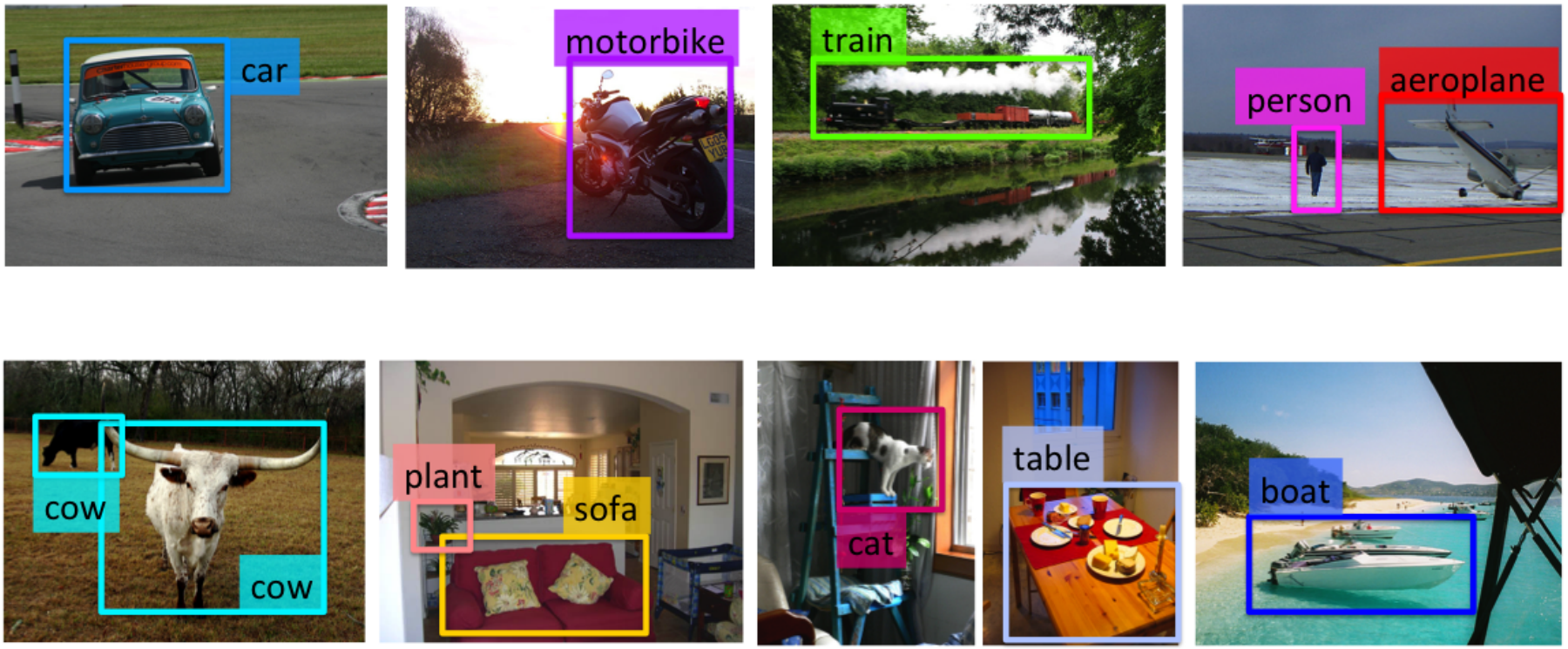
Object Detection

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

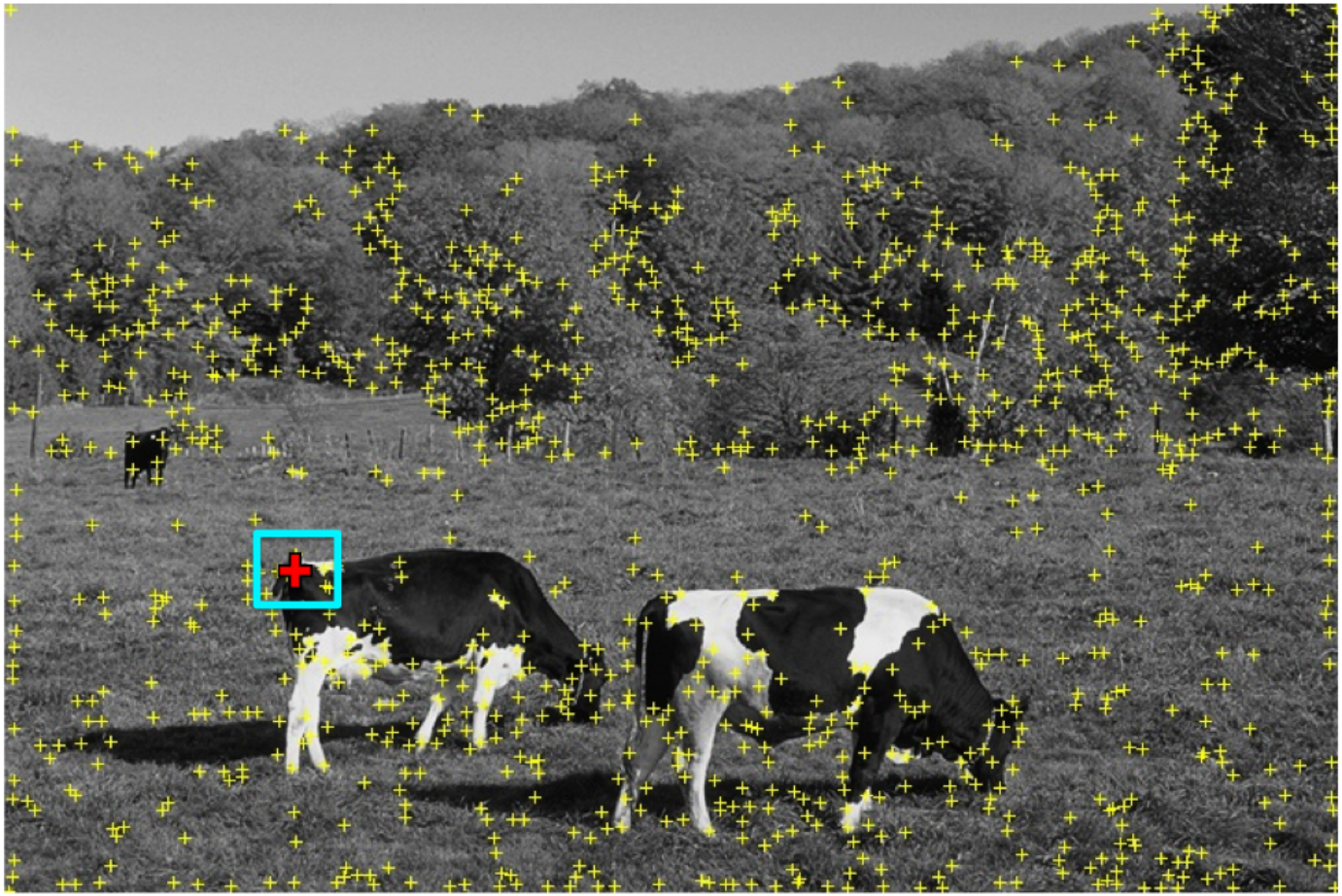
Interest Point Based Approaches

- 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

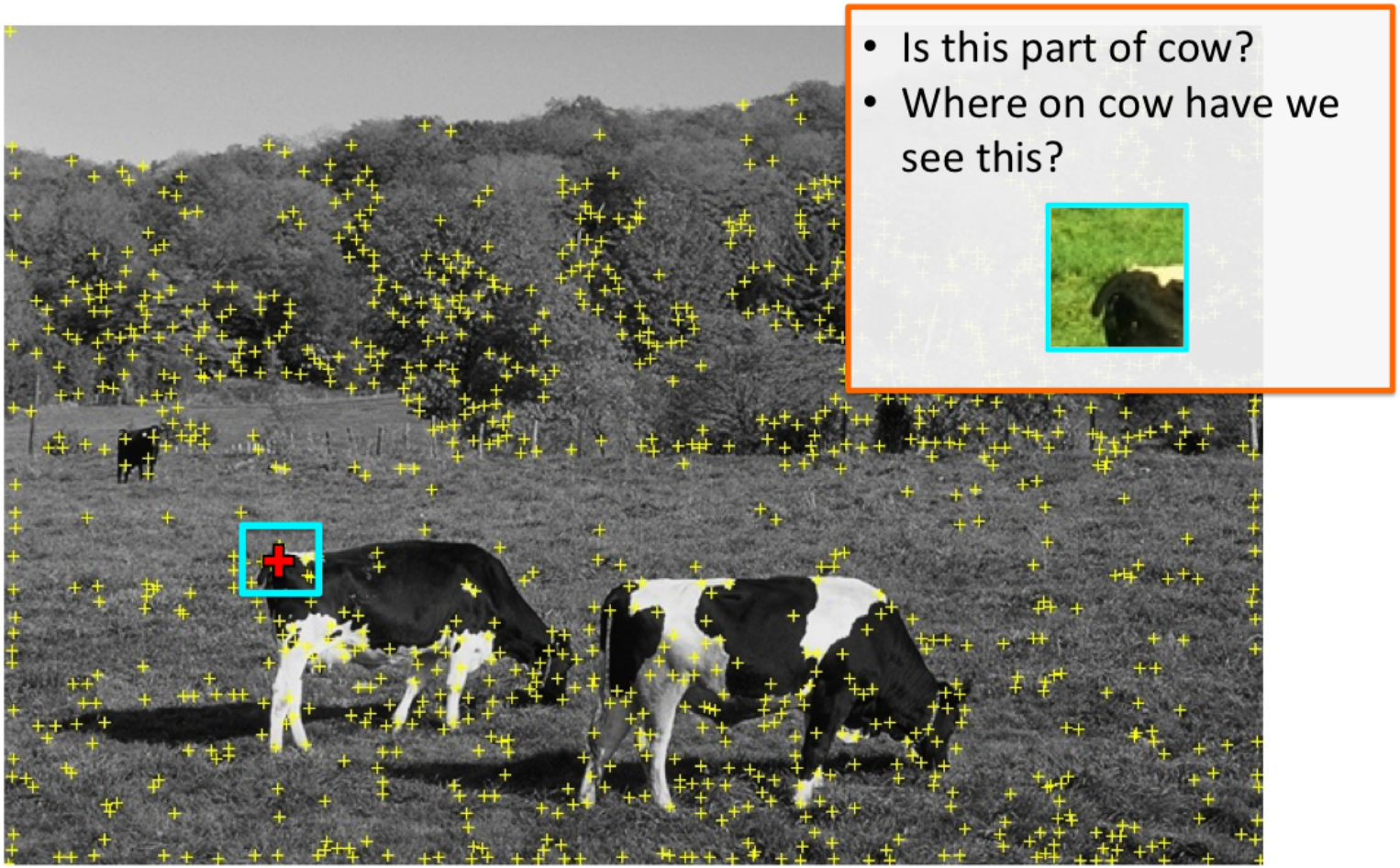
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Interest Point Based Approaches

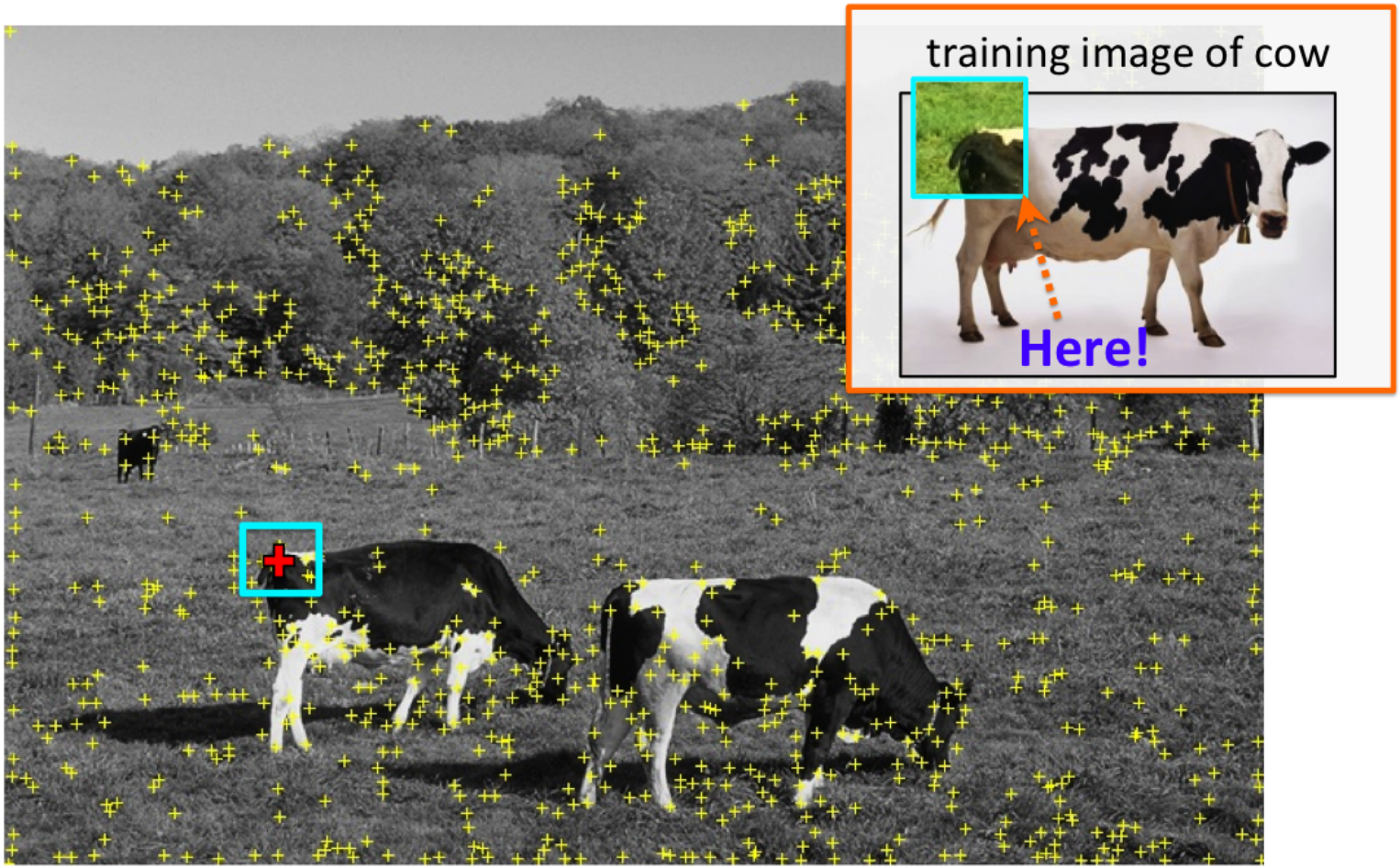
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Interest Point Based Approaches

