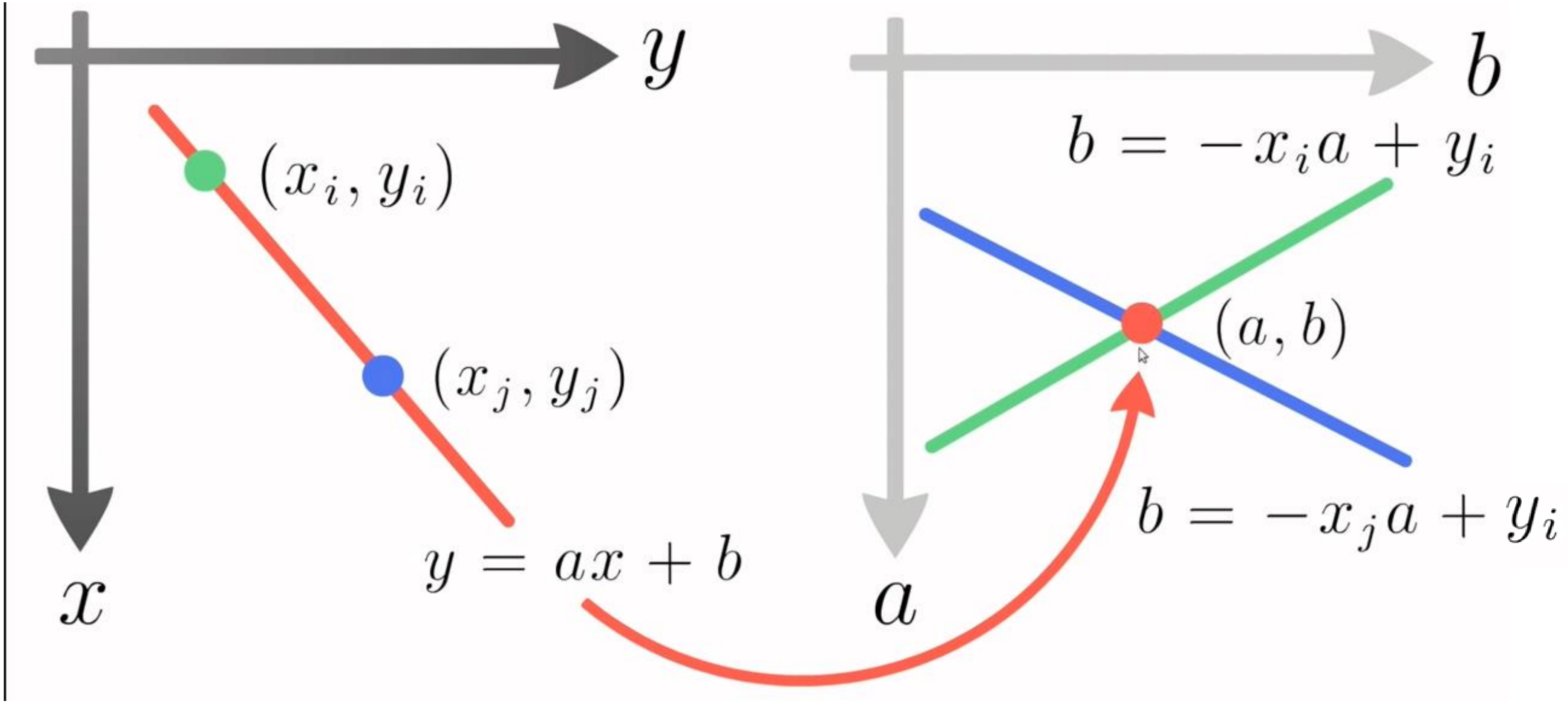
- It uses a voting procedure to find the most likely parameters of the shape to be detected

Hough Transform - Algorithm





Hough Transform - Algorithm



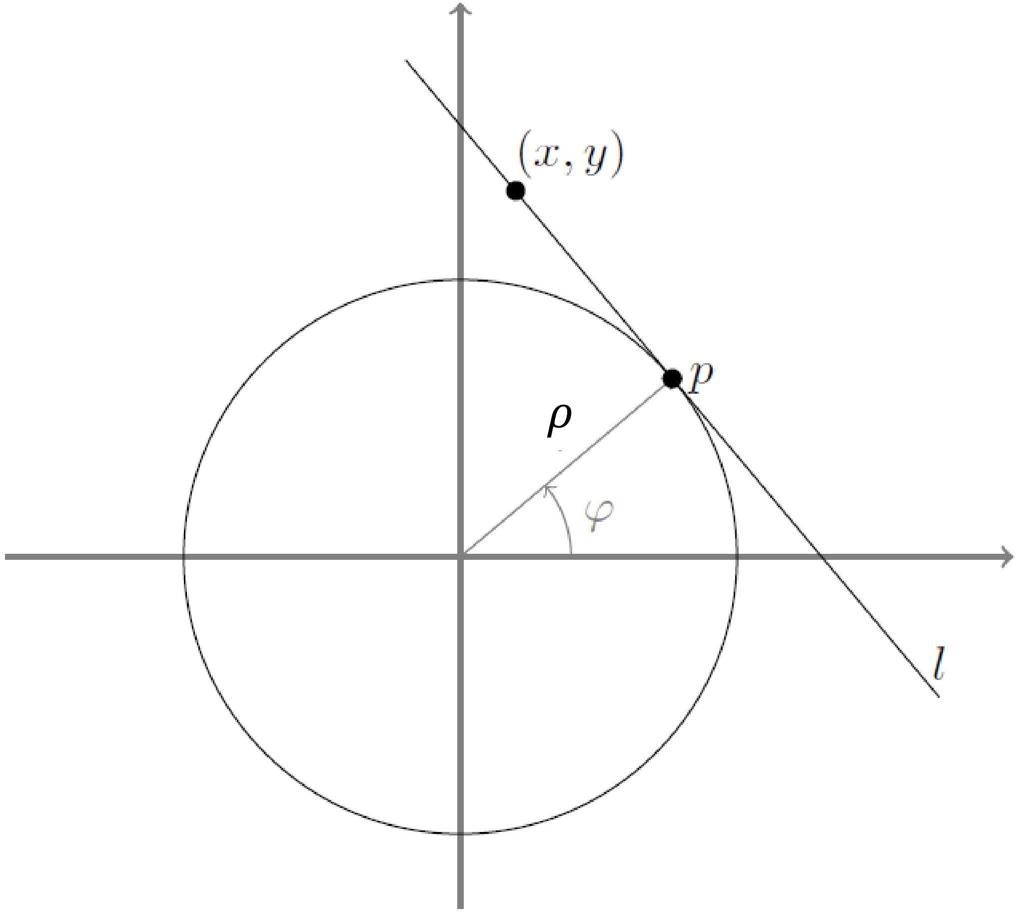
Hough Transform

- In polar coordinates, a line l is represented by (ρ, θ) such that l is the tangent to the circle of radius ρ at a point p forming an angle θ with the x-axis

