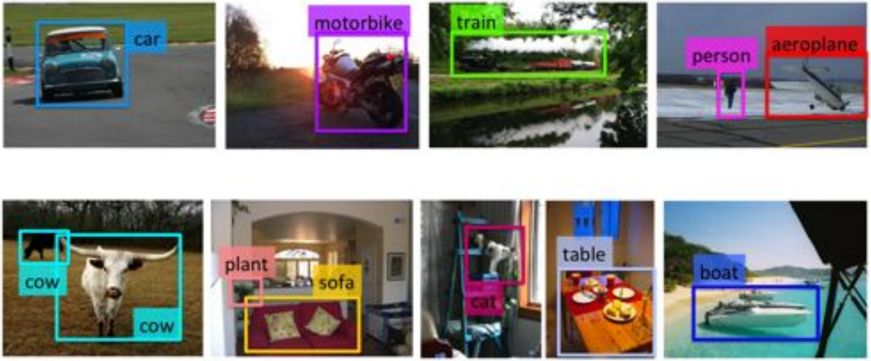
Object Detection

Object Detection

- The goal of object detection is to localize objects in an image and tell their class
- Localization: **place a tight bounding box around object**
- Most approaches find only objects of one or a few specific classes, e.g. car or cow



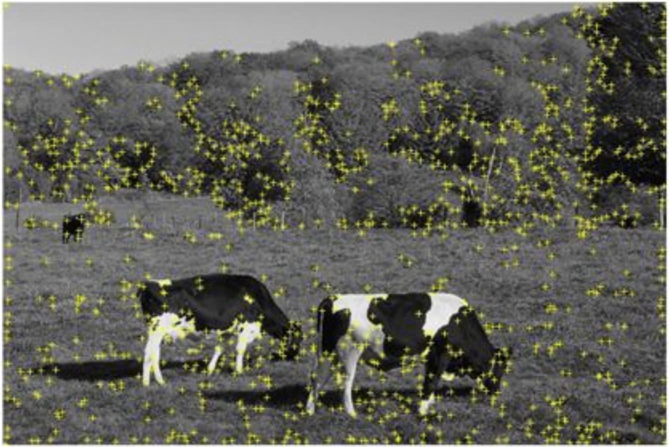
Type of Approaches

Different approaches tackle detection differently. They can roughly be categorized into three main types:

- Find **interest points**, followed by Hough voting

Interest Point Based Approaches

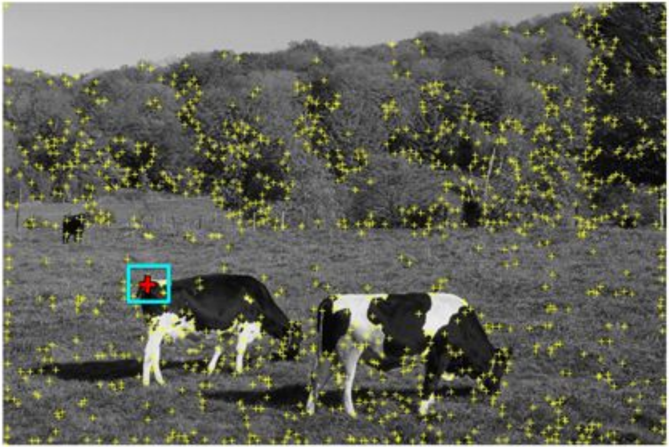
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Interest Point Based Approaches

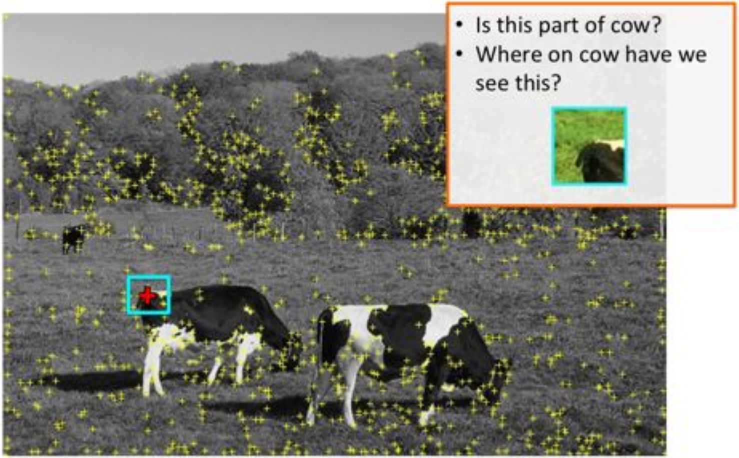
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Interest Point Based Approaches

- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points

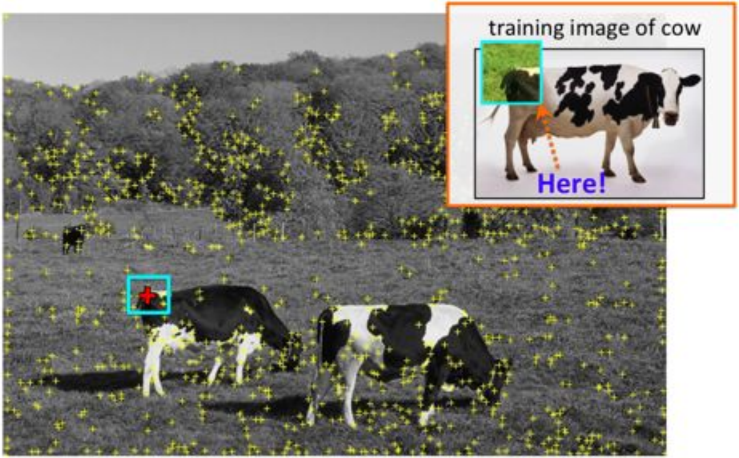


- Is this part of cow?
- Where on cow have we see this?

Interest points

Interest Point Based Approaches

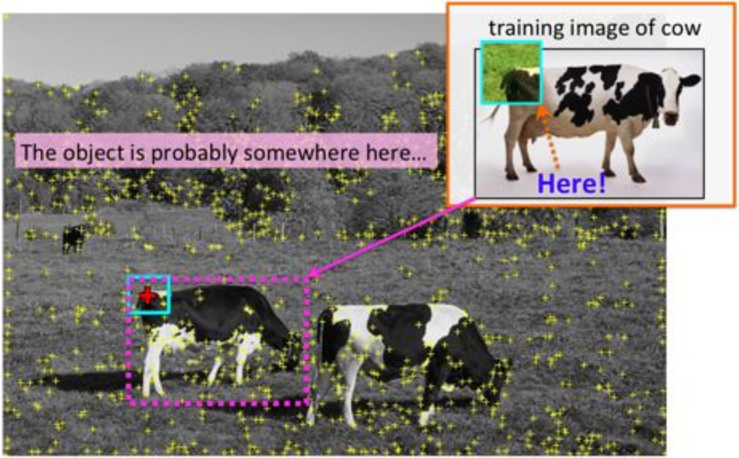
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Interest Point Based Approaches

- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Type of Approaches

Different approaches tackle detection differently. They can roughly be categorized into three main types:

- Find **interest points**, followed by Hough voting
- **Sliding windows: "slide" a box around image and classify each image crop** inside a box (contains object or not?)
- Generate **region (object) proposals**, and classify each region

Type of Approaches

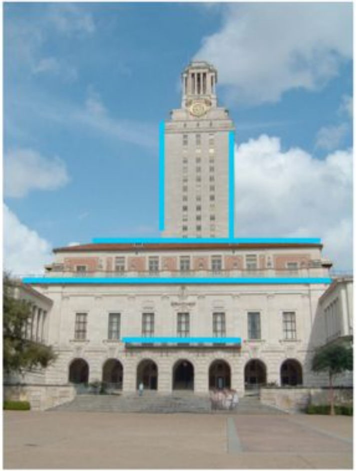
Different approaches tackle detection differently. They can roughly be categorized into three main types:

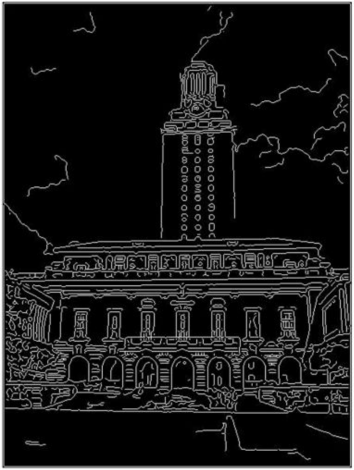
- Find **interest points**, followed by Hough voting ← **Let's first look at one example method for this**
- **Sliding windows**: "slide" a box around image and classify each image crop inside a box (contains object or not?)
- Generate **region (object) proposals**, and classify each region

Object Detection via Hough Voting:

Start with Simple: Line Detection

- How can I find lines in this image?