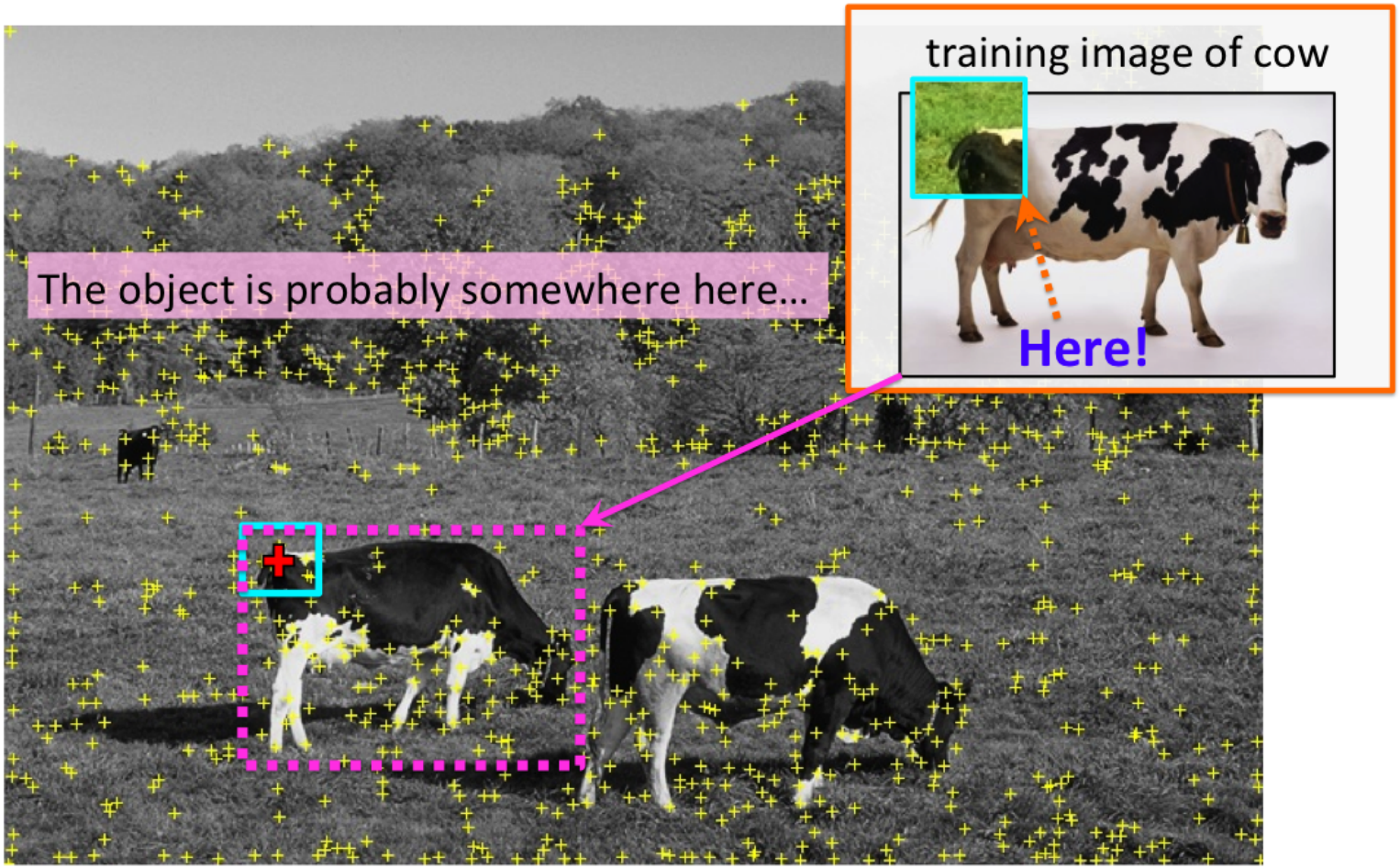
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



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



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?)

Sliding Window Approaches

- Slide window and ask a classifier: "Is sheep in window or not?"

0.1
confidence

[Slide: R. Urtasun]

Sliding Window Approaches

- Slide window and ask a classifier: "Is sheep in window or not?"

-0.2

[Slide: R. Urtasun]

Sliding Window Approaches

- Slide window and ask a classifier: "Is sheep in window or not?"

-0.1

[Slide: R. Urtasun]

Sliding Window Approaches

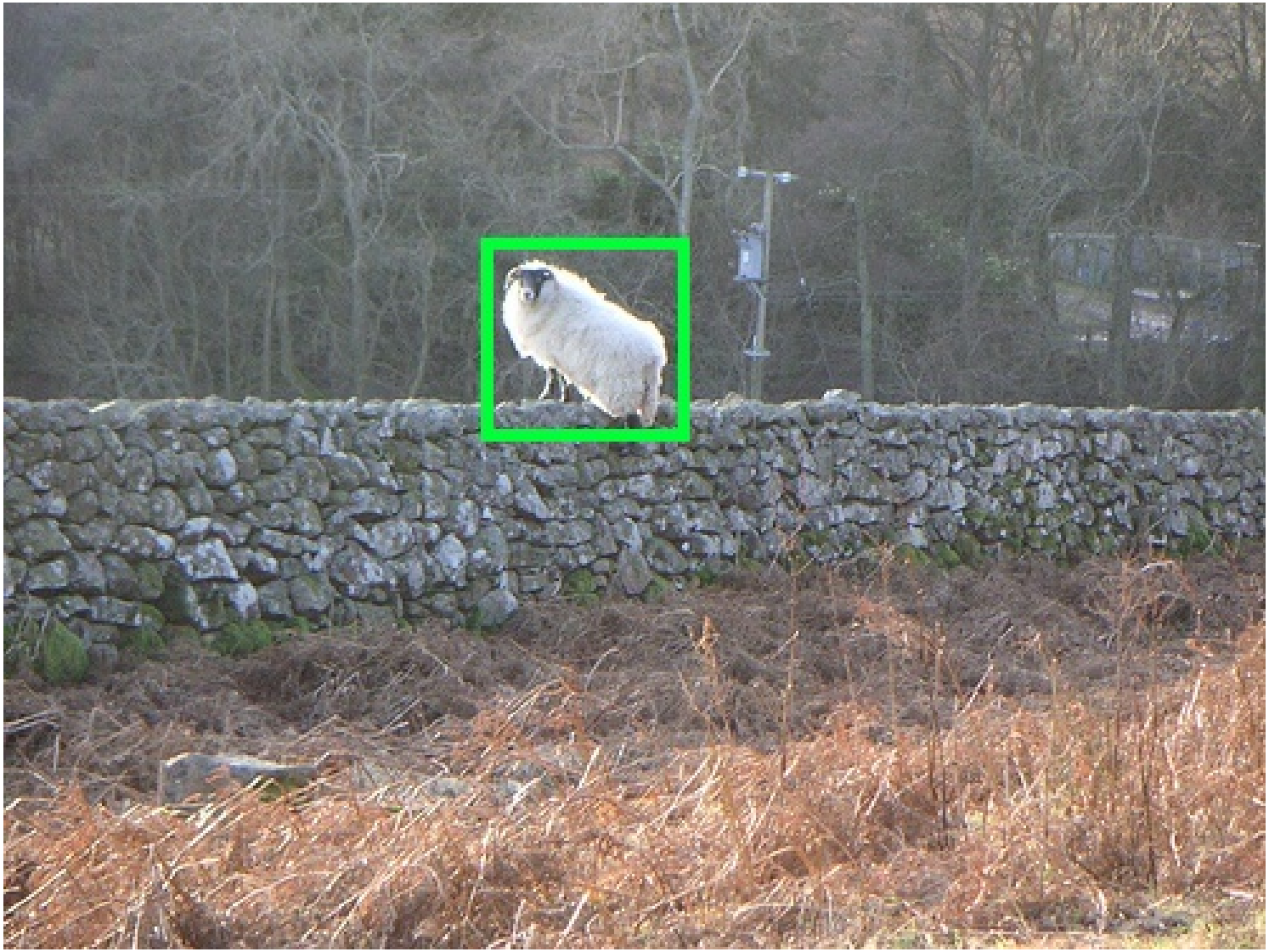
- Slide window and ask a classifier: "Is sheep in window or not?"

0.1

[Slide: R. Urtasun]

Sliding Window Approaches

- Slide window and ask a classifier: "Is sheep in window or not?"



...
1.5
...

[Slide: R. Urtasun]

Sliding Window Approaches

- Slide window and ask a classifier: "Is sheep in window or not?"

0.5

[Slide: R. Urtasun]

Sliding Window Approaches

- Slide window and ask a classifier: "Is sheep in window or not?"

0.4

[Slide: R. Urtasun]

Sliding Window Approaches

- Slide window and ask a classifier: "Is sheep in window or not?"

0.3

[Slide: R. Urtasun]

Sliding Window Approaches

- Slide window and ask a classifier: "Is sheep in window or not?"

0.1
confidence-
0.2
-0.1
0.1
...
1.5
...
0.5
0.4
0.3

[Slide: R. Urtasun]

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

Region Proposal Based Approaches

- Group pixels into object-like regions

Region Proposal Based Approaches

- Group pixels into object-like regions

Region Proposal Based Approaches

- Group pixels into object-like regions

Region Proposal Based Approaches

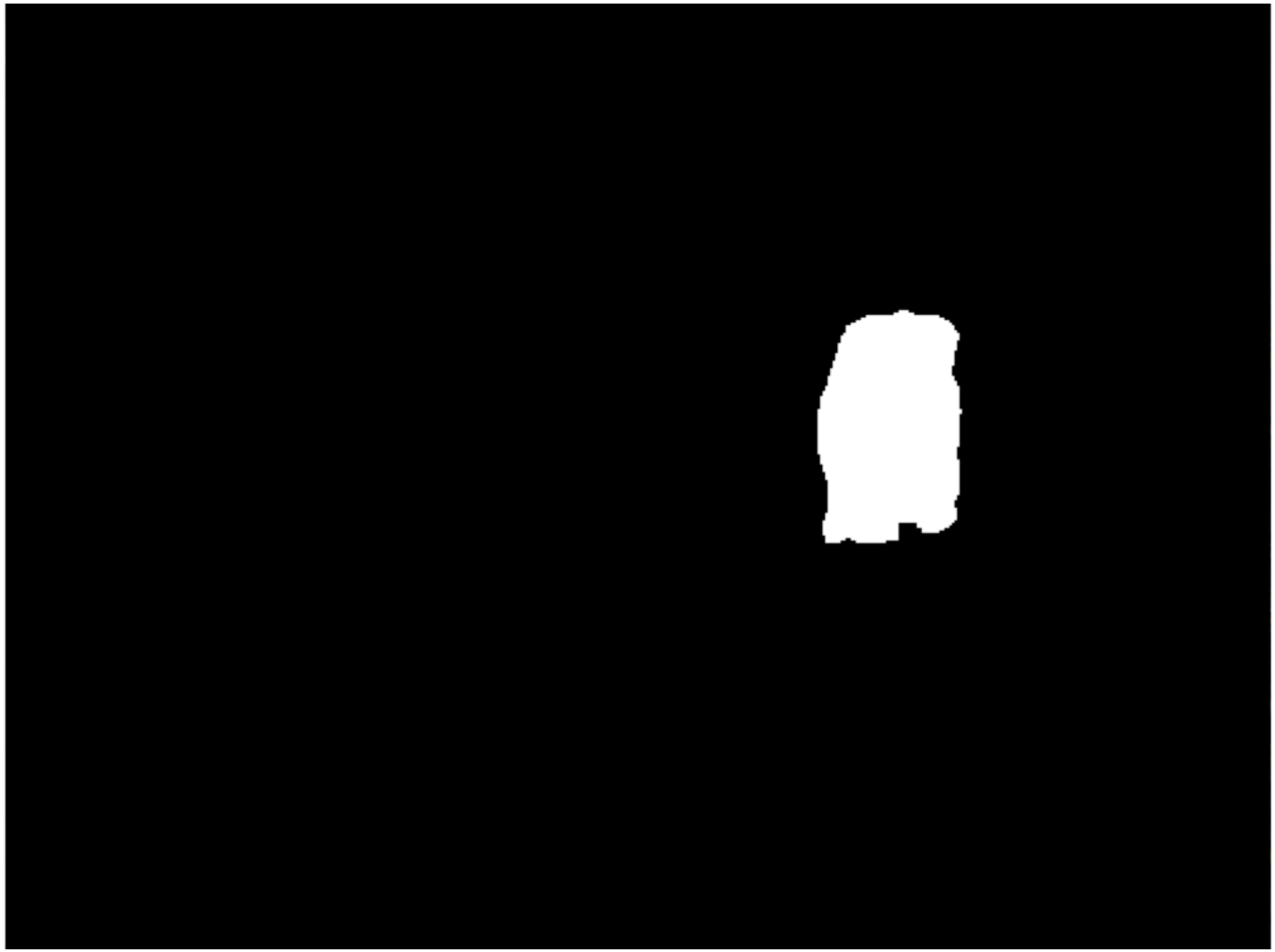
- Generate **many** different regions

Region Proposal Based Approaches

- Generate **many** different regions

Region Proposal Based Approaches

- Generate **many** different regions



Region Proposal Based Approaches

- The hope is that at least a few will cover real objects

Region Proposal Based Approaches

- The hope is that at least a few will cover real objects

Region Proposal Based Approaches

- Select a region

Region Proposal Based Approaches

- Crop out an image patch around it, throw to classifier (e.g., Neural Net)

classifier
``dog'' or not?

confidence: -2.5

Region Proposal Based Approaches

- Do this for every region

Region Proposal Based Approaches

- Do this for every region

Region Proposal Based Approaches

- Do this for every region

classifier
``dog'' or not?

confidence: 1.5

Dog!!!

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:

Implicit Shape Model

B. Leibe, A. Leonardis, B. Schiele

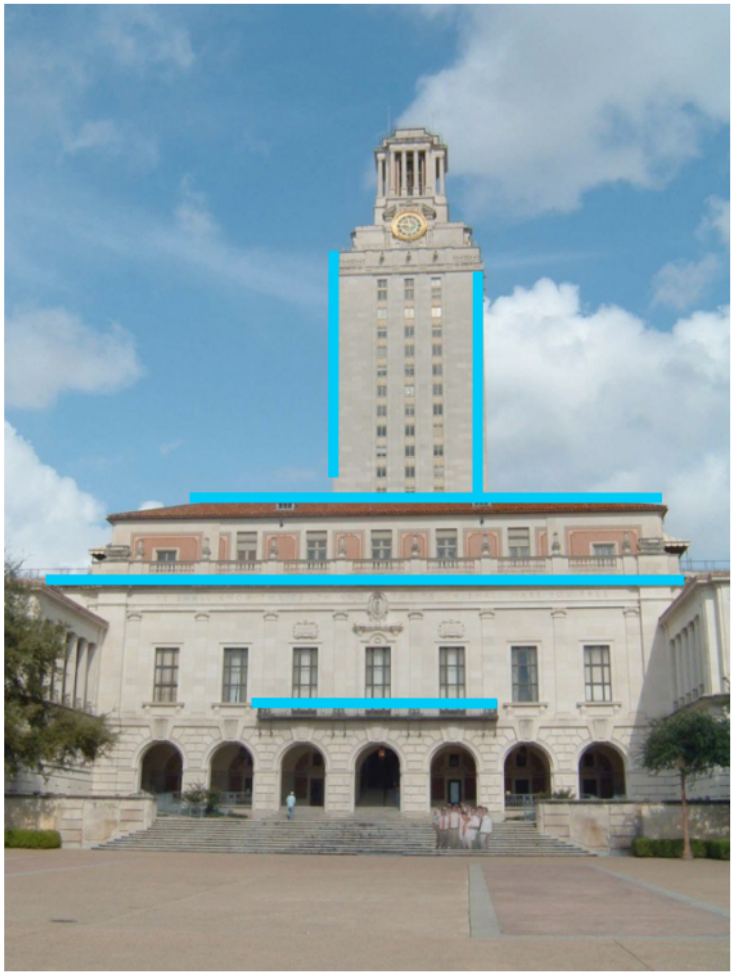
Robust Object Detection with Interleaved Categorization and Segmentation