- Besides, the set of lines passing through the point (x, y) can be described in polar coordiantes by:

$$\rho = x\cos(\theta) + y\sin(\theta)$$

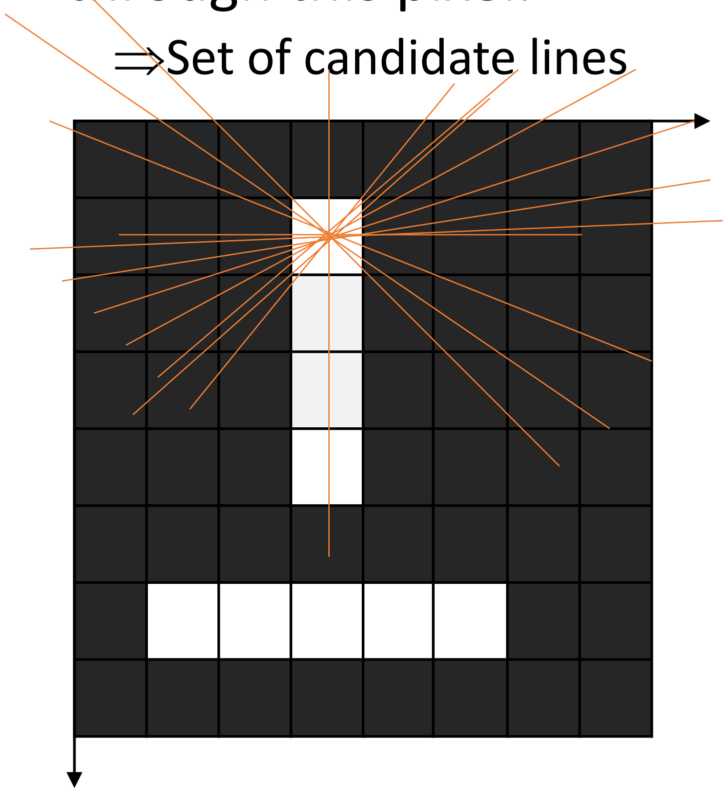
θ , for $\theta \in [0, \pi[$



Hough Transform - Algorithm

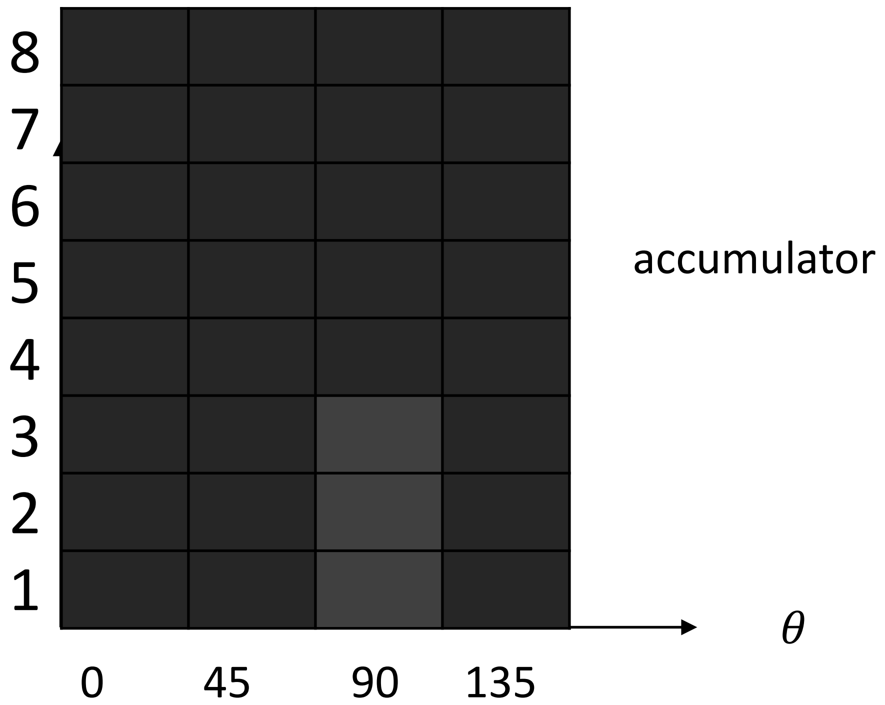
- For each pixel that could belong to a line (i.e. pixel that belong to the edges), compute the whole list of lines that pass through this pixel.

  $\Rightarrow$Set of candidate lines



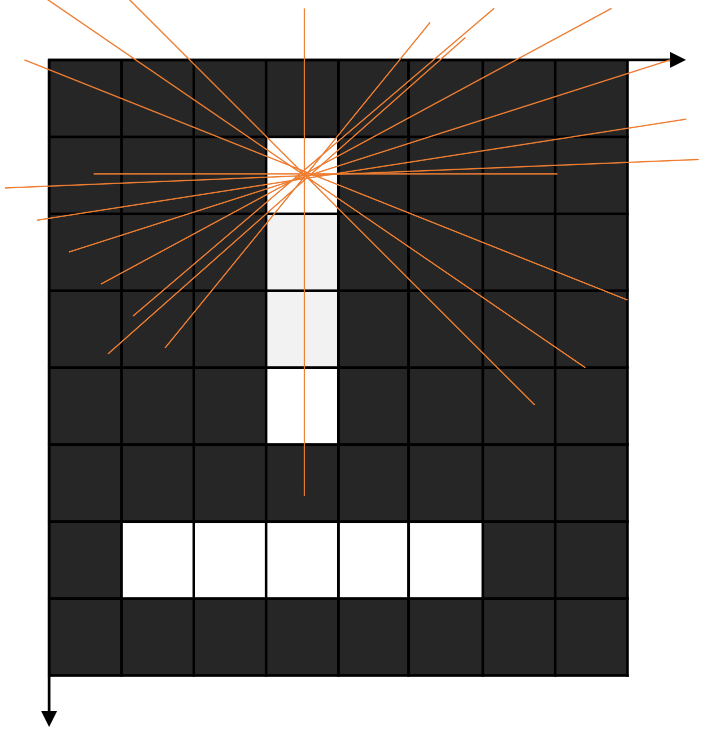
Discretisation step for θ: 45°

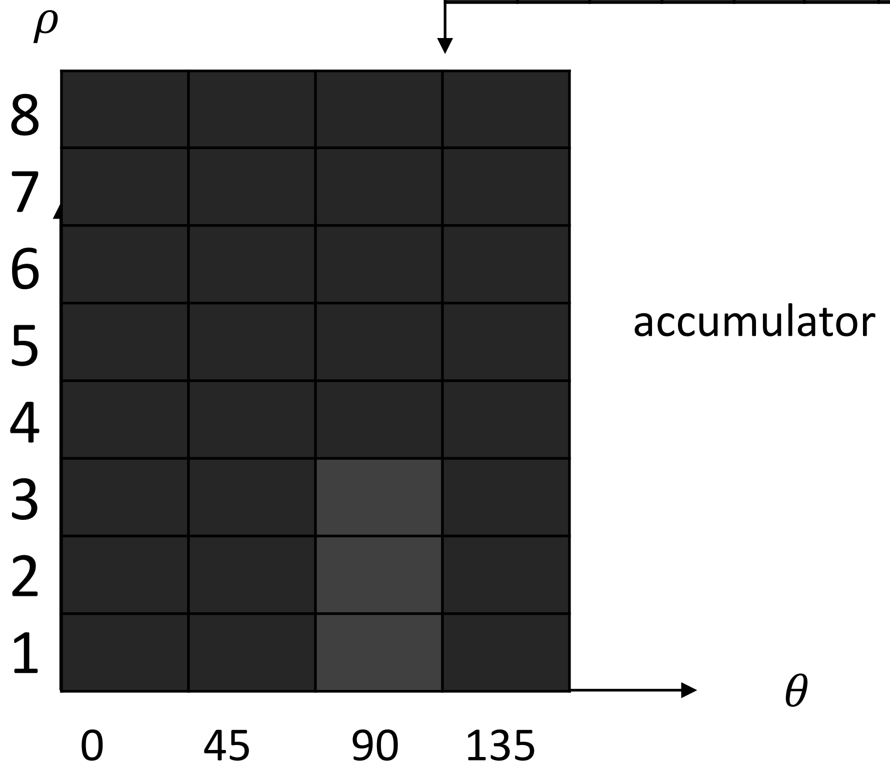
Discretisation step for ρ: 1



Hough Transform - Algorithm

- For each pixel that could belong to a line (i.e. pixel that belong to the edges), compute the whole list of lines that pass through this pixel.
  - ⇒Set of candidate lines
  - ⇒Update the accumulator with the voiting count