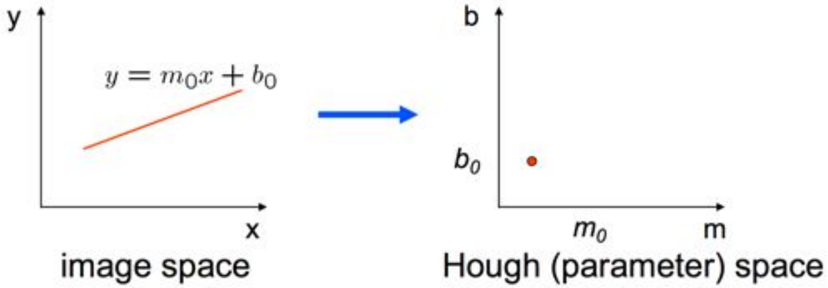


Hough Transform

- Idea: Voting (Hough Transform)
- Voting is a general technique where we let the features vote for all models that are compatible with it.
    - Cycle through features, cast votes for model parameters.
    - Look for model parameters that receive a lot of votes.

Hough Transform: Line Detection

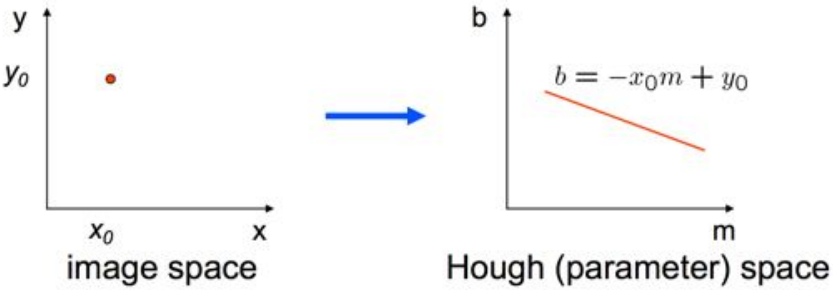
- Hough space: parameter space



- Connection between image (x, y) and Hough (m, b) spaces
  - A line in the image corresponds to a point in Hough space
  - What does a point (x_0, y_0) in the image space map to in Hough space?

Hough Transform: Line Detection

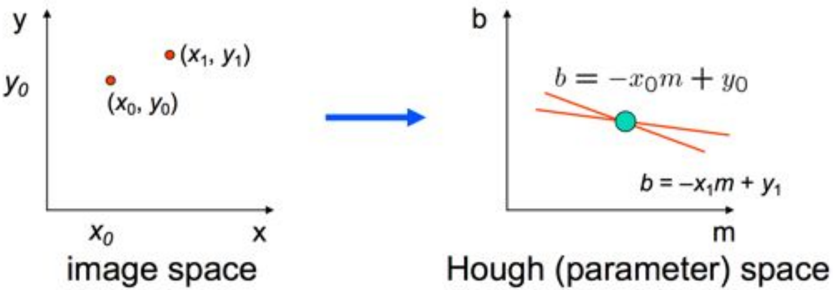
- Hough space: parameter space



- Connection between image (x, y) and Hough (m, b) spaces
  - A line in the image corresponds to a point in Hough space
  - A point in image space votes for all the lines that go through this point. This votes are a line in the Hough space.