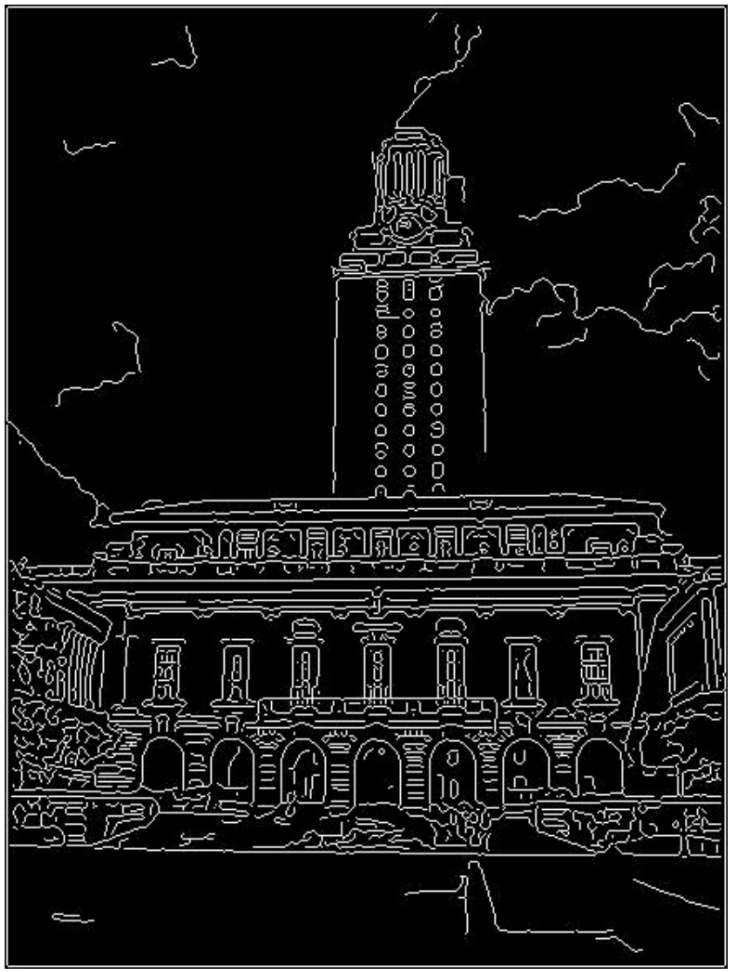
IJCV, 2008

Paper: `http://www.vision.rwth-aachen.de/publications/pdf/leibe-interleaved-ijcv07final.pdf`

Start with Simple: Line Detection

- How can I find lines in this image?





[Source: K. Grauman]

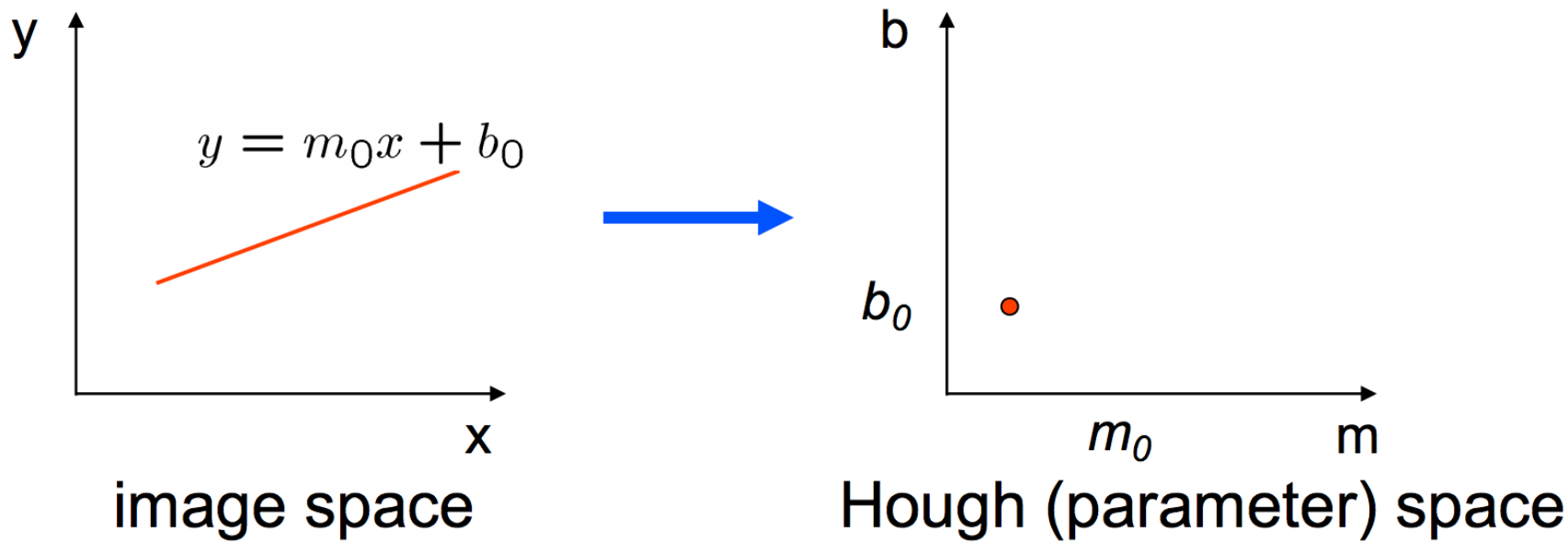
Hough Transform

- 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.

[Source: K. Grauman]

Hough Transform: Line Detection

- Hough space: parameter space

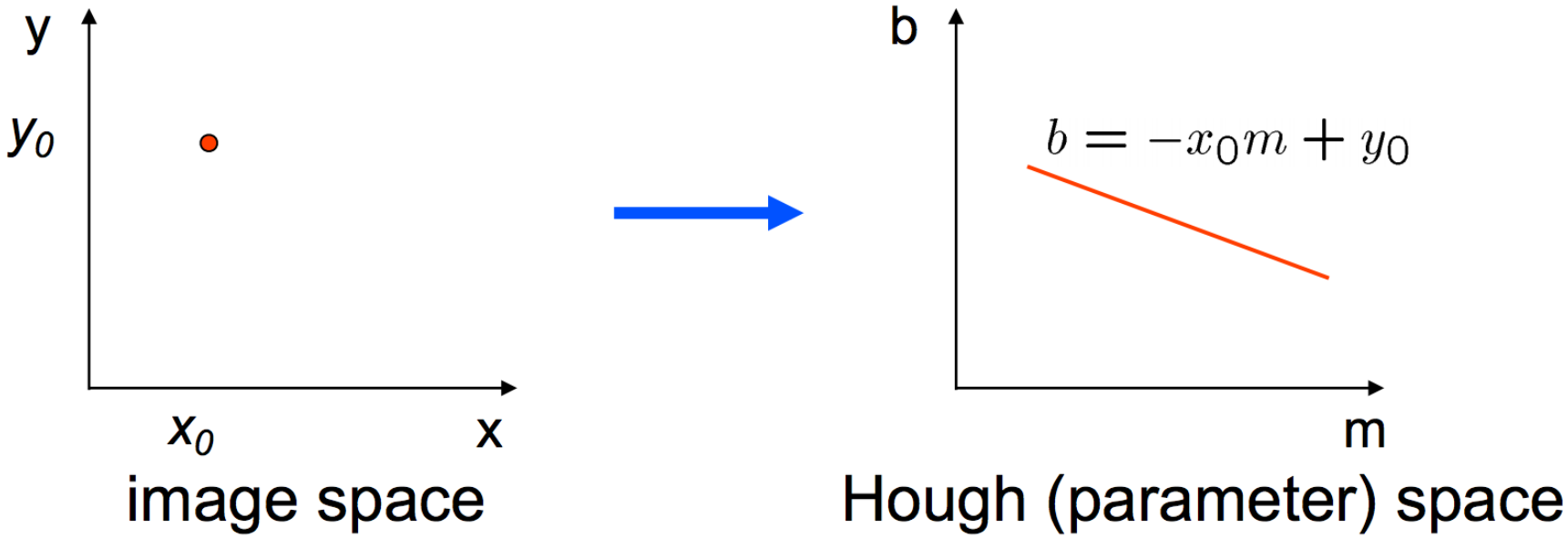


- Connection between image (x, y) and Hough (m, b) spaces
  - A line in the image corresponds to a point in Hough space
  - What does a point (x_0, y_0) in the image space map to in Hough space?

[Source: S. Seitz]

Hough Transform: Line Detection

- Hough space: parameter space

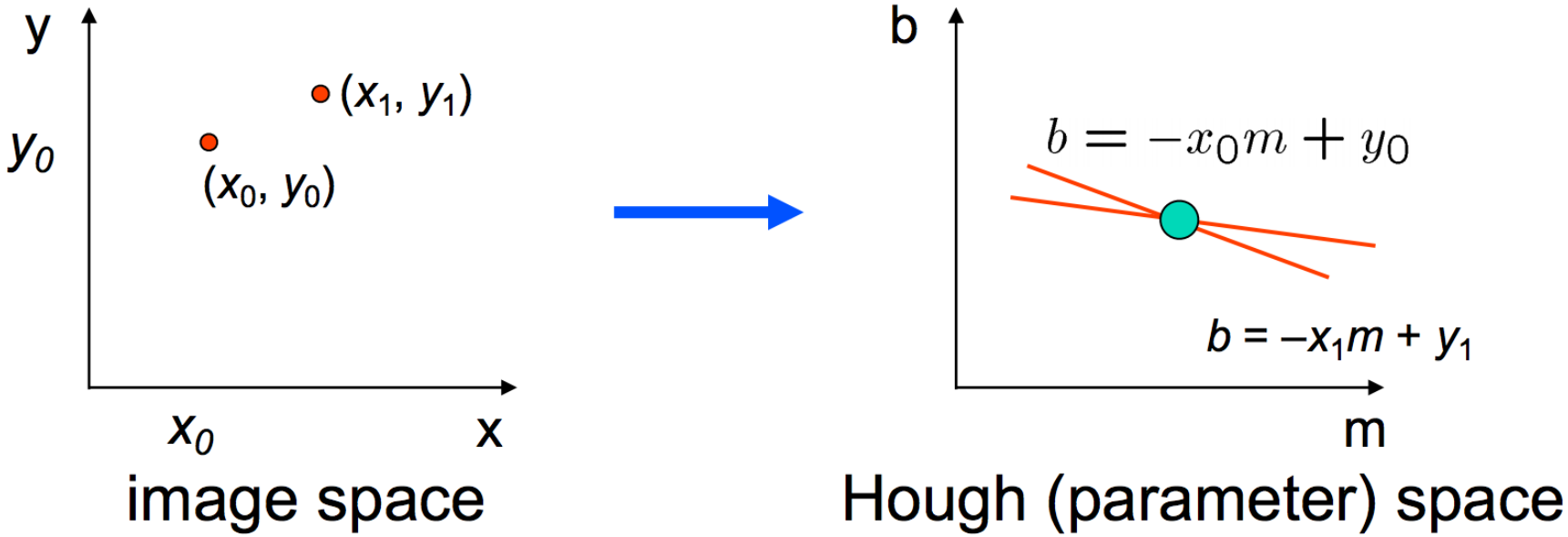


- Connection between image (x, y) and Hough (m, b) spaces
  - A line in the image corresponds to a point in Hough space
  - A point in image space votes for all the lines that go through this point. This votes are a line in the Hough space.

[Source: S. Seitz]

Hough Transform: Line Detection

- Hough space: parameter space

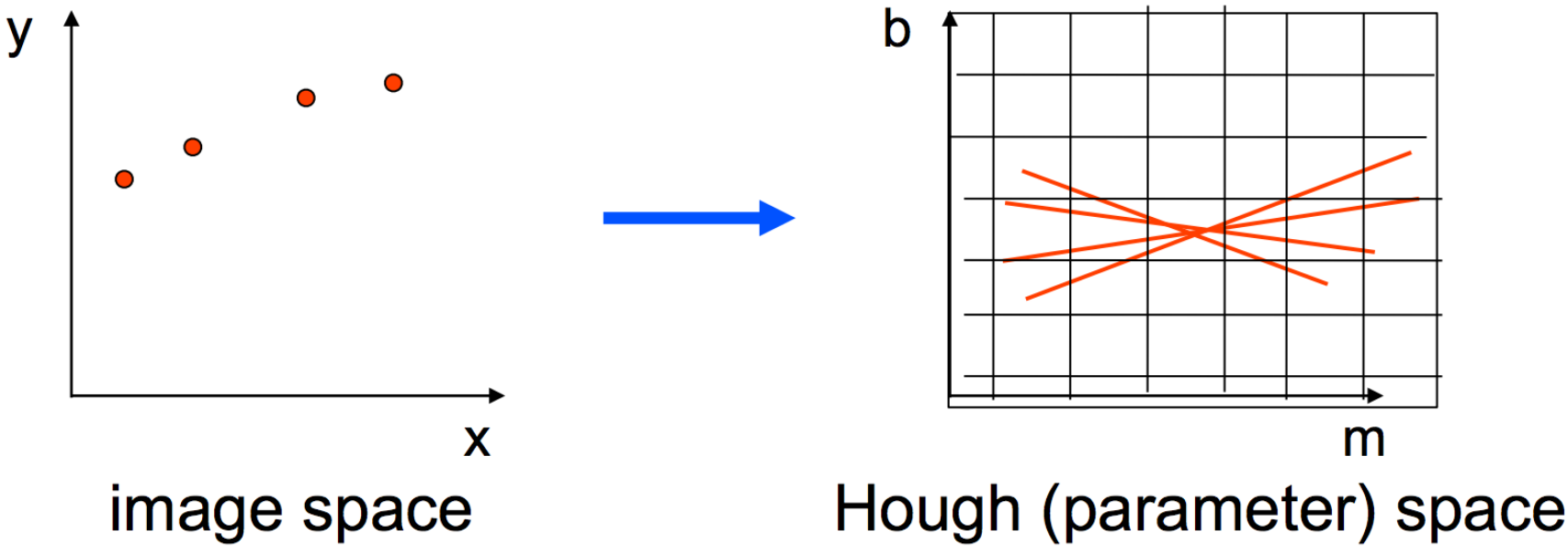


- Two points: Each point corresponds to a line in the Hough space
- A point where these two lines meet defines a line in the image!