Hough Transform

- Maximas are representing the lines

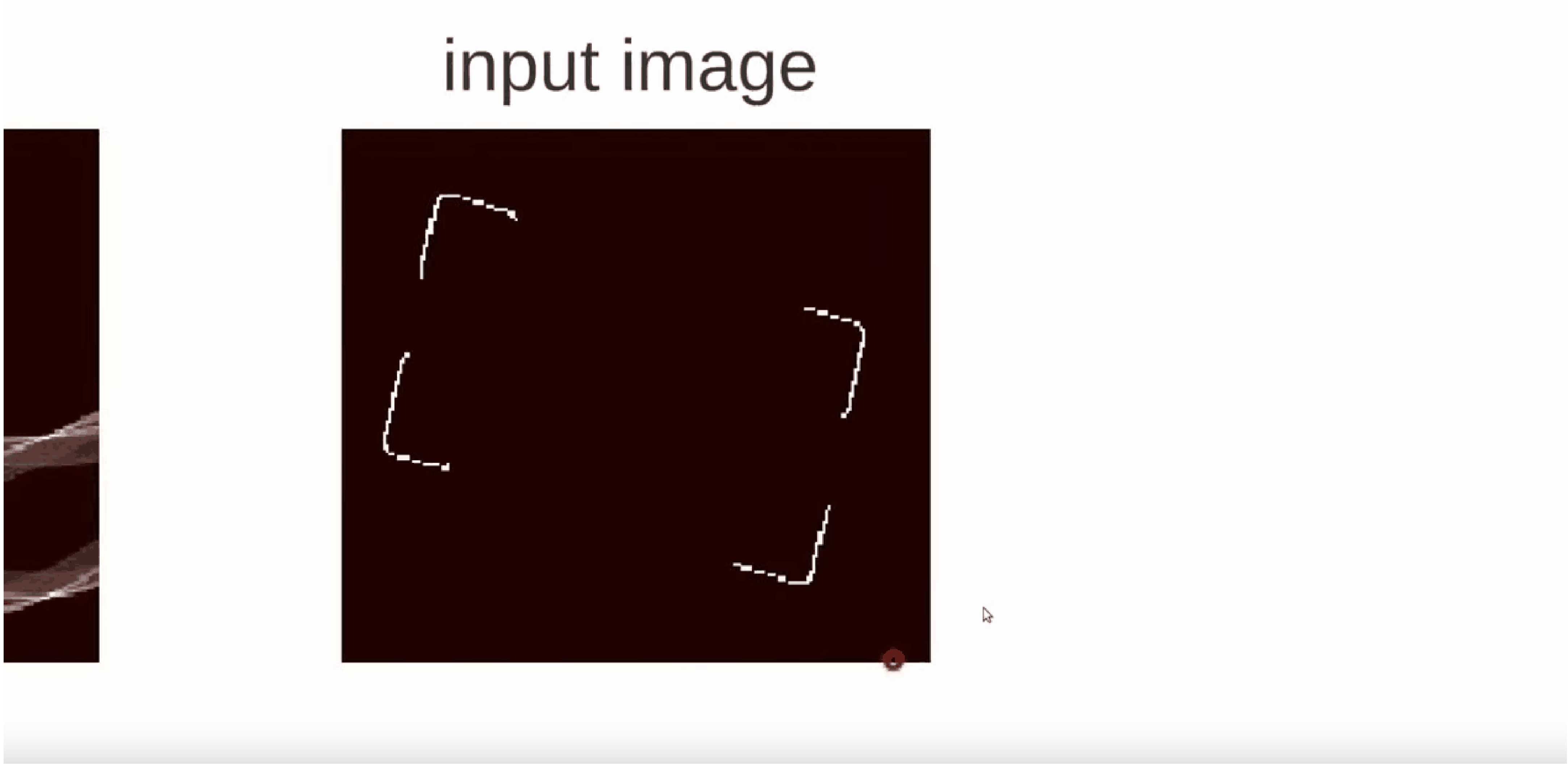
- Threshold per number of votes
- Select only lines that are sufficiently appart (between line gap)

Hough Transform

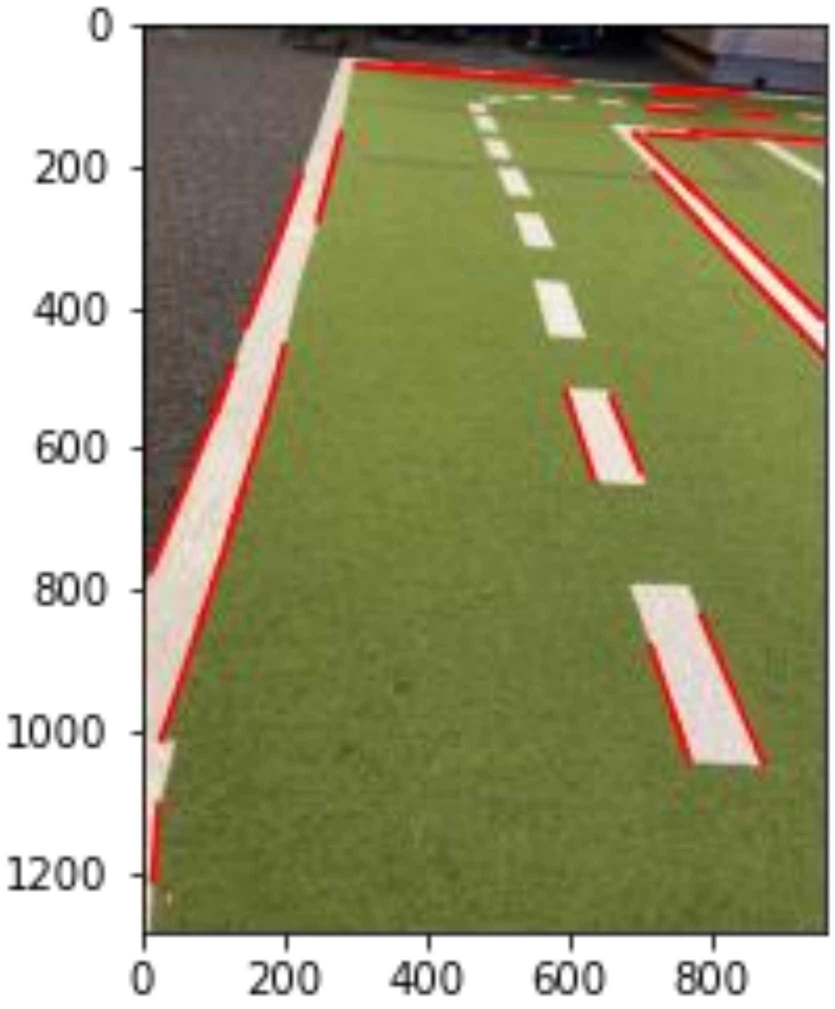
- Maximas are representing the lines

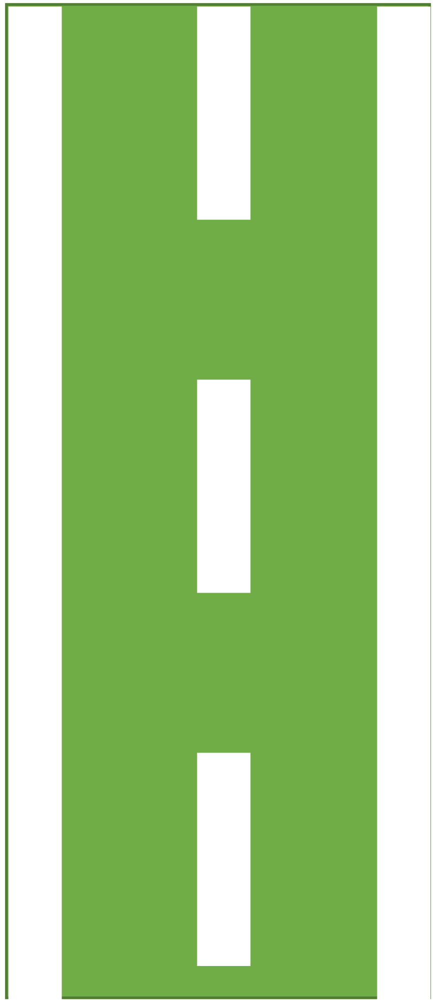
- Threshold per number of votes
- Select only lines that are sufficiently appart (between line gap)

Hough Transform



Go Further





Compute intersections or use line equations to find the homography.