Hough Transform: Line Detection

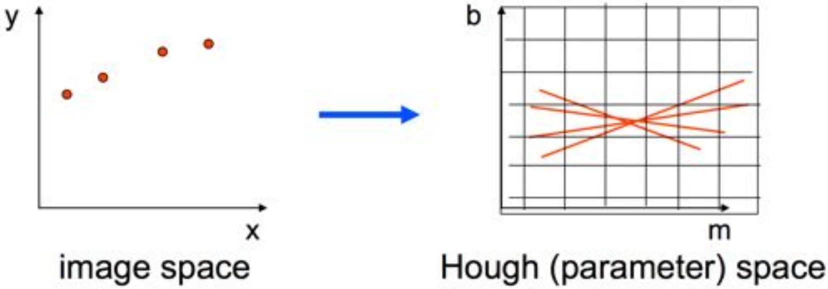
- Hough space: parameter space



- Two points: Each point corresponds to a line in the Hough space
- A point where these two lines meet defines a line in the image!

Hough Transform: Line Detection

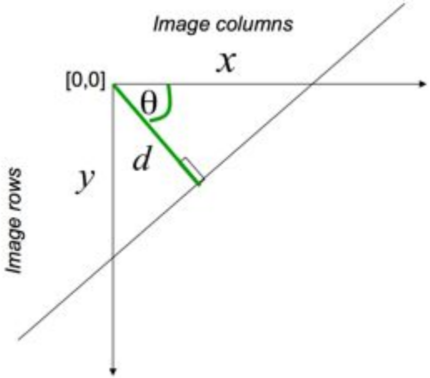
- Hough space: parameter space



- Vote with each image point
- Find peaks in Hough space. Each peak is a line in the image.

Hough Transform: Line Detection

- Issues with usual (m, b) parameter space: undefined for vertical lines
- A better representation is a polar representation of lines



d: perpendicular distance from line to origin

θ: angle the perpendicular makes with the x-axis

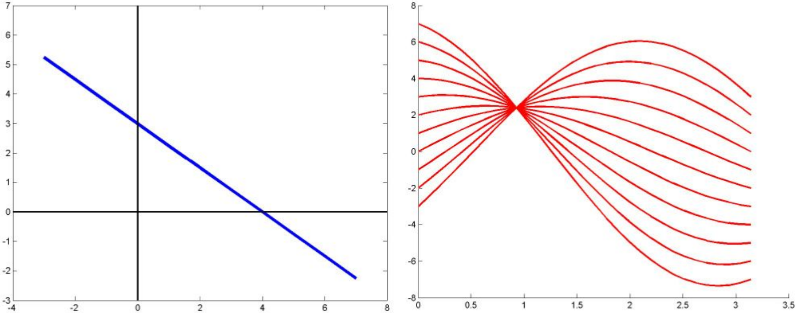
$$x\cos\theta - y\sin\theta = d$$

Point in image space → sinusoid segment in Hough space

Example Hough Transform

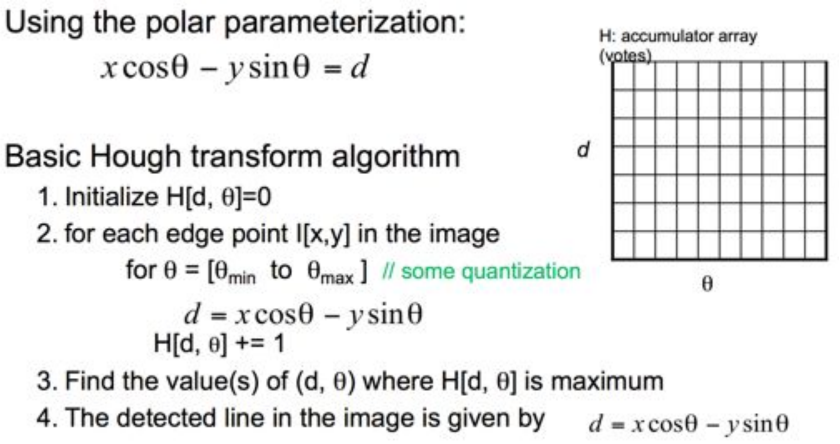
With the parameterization $x\cos\theta + y\sin\theta = d$

- Points in picture represent sinusoids in parameter space
- Points in parameter space represent lines in picture
- Example $0.6x + 0.4y = 2.4$, Sinusoids intersect at $d = 2.4$, $\theta = 0.9273$



Hough Transform: Line Detection

- **Hough Voting algorithm**



Using the polar parameterization:

$$x\cos\theta - y\sin\theta = d$$

Basic Hough transform algorithm

```
1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image
        for θ = [$\theta_{min}$ to $\theta_{max}$ ]  // some quantization
            $d = x\cos\theta - y\sin\theta$
        H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by   $d = x\cos\theta - y\sin\theta$
```