[Source: S. Seitz]

Hough Transform: Line Detection

- Hough space: parameter space

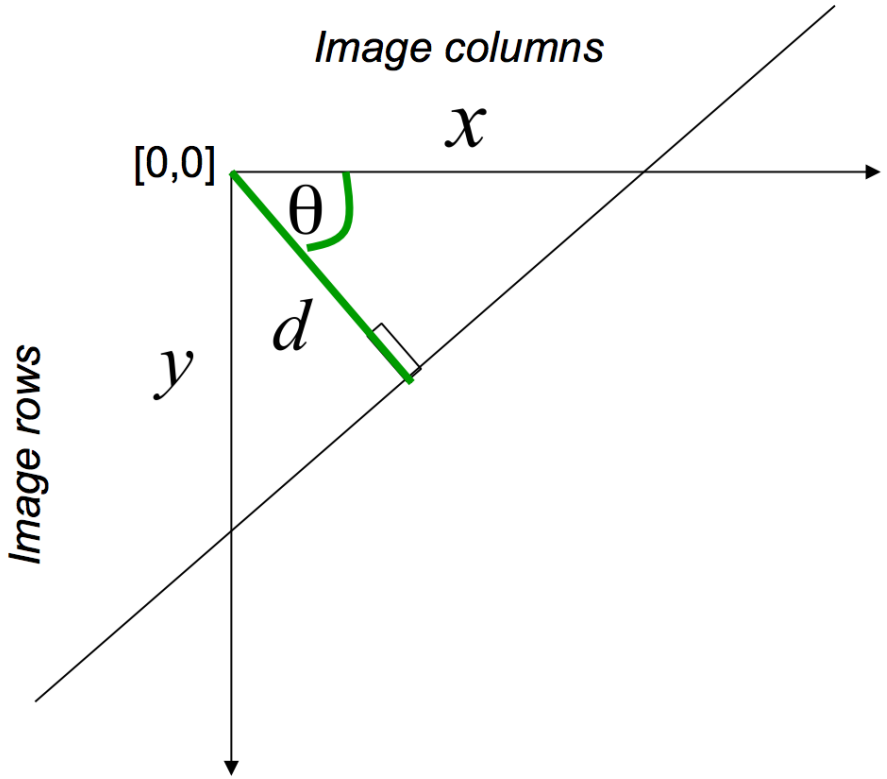


- Vote with each image point
- Find peaks in Hough space. Each peak is a line in the image.

[Source: S. Seitz]

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

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

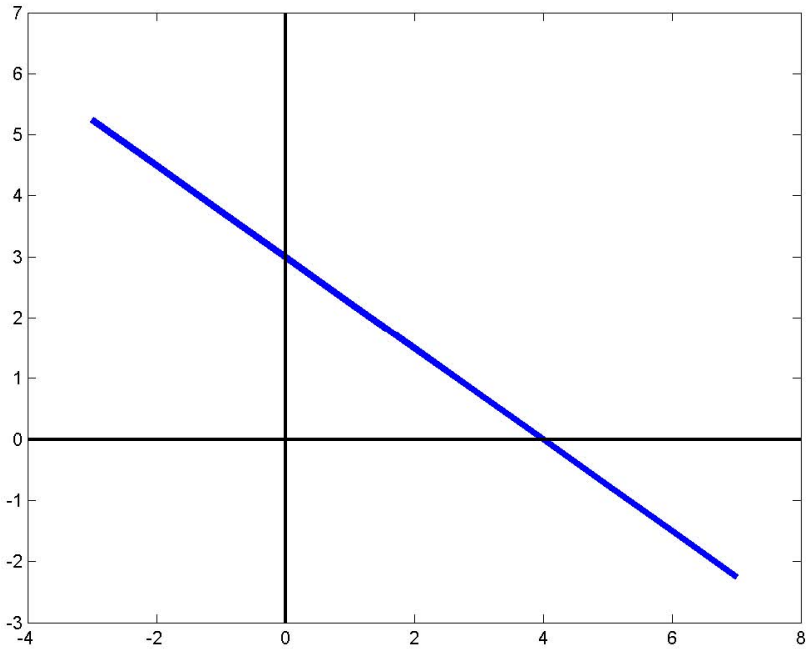
Point in image space → sinusoid segment in Hough space

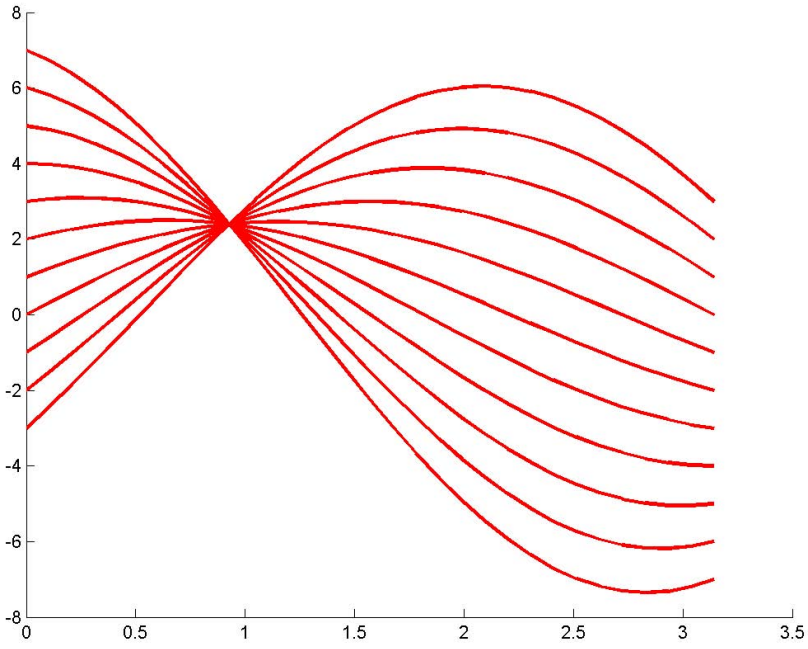
[Source: S. Seitz]

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 $d = 2.4$, $\theta = 0.9273$





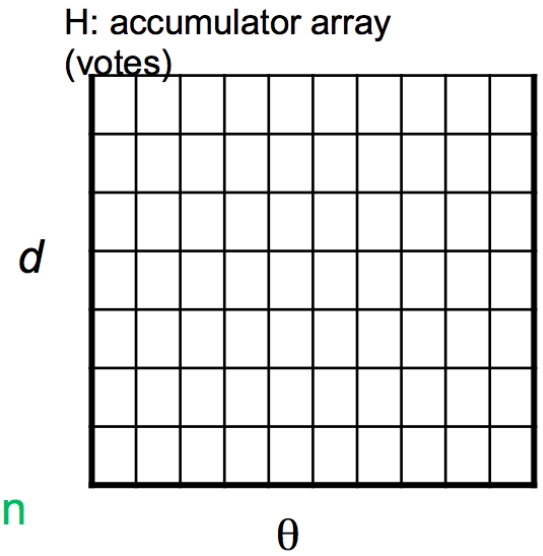
[Source: M. Kazhdan, slide credit: R. Urtasun]

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 = [θ_{min} to θ_{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 $d = x\cos\theta - y\sin\theta$

[Source: S. Seitz]

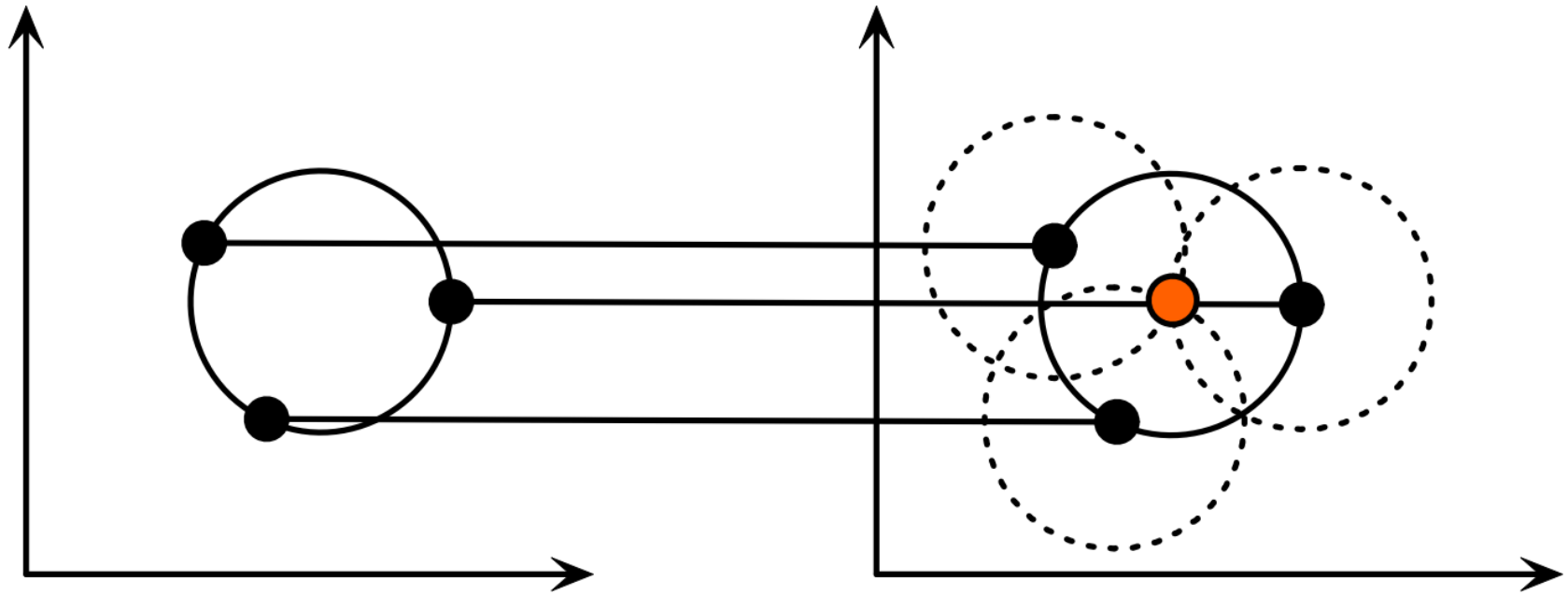
Hough Transform: Circle Detection

- 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



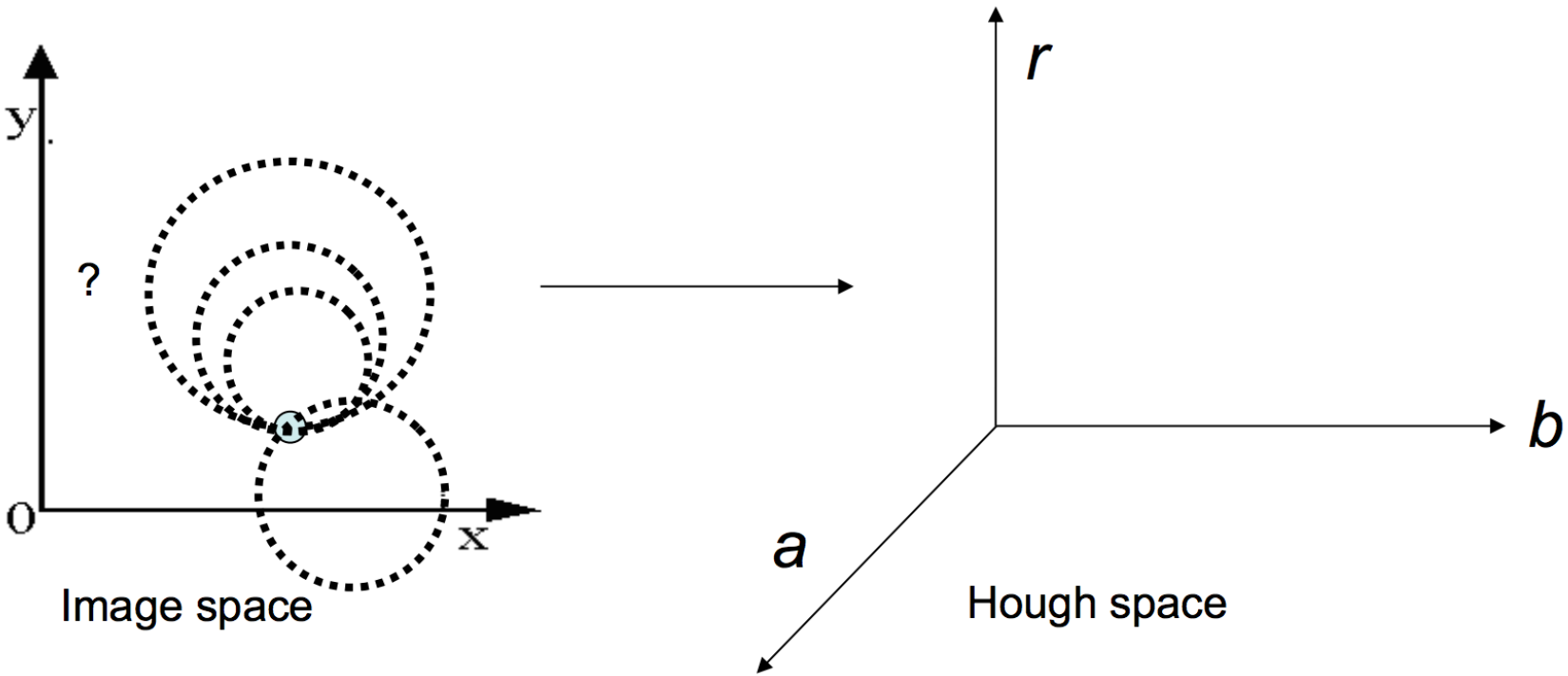
Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

[Source: H. Rhody]

Hough Transform: Circle Detection

What if we don't know r?

- Hough space: ?

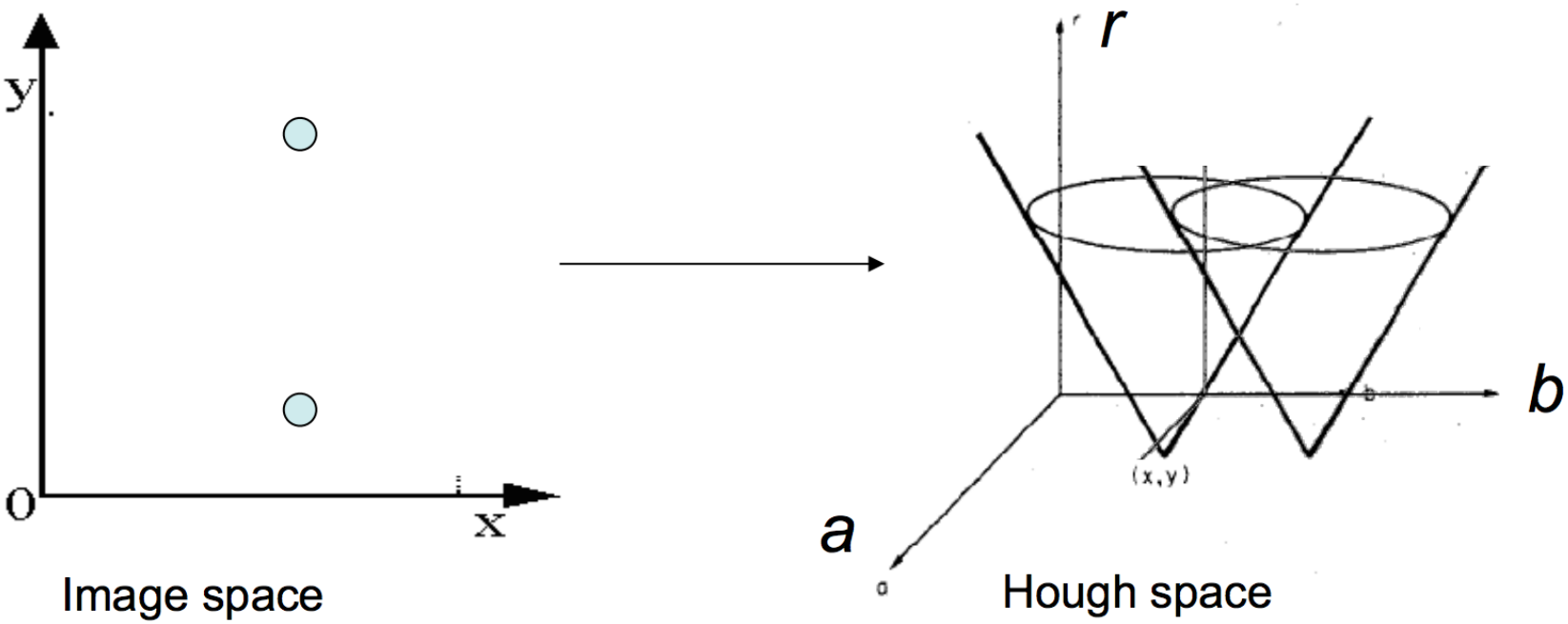


[Source: K. Grauman]

Hough Transform: Circle Detection

What if we don't know r?