Homography

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K \cdot [R|t] \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

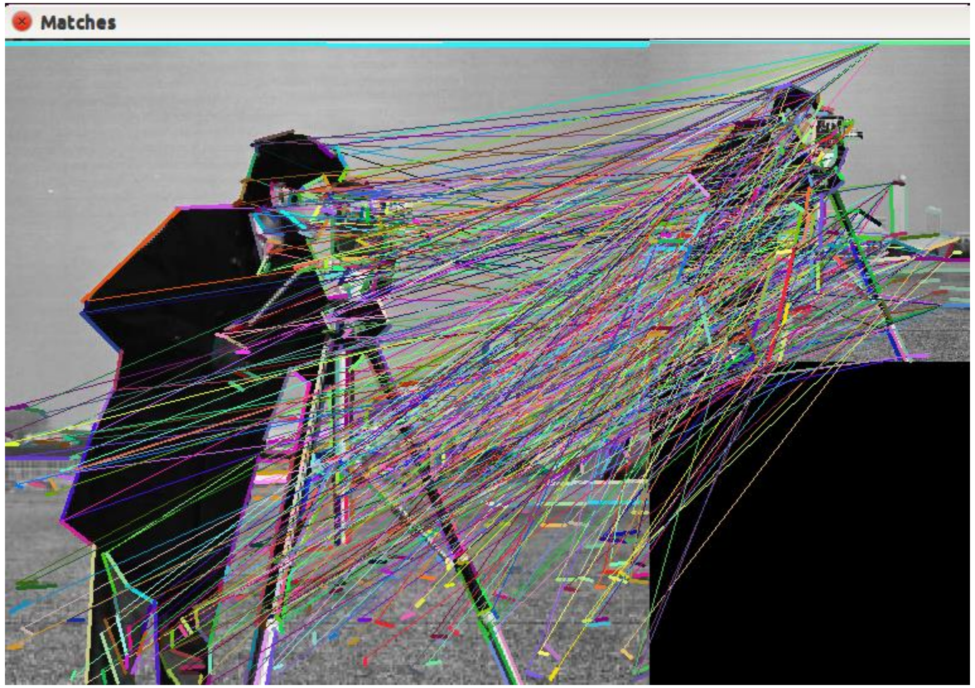
H: Homography
K: Intrinsic Parameters
[R|t]: Extrinsic Parameters

Descriptors Matching
Transformation estimation

Calibration

Other Lines Descriptors

- LSD extractor
- Compute lines and descriptors



Tracking

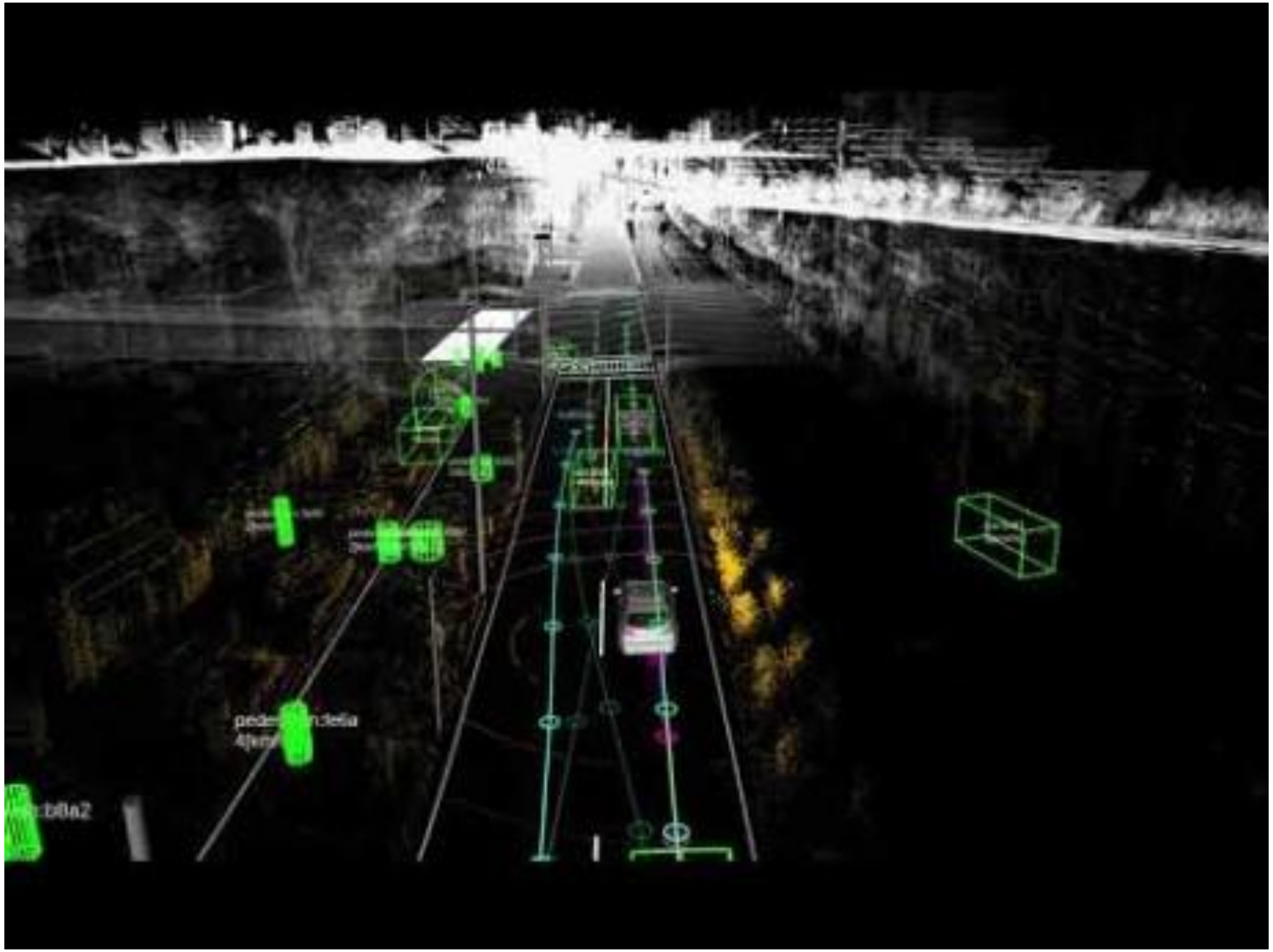
- Object in motion
- Robot in motion

- Object Detection
  - => Frame by frame

- Motion estimation

- Reduce load from frame to frame detection

Tracking – Autonomous Vehicle



Tracking

- Step 1:
  - Detecting
  - Feature extraction and id naming
- Step 2:
  - Matching / Retrieval
    - Data association
    - Similarity measurement
    - Correlation

Reasoning with strong priors

CONNECT
THE DOTS

Tracking – Problem Statement

- Input: Target
- Objective: Estimate the target state over time
- State:
  - **Position**
  - Appearance
  - Shape
  - Velocity
  - Affine transformation w.r.t previous patch

→

1/ Object representation
2 /Similarity measure
3/ Searching process

Tracking - Challenges

- Variations due to geometric changes (pose, articulation, scale)
- Variations due to photometric factors (illumination, appearance)
- Occlusions
- Non-linear motion
- Very limited resolution, blurry (standard recognition might fail)
- Similar objects in the scene

Tracking - Challenges

- Variations due to geometric changes (pose, articulation, scale)
- Variations due to photometric factors (illumination, appearance)
- Occlusions
- Non-linear motion
- Very limited resolution, blurry (standard recognition might fail)
- Similar objects in the scene

Tracking – Object Representation

- Goal:
  - we want a representation that is:
    - –Descriptive enough to disambiguate target VS background
    - –Flexible enough to cope with:
      - •Scale
      - •Pose
      - •Illumination
      - •Partial occlusions

Tracking – Object Representation

- Object approximation:
  - –Segmentation / Polygonal approximation
  - –Bounding ellipse/box
  - –Position only