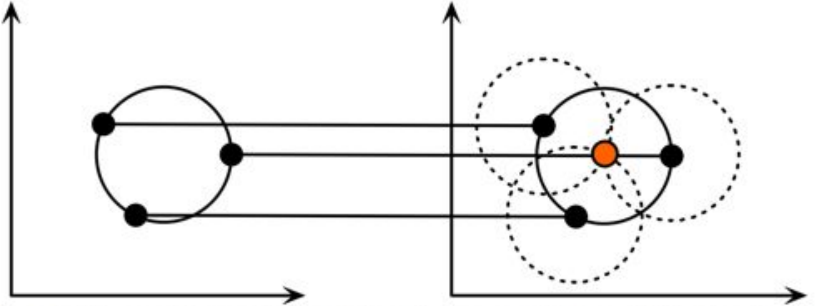
Hough Transform: Circle Detection

- What about circles? How can I fit circles around these coins?

Hough Transform: Circle Detection

Assume we are looking for a circle of known radius r

- Circle: $(x - a)^2 + (y - b)^2 = r^2$
- Hough space (a, b): A point (x_0, y_0) maps to $(a - x_0)^2 + (b - y_0)^2 = r^2 \;\rightarrow\;$ a circle around (x_0, y_0) with radius r
- Each image point votes for a circle in Hough space

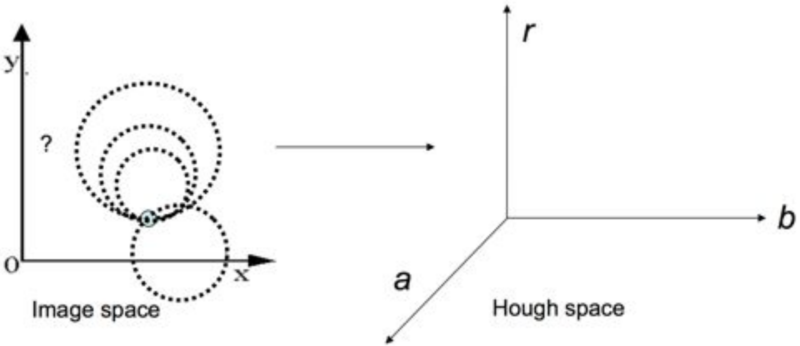


Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

Hough Transform: Circle Detection

What if we don't know *r*?

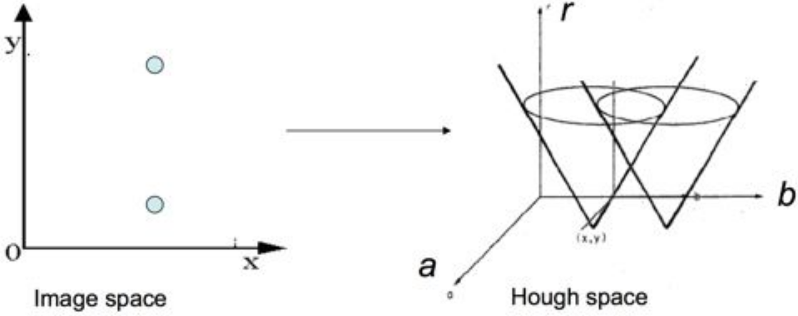
- Hough space: ?



Hough Transform: Circle Detection

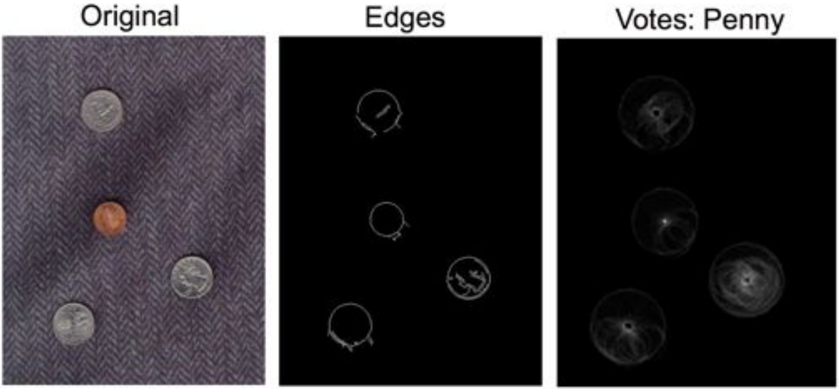
What if we don't know r?