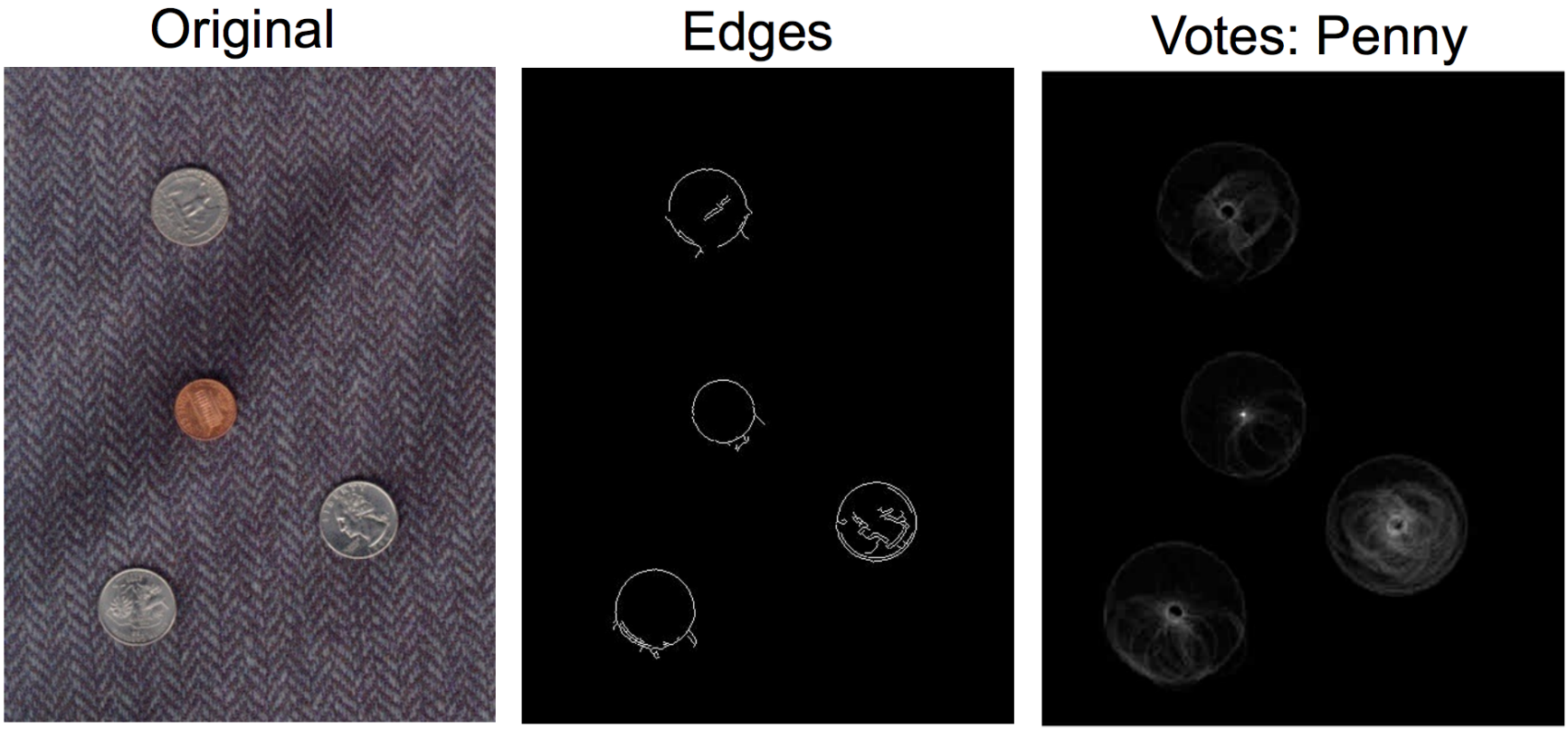
- Hough space: conics



[Source: K. Grauman]

Hough Transform: Circle Detection

- Find the coins

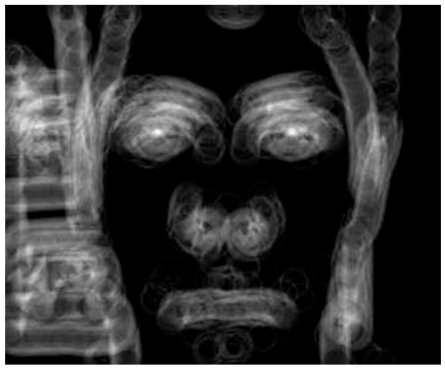


[Source: K. Grauman]

Hough Transform: Circle Detection

- Iris detection

Gradient+threshold



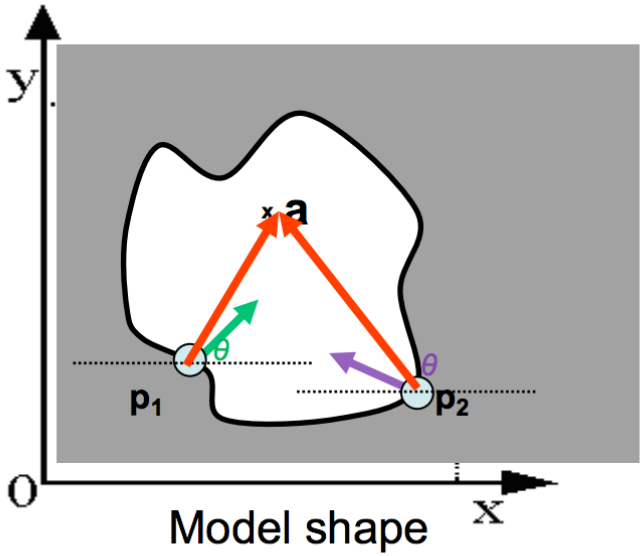
Hough space
(fixed radius)

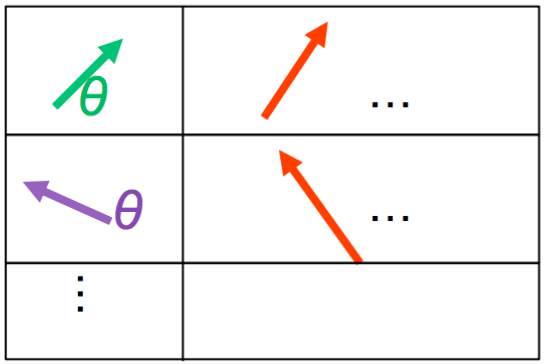
Max detections

[Source: K. Grauman]

Generalized Hough Voting

- Hough Voting for general shapes





Offline procedure:

At each boundary point, compute displacement vector: $\mathbf{r} = \mathbf{a} - \mathbf{p}_i$.