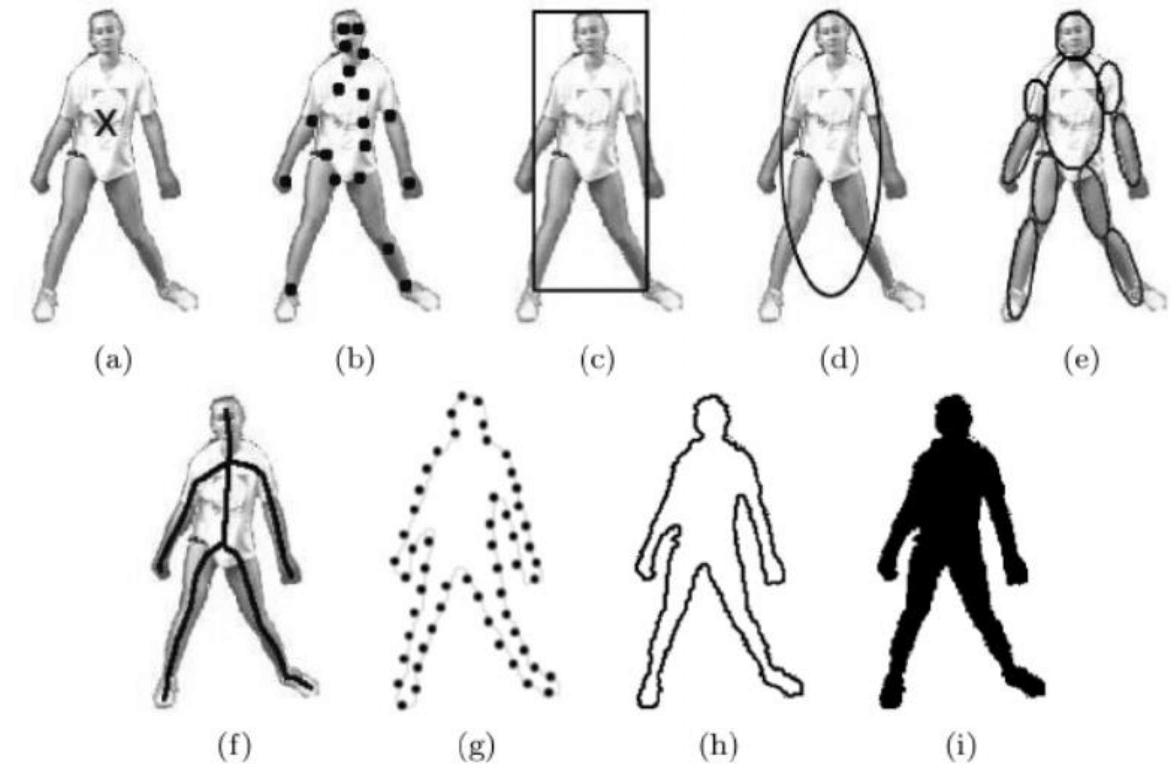


Fig. 1. Object representations. (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) complete object contour, (h) control points on object contour, (i) object silhouette.

Yilmaz et al. Object tracking: A survey. ACM Computing Surveys, 2006

Tracking – Affinity Measuring

- General

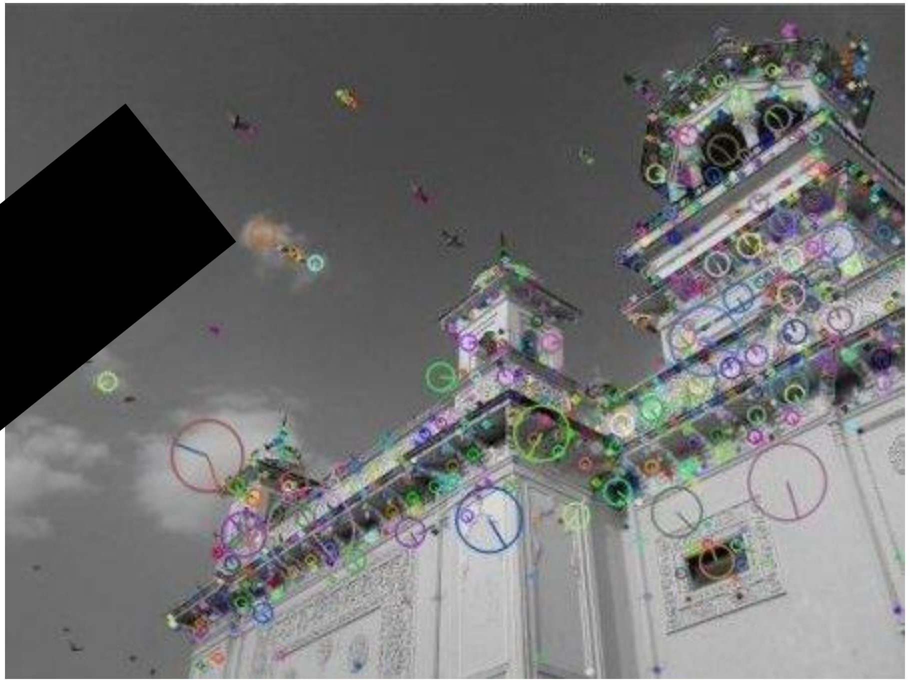
$$aff(x, y) = \exp\left(-\frac{1}{2\sigma_d^2}\|f(x) - f(y)\|^2\right)$$

- Example:
  - Distance
  - Intensity
  - Color
  - Region

Tracking – Object Representation with High Level Features

- SIFT
- BoW
- SURF
- Haar
- BRIEF/ORB
- FREAK

CNN

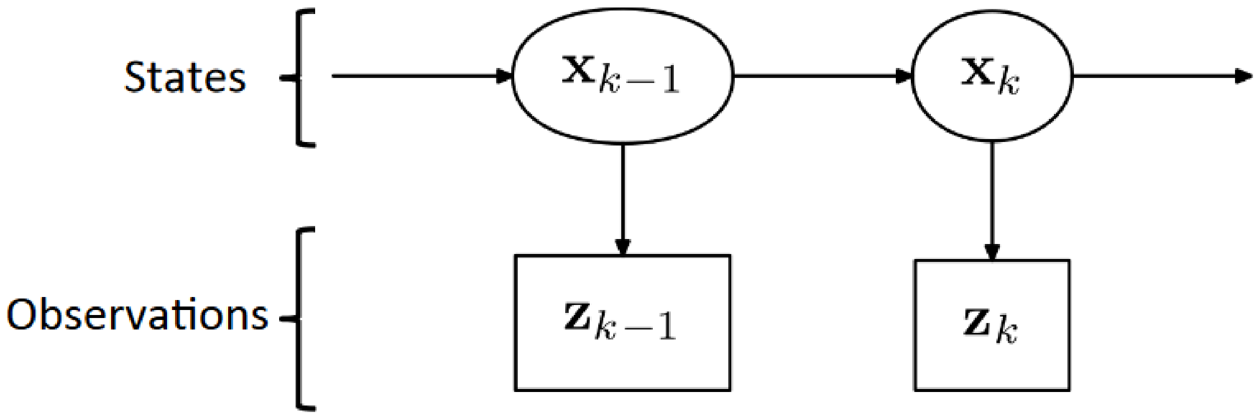


Tracking – Single Target

- Input: Bounding box at starting frame
- Output: next bounding boxes across the next frames

Tracking as Probabilistic Problem

- Hidden Markov Model



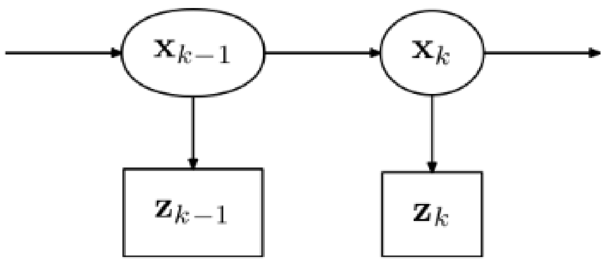
- Markov assumptions

$$p(x_k \mid x_{1:k-1}) = p(x_k \mid x_{k-1})$$

$$p(z_k \mid x_{1:k}) = p(z_k \mid x_k)$$

Tracking as Probabilistic Problem

- Recursive Bayes filters
- Find posterior $p(x_k \mid z_{1:k})$
- State eq. (motion dynamics) $f(x_k \mid x_{k-1})$
- Observation eq. (image) $g(z_k \mid x_k)$