- Hough space: conics



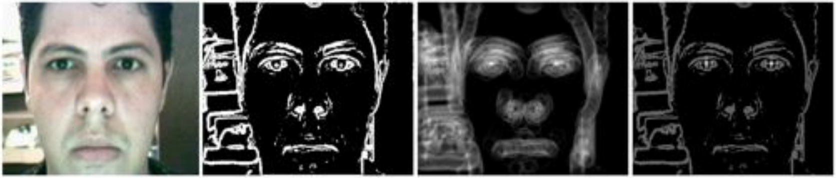
Hough Transform: Circle Detection

- Find the coins



Hough Transform: Circle Detection

- Iris detection



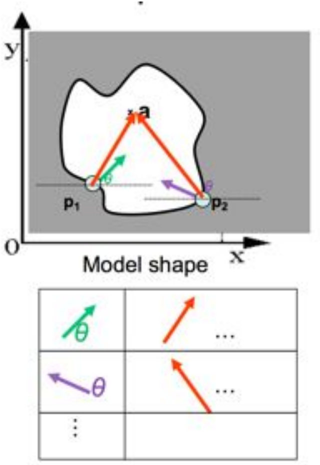
Gradient+threshold

Hough space
(fixed radius)

Max detections

Generalized Hough Voting

- Hough Voting for general shapes



<u>Offline procedure:</u>

At each boundary point, compute displacement vector: **r = a – p**$_i$.

Store these vectors in a table indexed by gradient orientation θ.

Implicit Shape Model

- Implicit Shape Model adopts the idea of voting
- Basic idea:
    - Find interest points in an image
    - Match patch around each interest point to a training patch
    - Vote for object center given that training instance

Scale Invariant Voting

Scale-invariant feature selection

- Scale-invariant interest points
- Rescale extracted patches
- Match to constant-size codebook

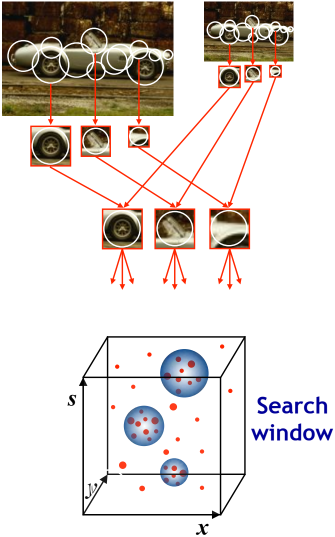
Generate scale votes

- Scale as 3rd dimension in voting space

$$\begin{aligned} x_{vote} &= x_{img} - x_{occ}(s_{img}/s_{occ}) \\ y_{vote} &= y_{img} - y_{occ}(s_{img}/s_{occ}) \\ s_{vote} &= s_{img}/s_{occ} \end{aligned}$$

- Search for maxima in 3D voting space

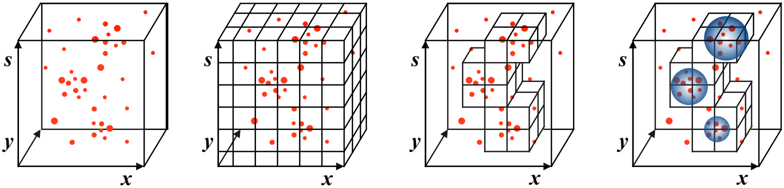
Scale Invariant Voting



Scale Voting: Efficient Computation

Continuous Generalized Hough Transform

- Binned accumulator array similar to standard Gen. Hough Transf.
- Quickly identify candidate maxima locations
- Refine locations by Mean-Shift search only around those points
- Avoid quantization effects by keeping exact vote locations.



Scale votes

Binned accum. array

Candidate maxima

Refinement (Mean-Shift)

Conclusion

- Exploits a lot of parts (as many as interest points)
- Very simple Voting scheme: Generalized Hough Transform
-
-
-