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

Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980

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

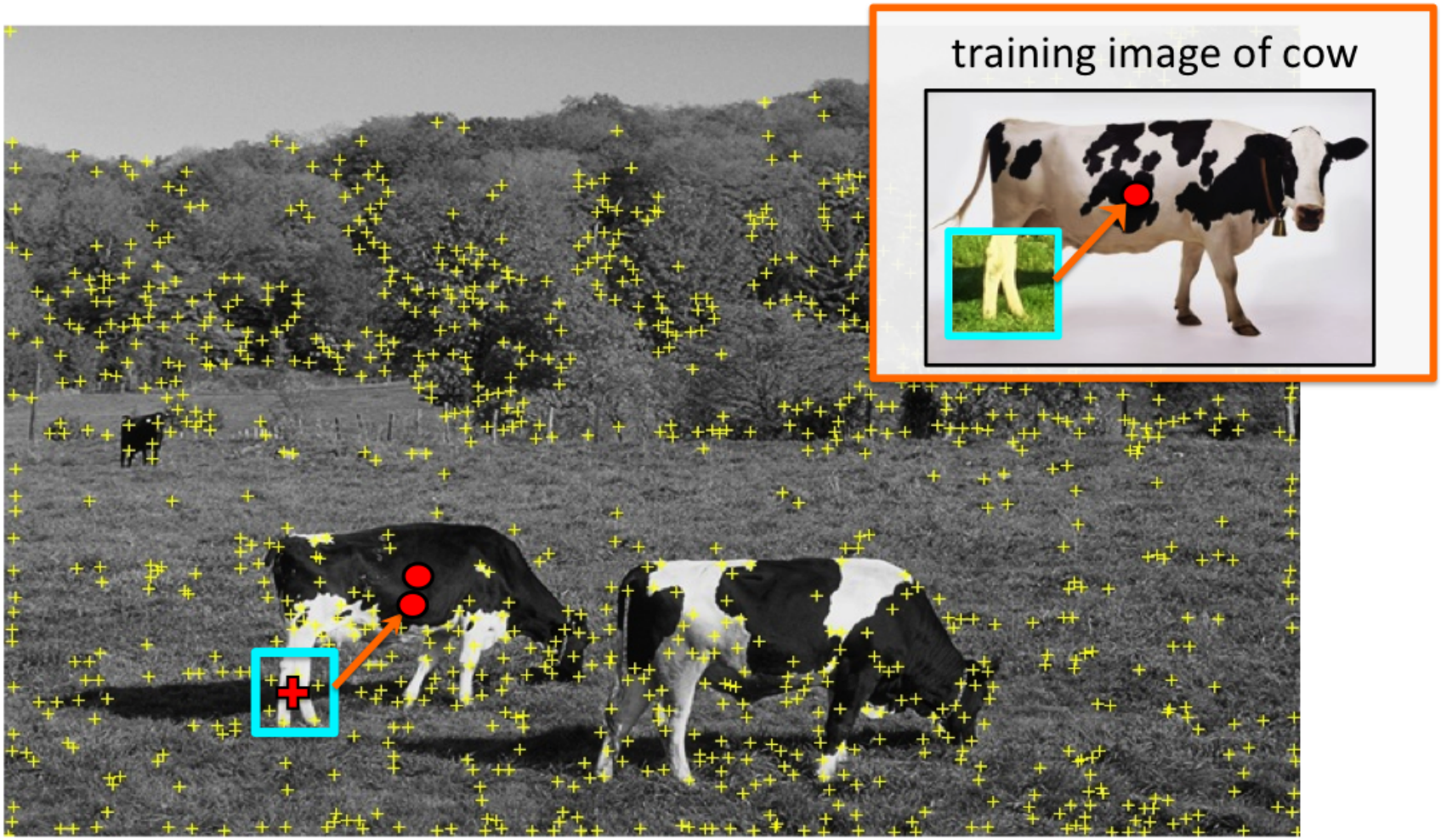
Implicit Shape Model: Basic Idea

- Vote for object center

vote for center of object

Implicit Shape Model: Basic Idea

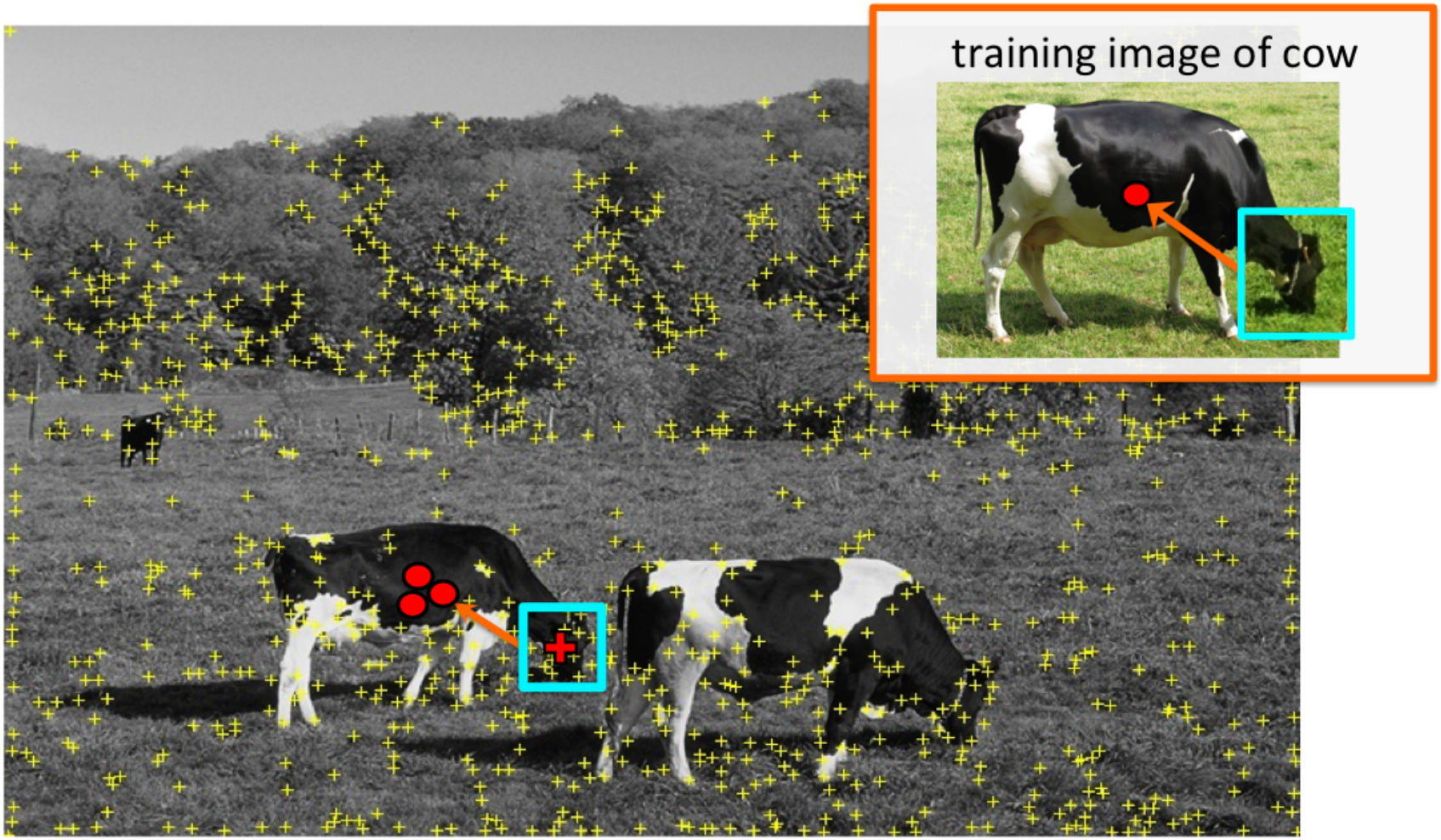
- Vote for object center



vote for center of object

Implicit Shape Model: Basic Idea

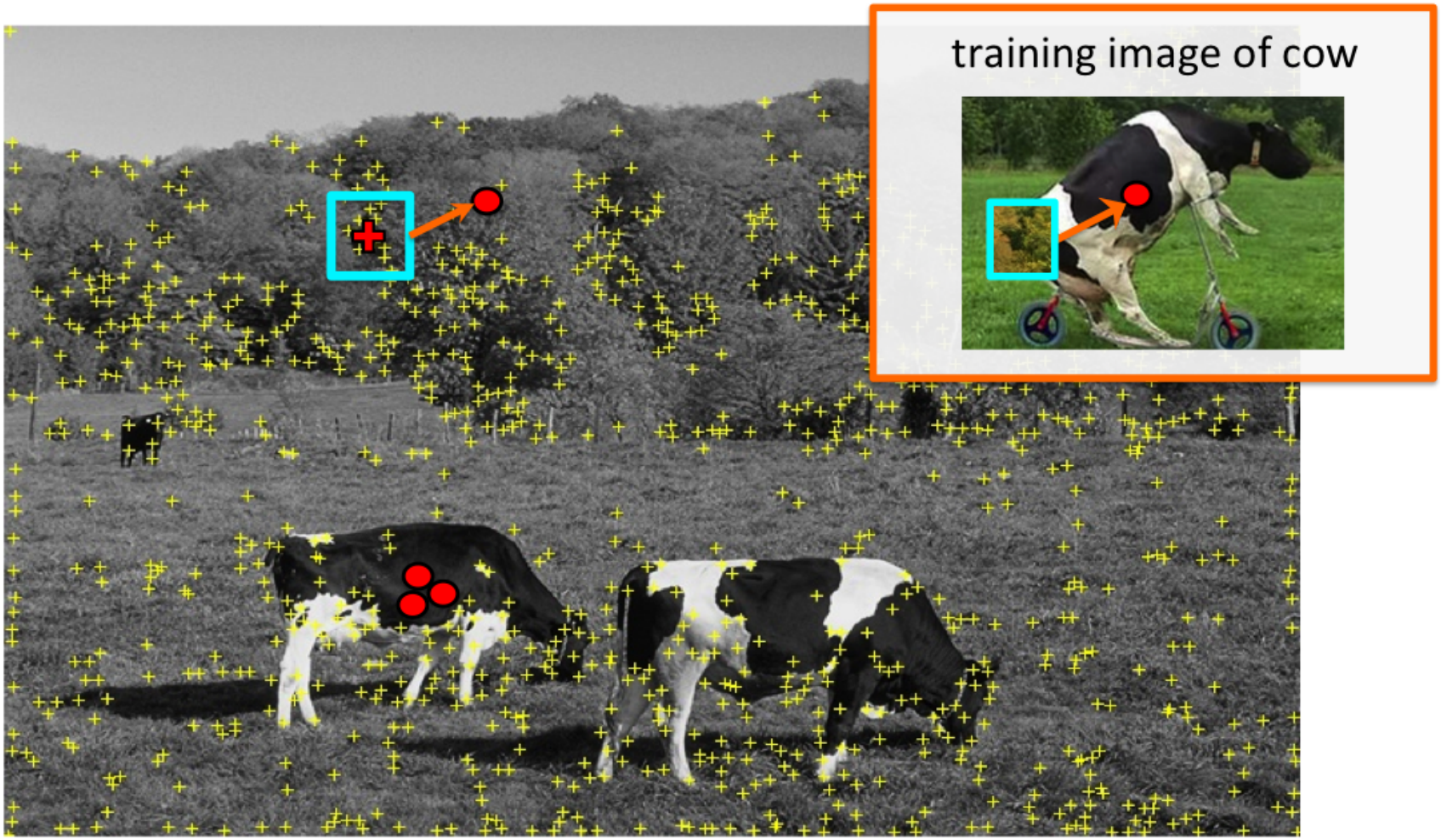
- Vote for object center



vote for center of object

Implicit Shape Model: Basic Idea

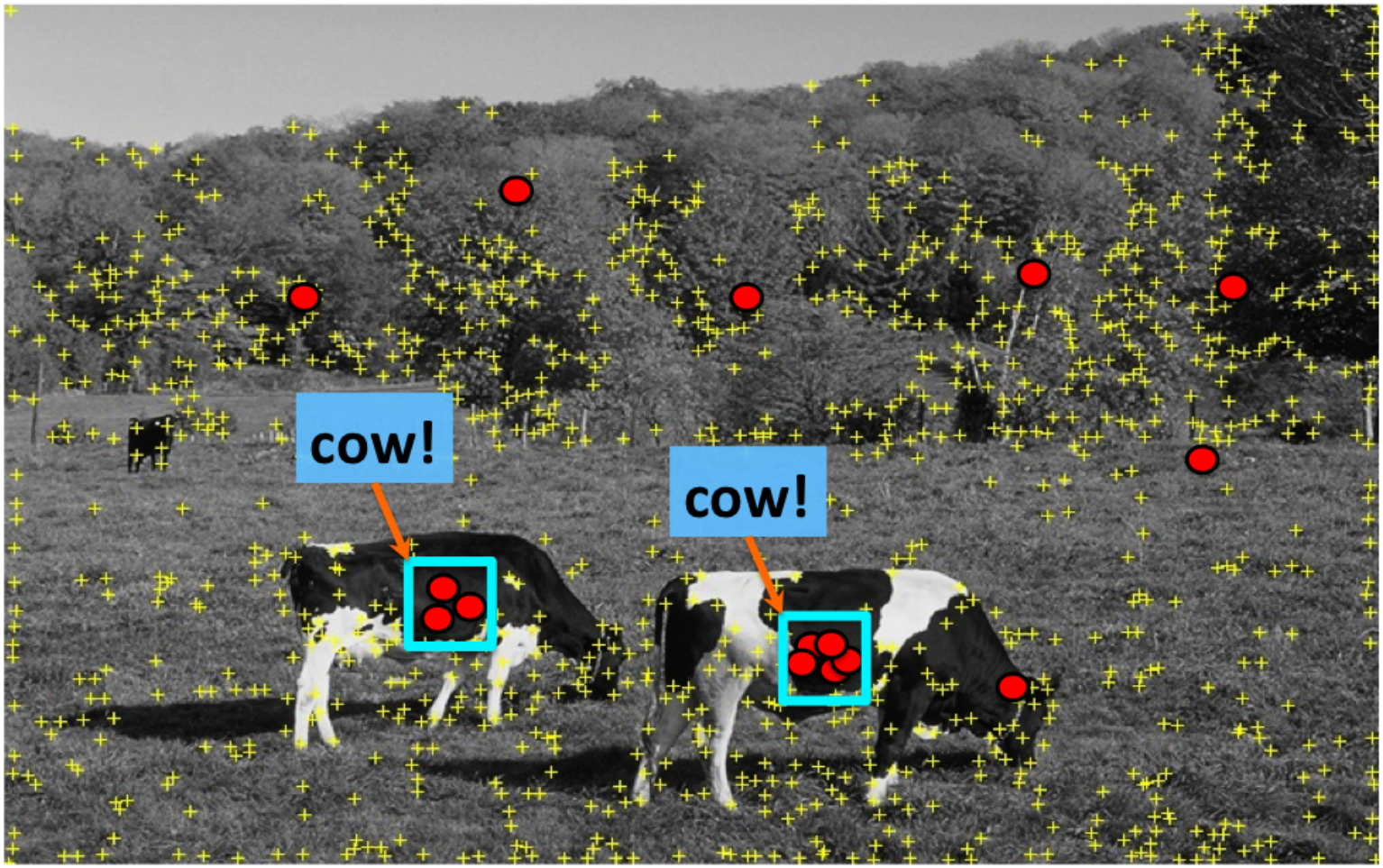
- Vote for object center



of course some wrong votes are bound to happen...

Implicit Shape Model: Basic Idea

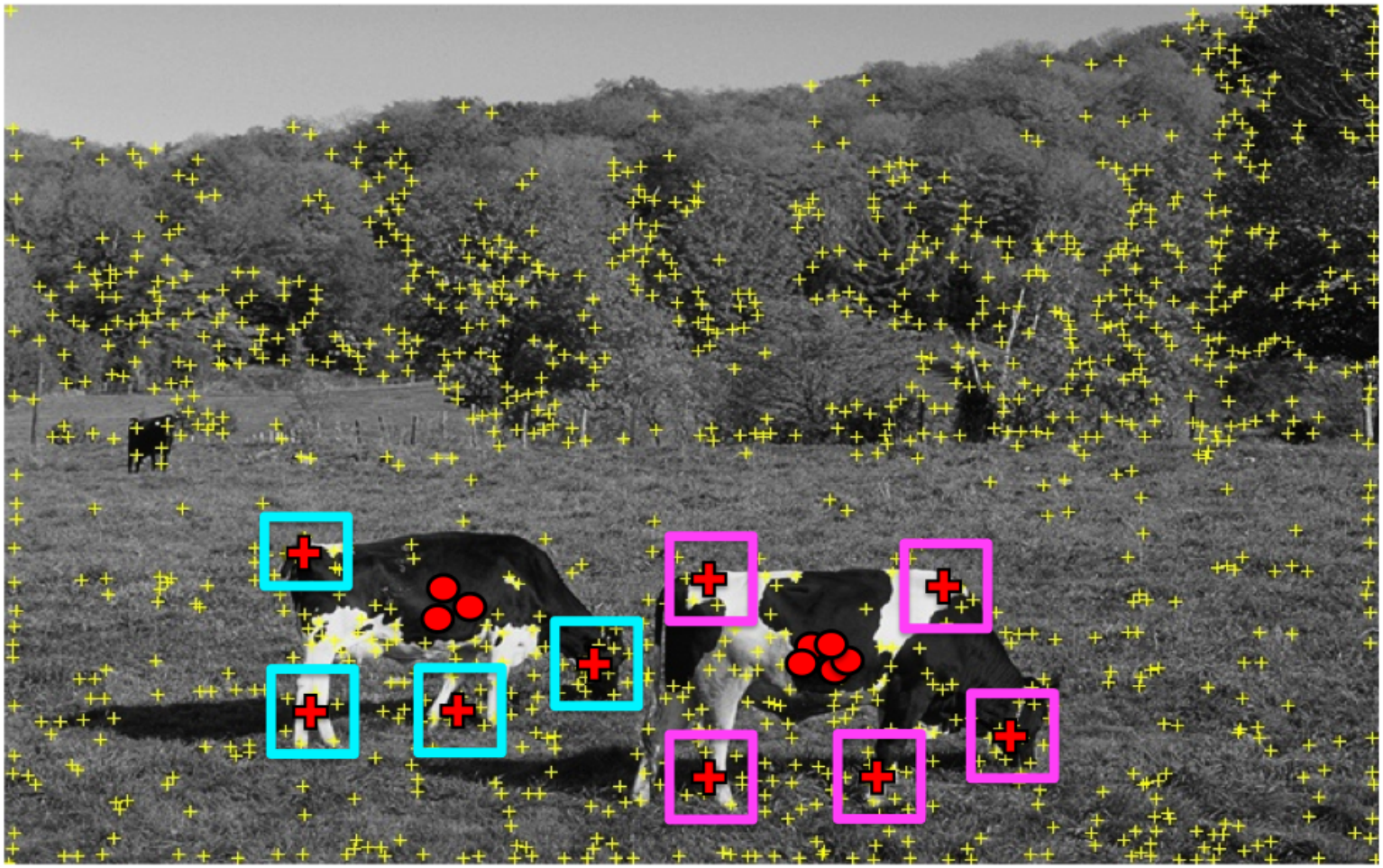
- Vote for object center



But that's ok. We want only **peaks** in voting space.

Implicit Shape Model: Basic Idea

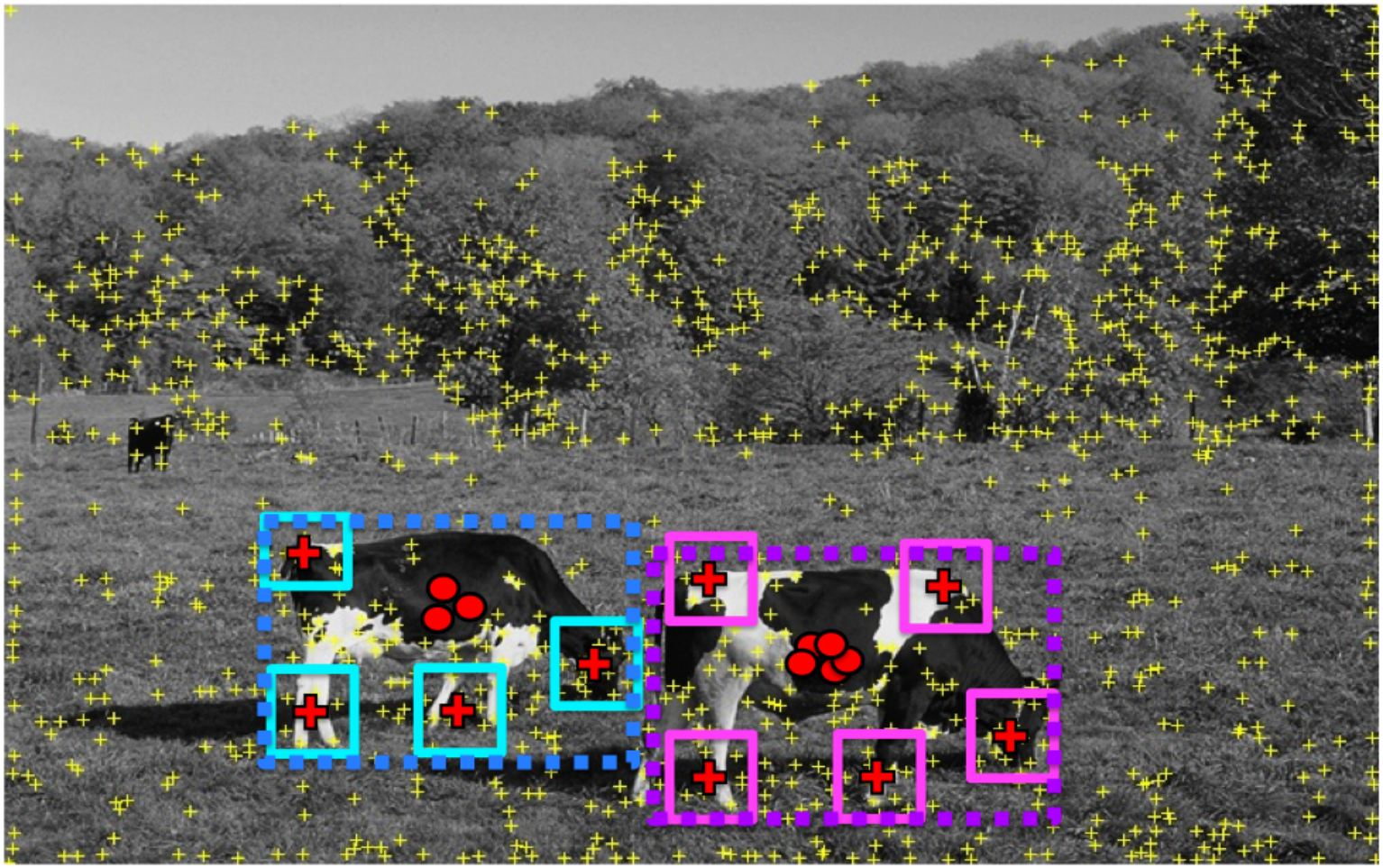
- Find the patches that produced the peak



Find patches that voted for the peaks (back-projection).