- Prediction

Previous posterior

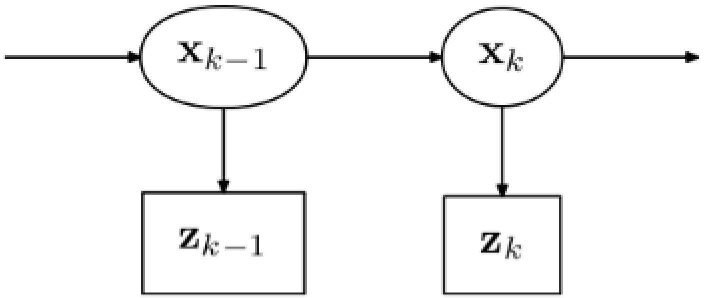
$$p(x_k \mid z_{1:k-1}) = \int f(x_k \mid x_{k-1}) \boxed{p(x_{k-1} \mid z_{1:k-1})} dx_{k-1}$$

- Update

$$p(x_k \mid z_{1:k}) = \frac{g(z_k \mid x_k) p(x_k \mid z_{1:k-1})}{\int g(z_k \mid x_k) p(x_k \mid z_{1:k-1}) dx_k}$$

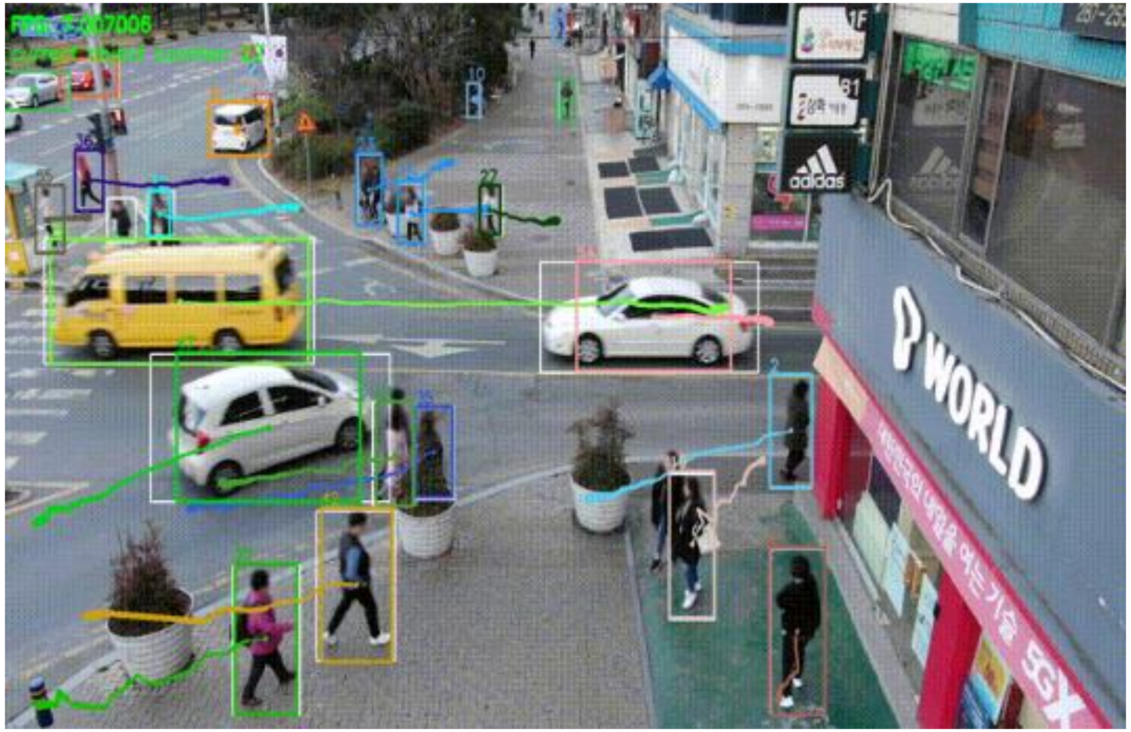
Tracking as Probabilistic Problem

- Solving Bayes Equations
  - Gaussian & Linear
    - Kalman filter [1]
  - Gaussian non-linear
    - Extended Kalman filter
  - Non-Gaussian non-linear
    - Monte Carlo methods (Condensation [2])
  - Hill-climbing on posterior
    - Mean-shift