Implicit Shape Model: Basic Idea

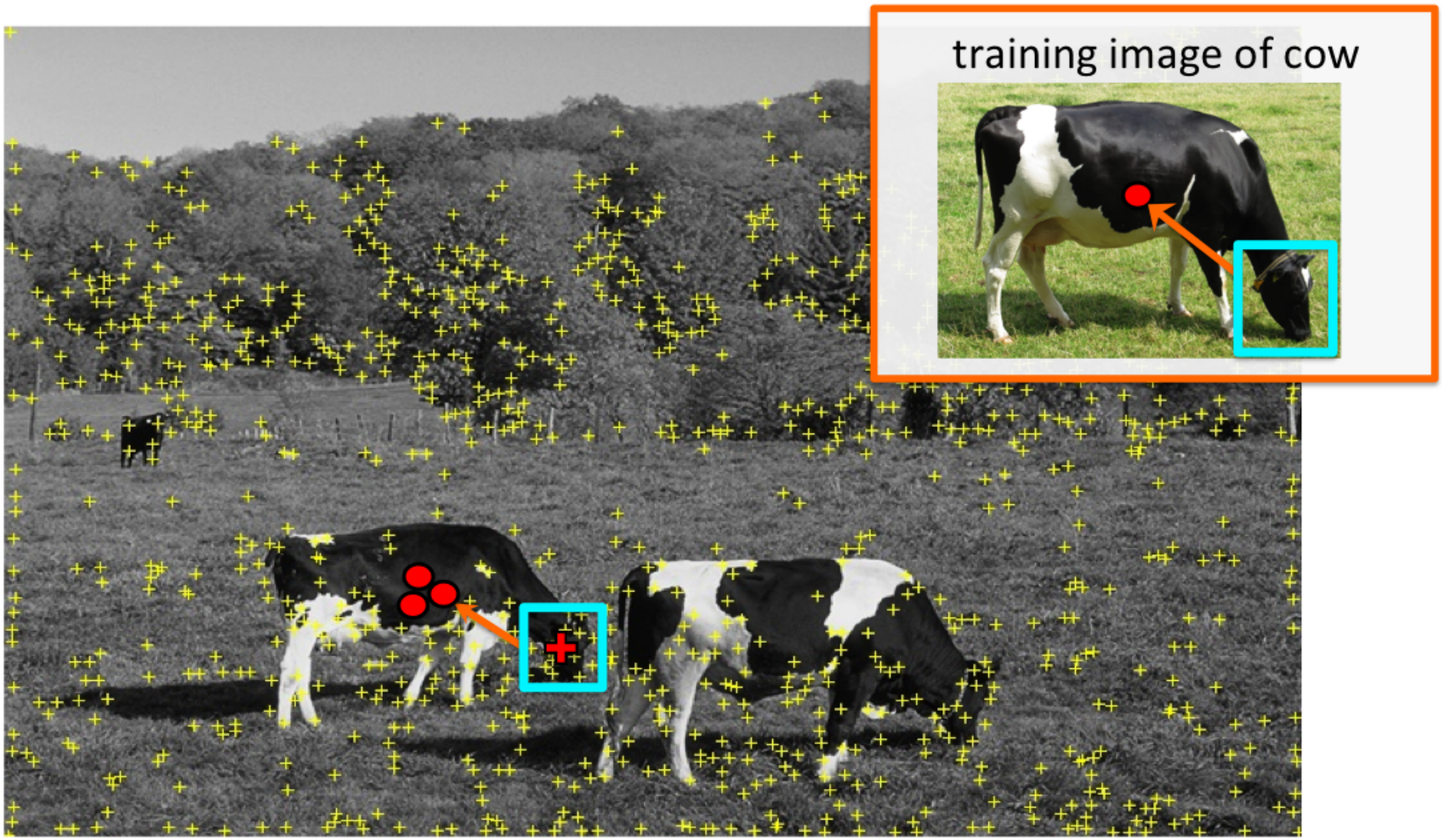
- Place a box around these patches $\rightarrow$ objects!



Find full objects based on the back-projected patches.

Implicit Shape Model: Basic Idea

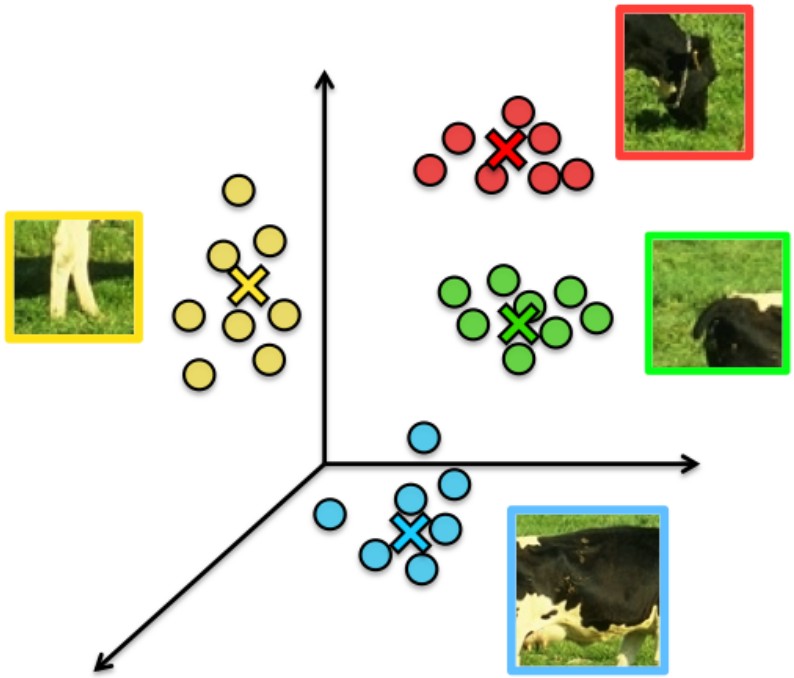
- Really easy. Only one problem... Would be slow... How do we make it fast?



we need to match a patch around each yellow + to all patches in all training images $\rightarrow$ **SLOW**

Implicit Shape Model: Basic Idea

- **Visual vocabulary** (we saw this for retrieval)
- Compare each patch to a small set of visual words (clusters)



Visual words (visual codebook)!

Implicit Shape Model: Basic Idea

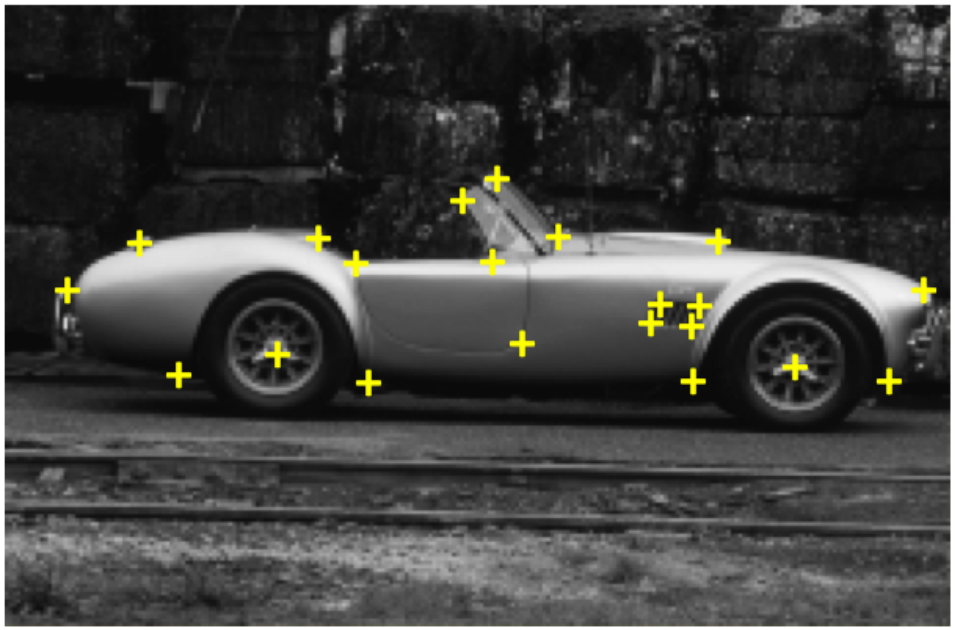
- Training: Getting the vocabulary

training image

Implicit Shape Model: Basic Idea

- Find interest points in each training image

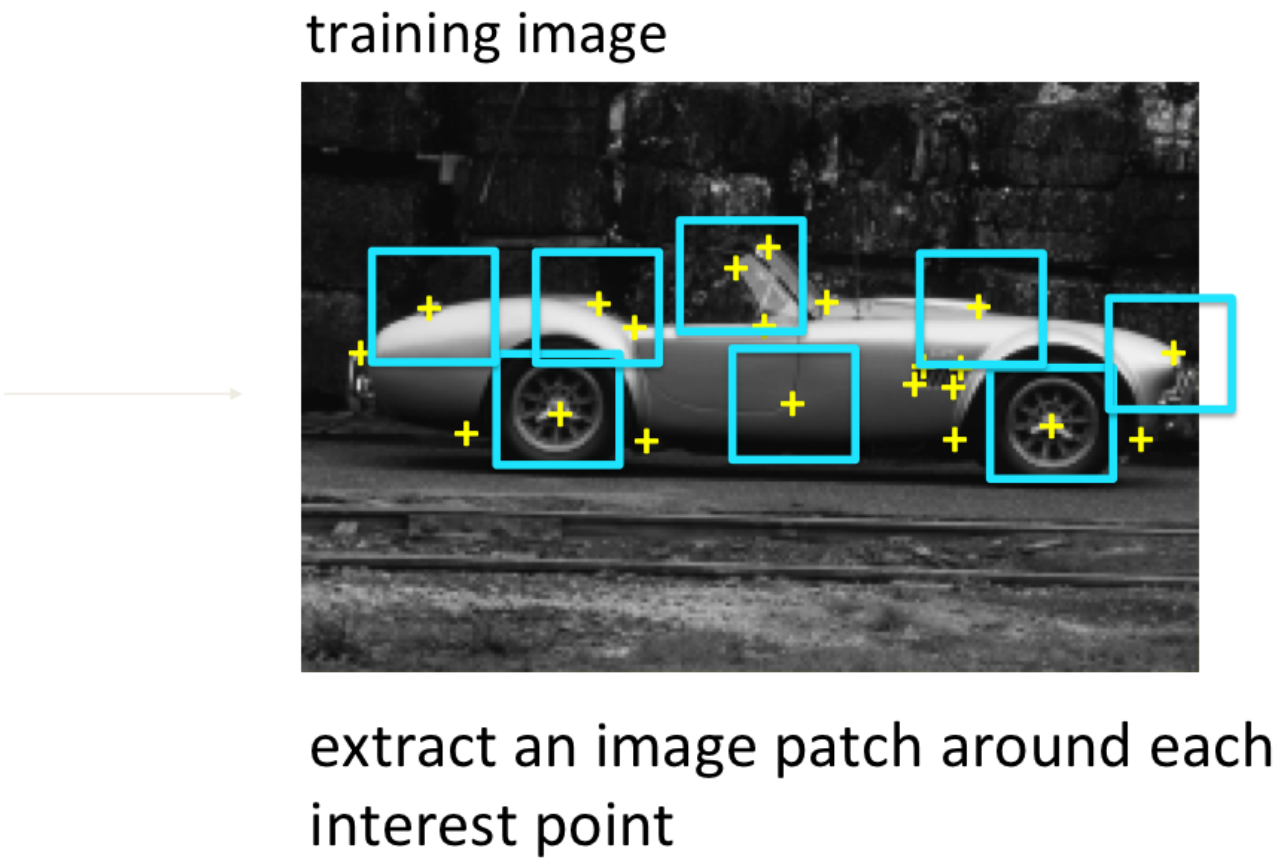
training image



detect interest points (e.g. Harris)

Implicit Shape Model: Basic Idea

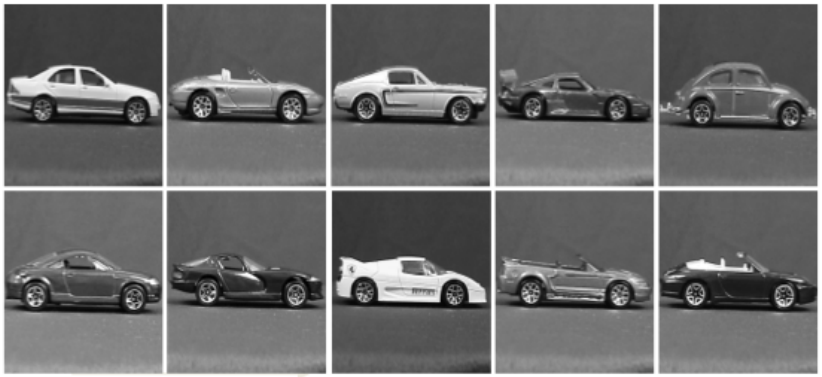
- Collect patches around each interest point



Implicit Shape Model: Basic Idea

- Collect patches across all training examples

training images

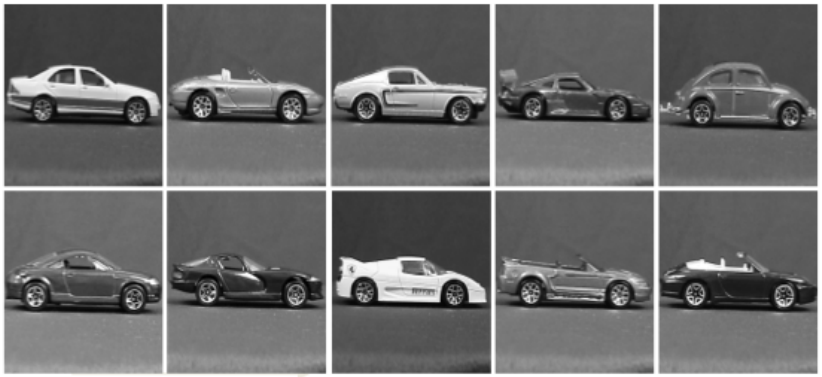


collect all patches

Implicit Shape Model: Basic Idea

- Cluster the patches to get a small set of "representative" patches

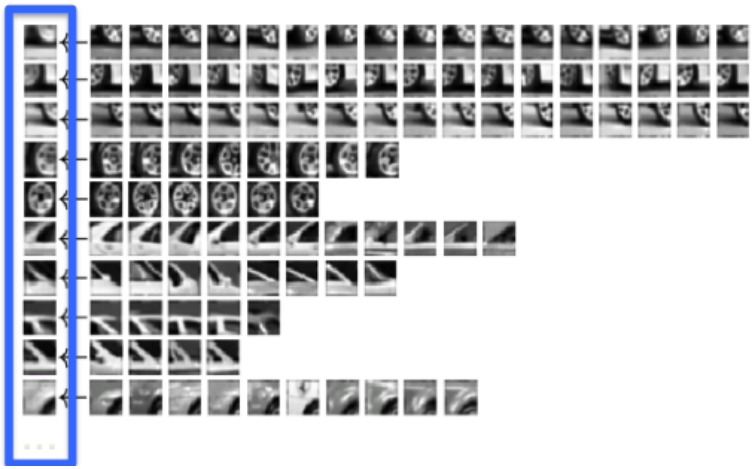
training images



collect all patches

- cluster the patches to get a few ``representative'' patches
- each cluster represented as the average of all patches that belong to the cluster

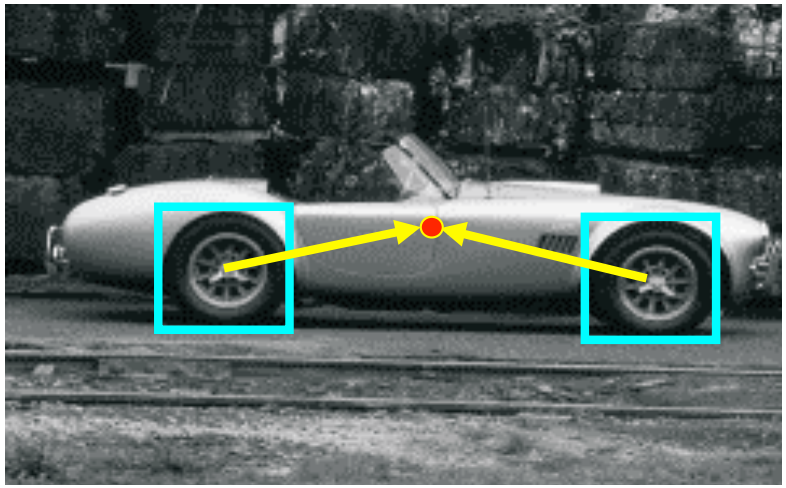
visual codebook



clusters

Implicit Shape Model: Training

- Represent each training patch with the closest visual word.
- Record the displacement vectors for each word across all training examples.