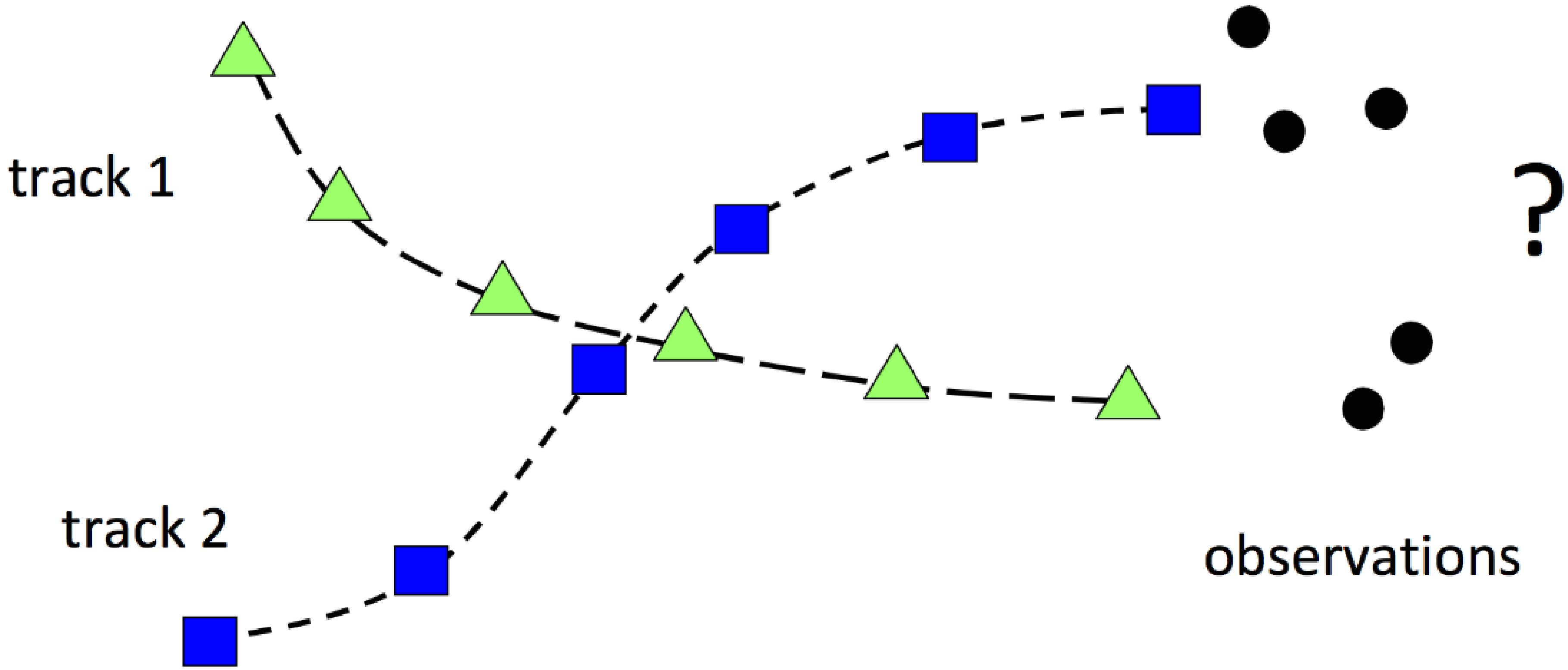


Tracking - Multi-target

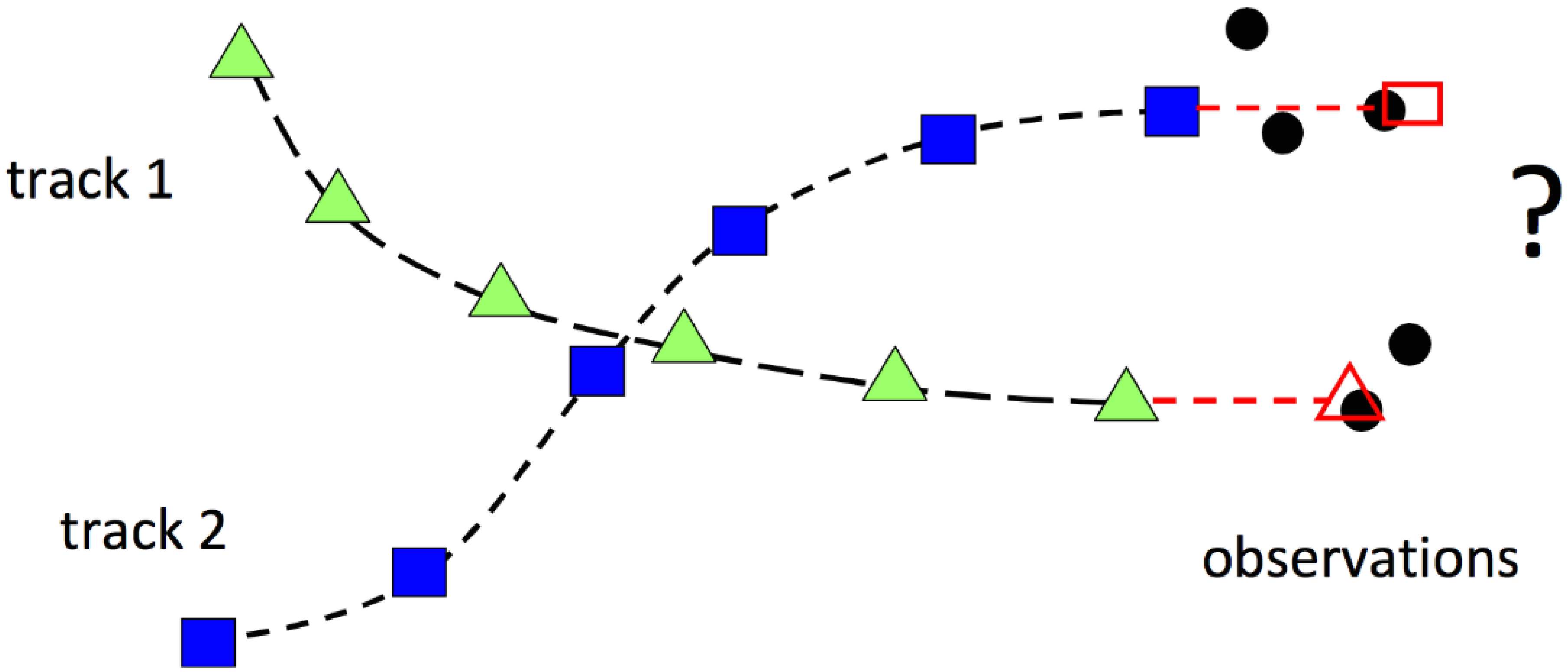
- Input: a set of detection
- Output: state of each detection



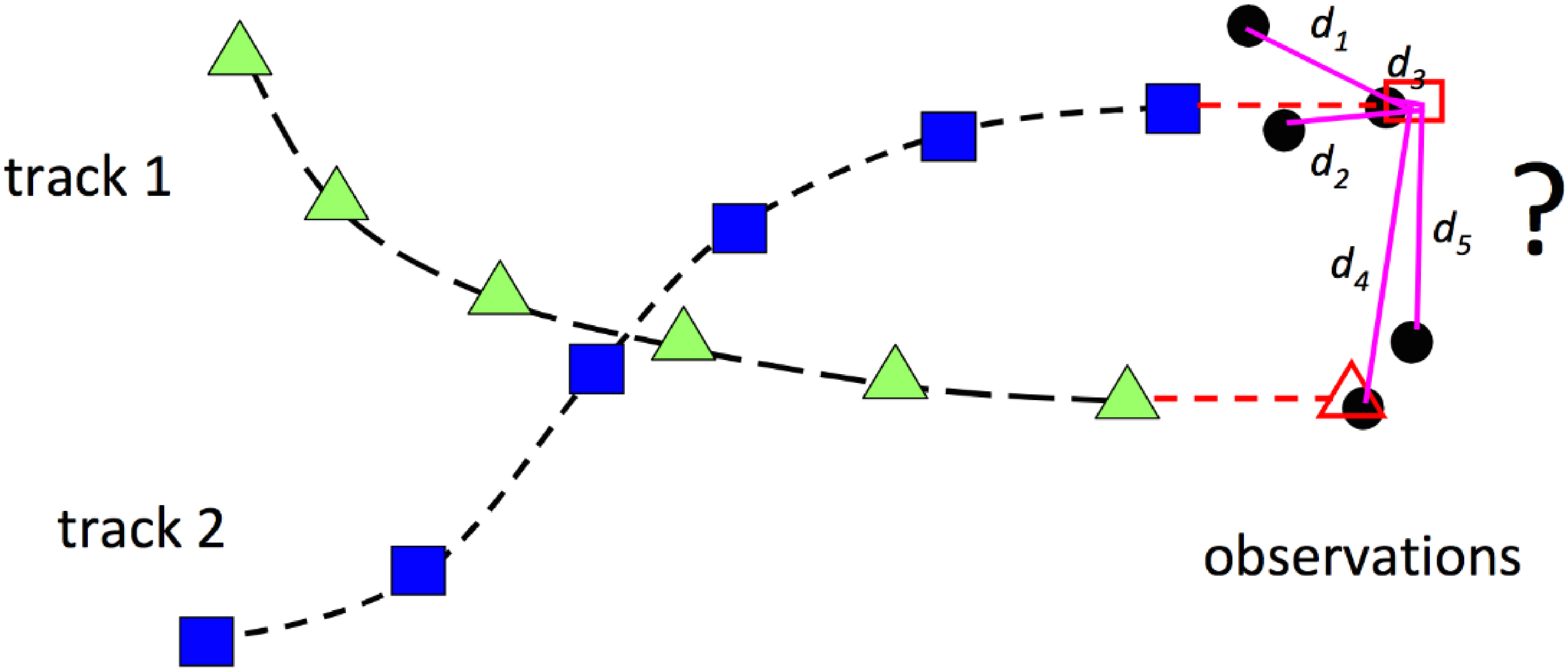
Tracking - Multi-target



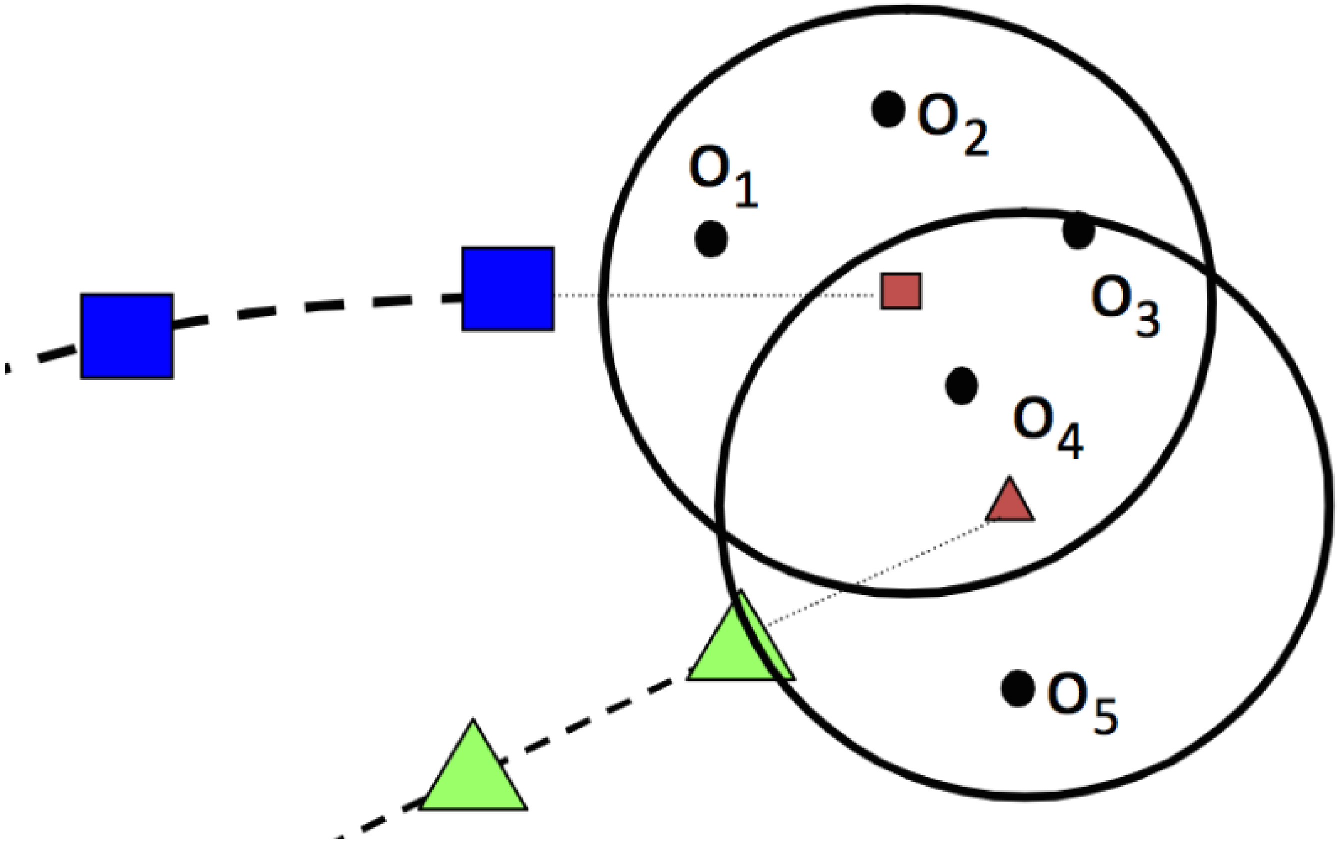
Tracking - Multi-target



Tracking - Multi-target



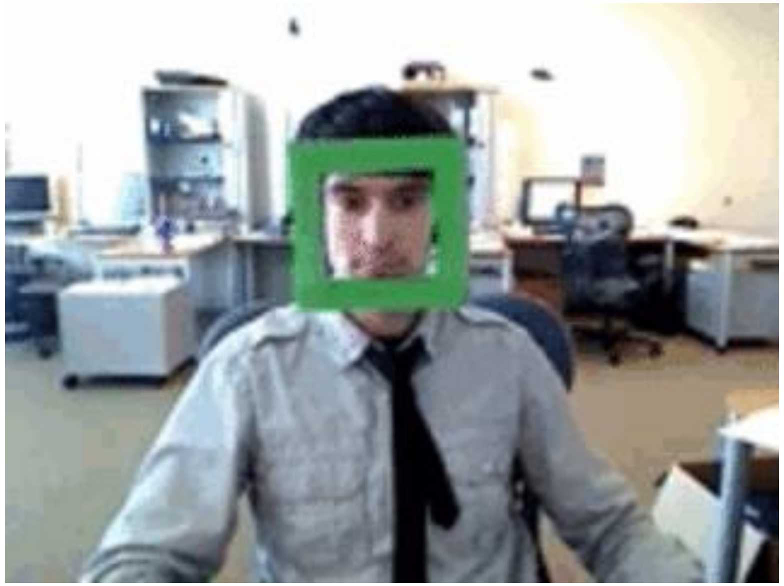
Tracking - Multi-target



	ai1	ai2
1	3.0	
2	5.0	
3	6.0	1.0
4	9.0	8.0
5		3.0

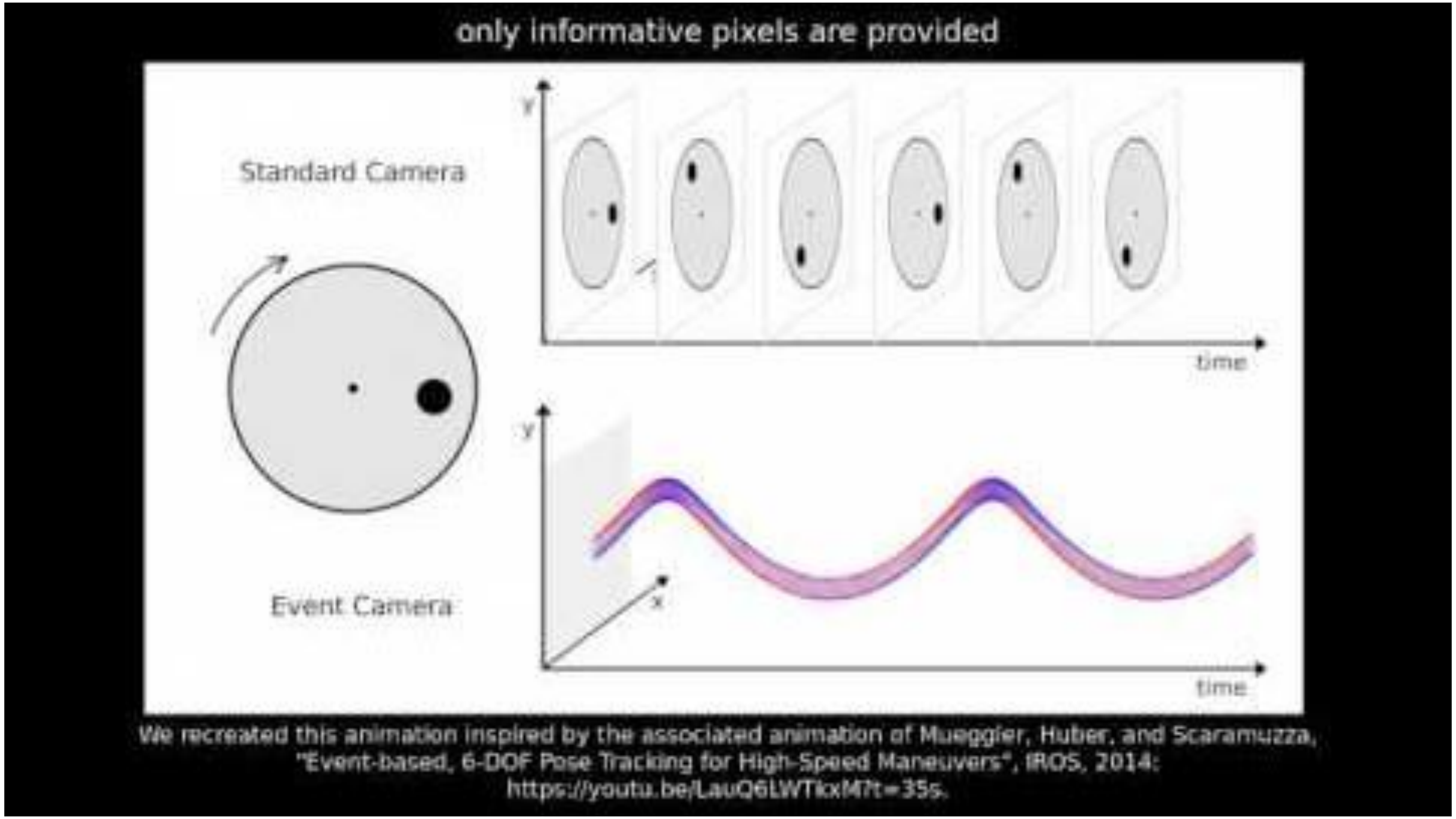
Tracking- Conclusion

- Faster than Detection
  - No need to recompute descriptors on the whole image
    - Select a ROI
  - Uses prior information
- Can help when detection fails
  - Missing frames
  - False detection



- OpenCV API:
  - BOOSTING
  - MIL
  - KCF …

Event-based camera



Event-based camera