Training image

Visual codeword with displacement vectors

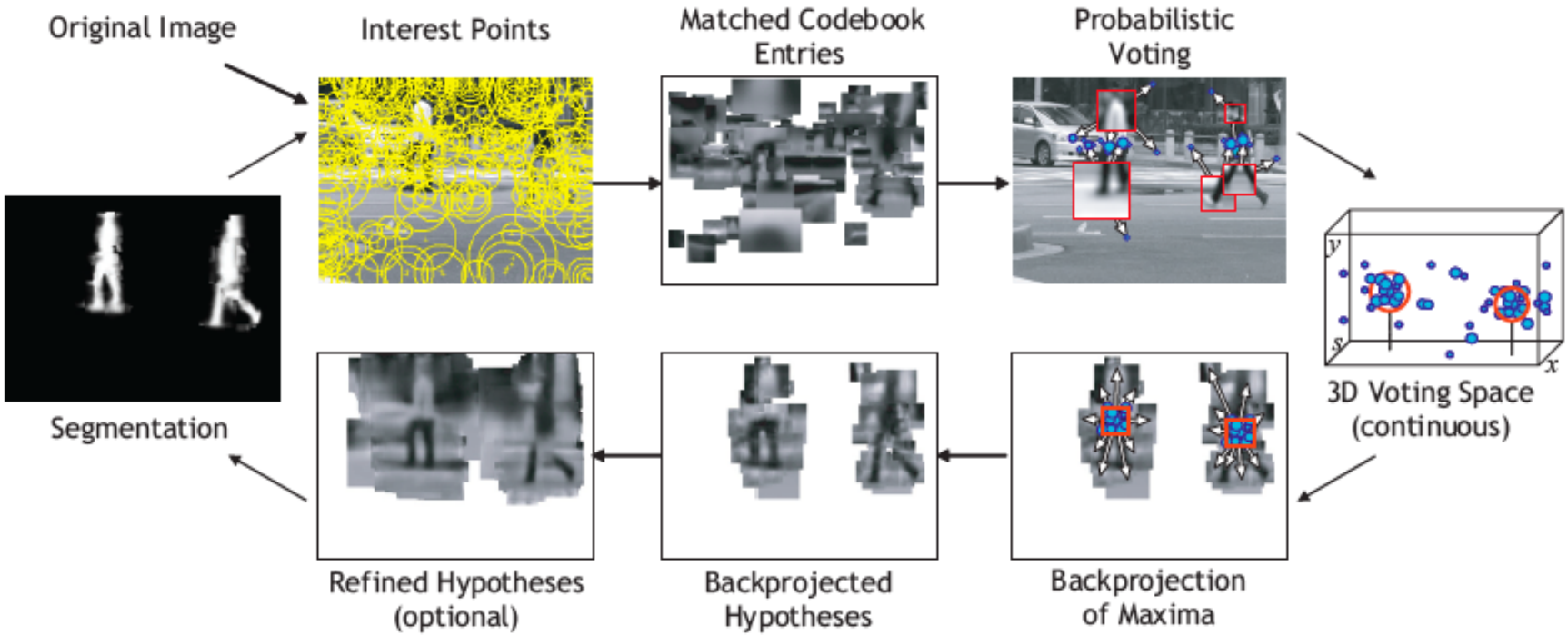
[Leibe et al. IJCV 2008]

Implicit Shape Model: Test

- At test times detect interest points
- Assign each patch around interest point to closes visual word
- Vote with all displacement vectors for that word

[Source: B. Leibe]

Recognition Pipeline



[Source: B. Leibe]

Recognition Summary

- Apply interest points and extract features around selected locations.
- Match those to the codebook.
- Collect consistent configurations using Generalized Hough Transform.
- Each entry votes for a set of possible positions and scales in continuous space.
- Extract maxima in the continuous space using Mean Shift.
- Refinement can be done by sampling more local features.

[Source: R. Urtasun]

Example

Original image

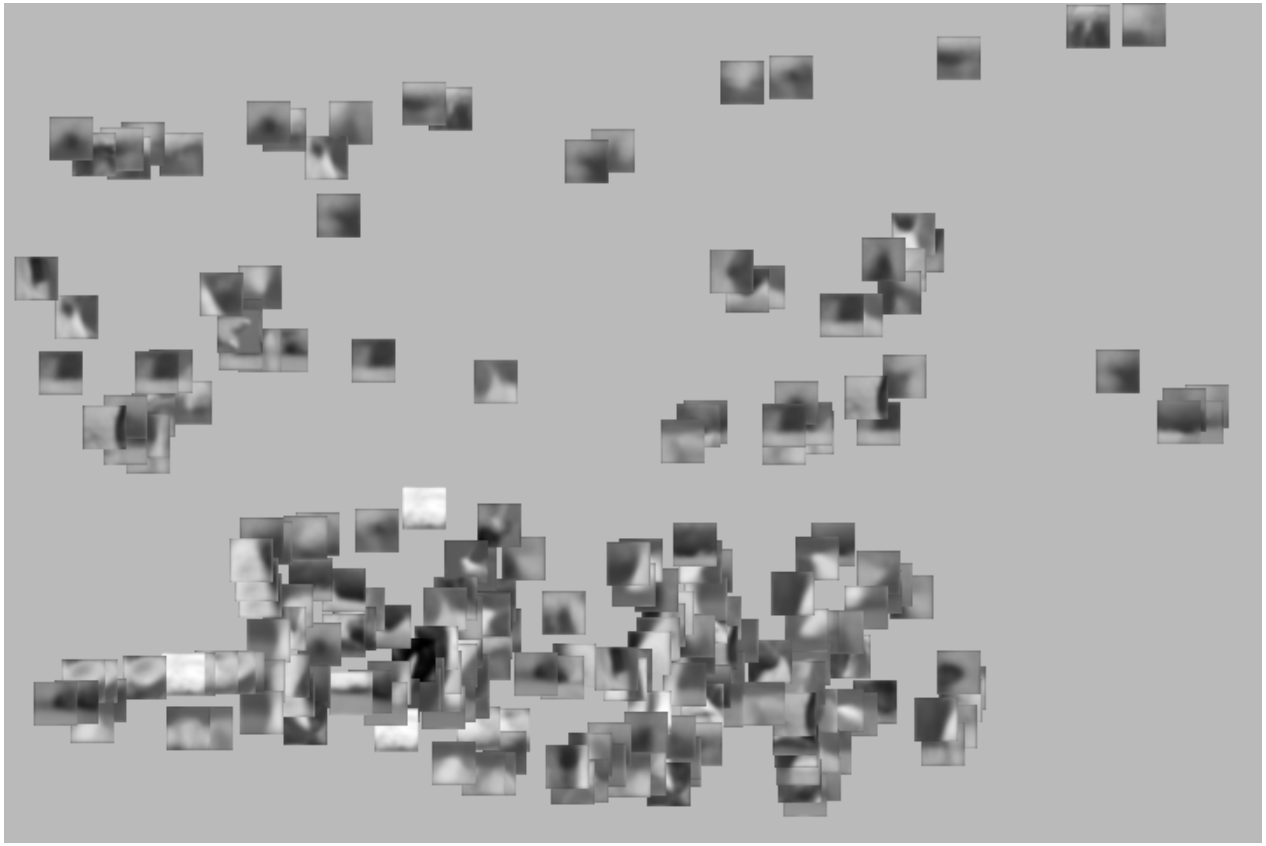
[Source: B. Leibe, credit: R. Urtasun]

Example

Interest points

[Source: B. Leibe, credit: R. Urtasun]

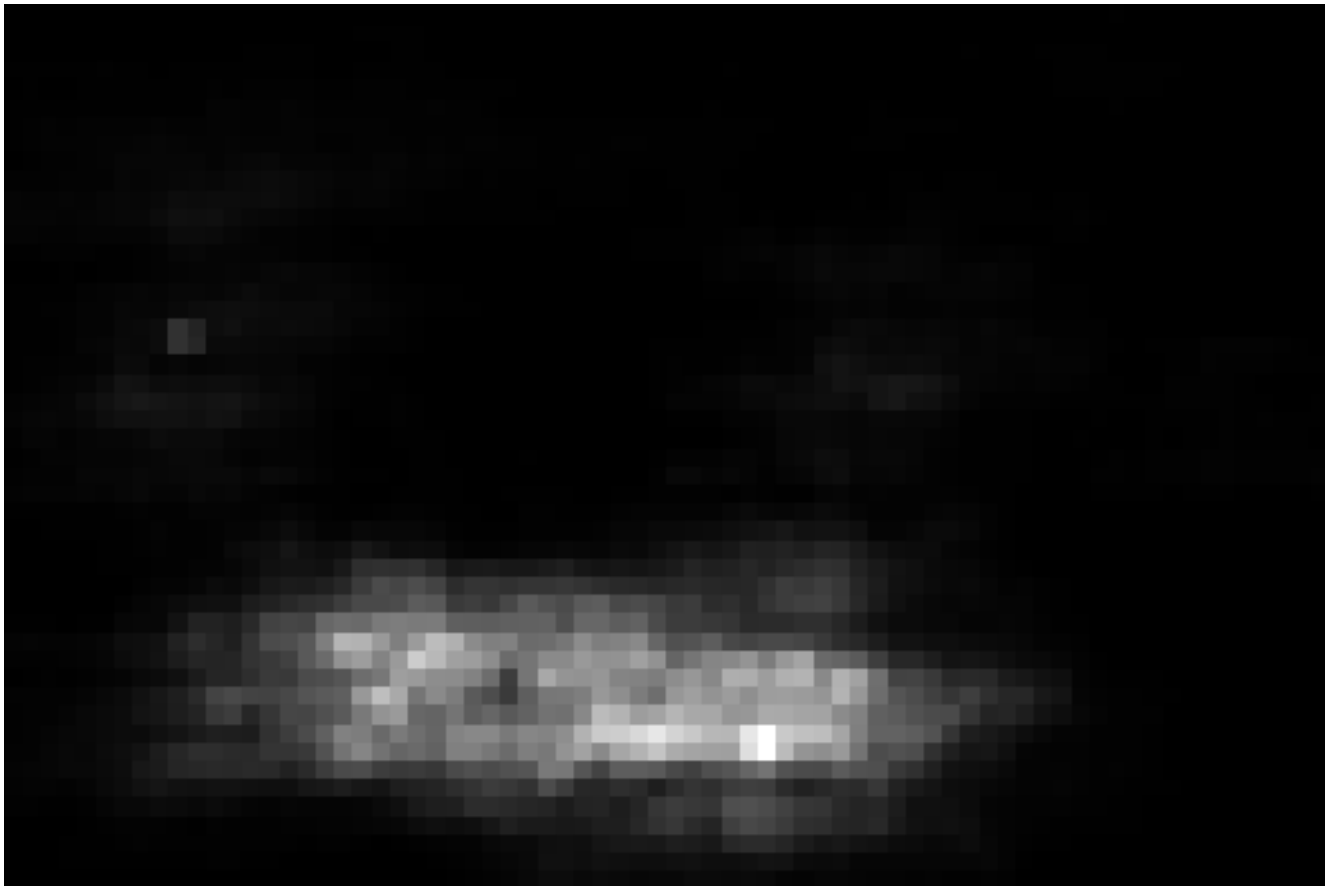
Example



Matched patches

[Source: B. Leibe, credit: R. Urtasun]

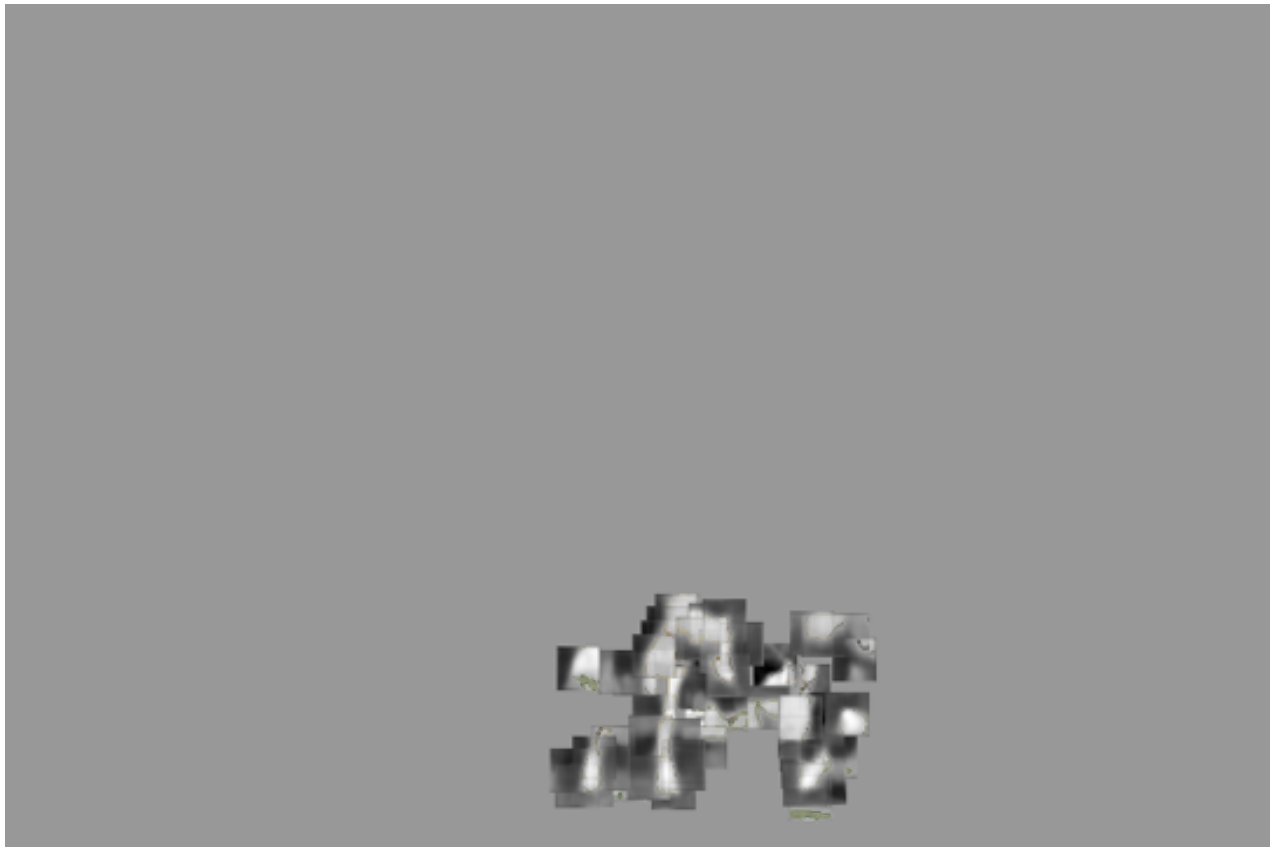
Example



Voting space

[Source: B. Leibe, credit: R. Urtasun]

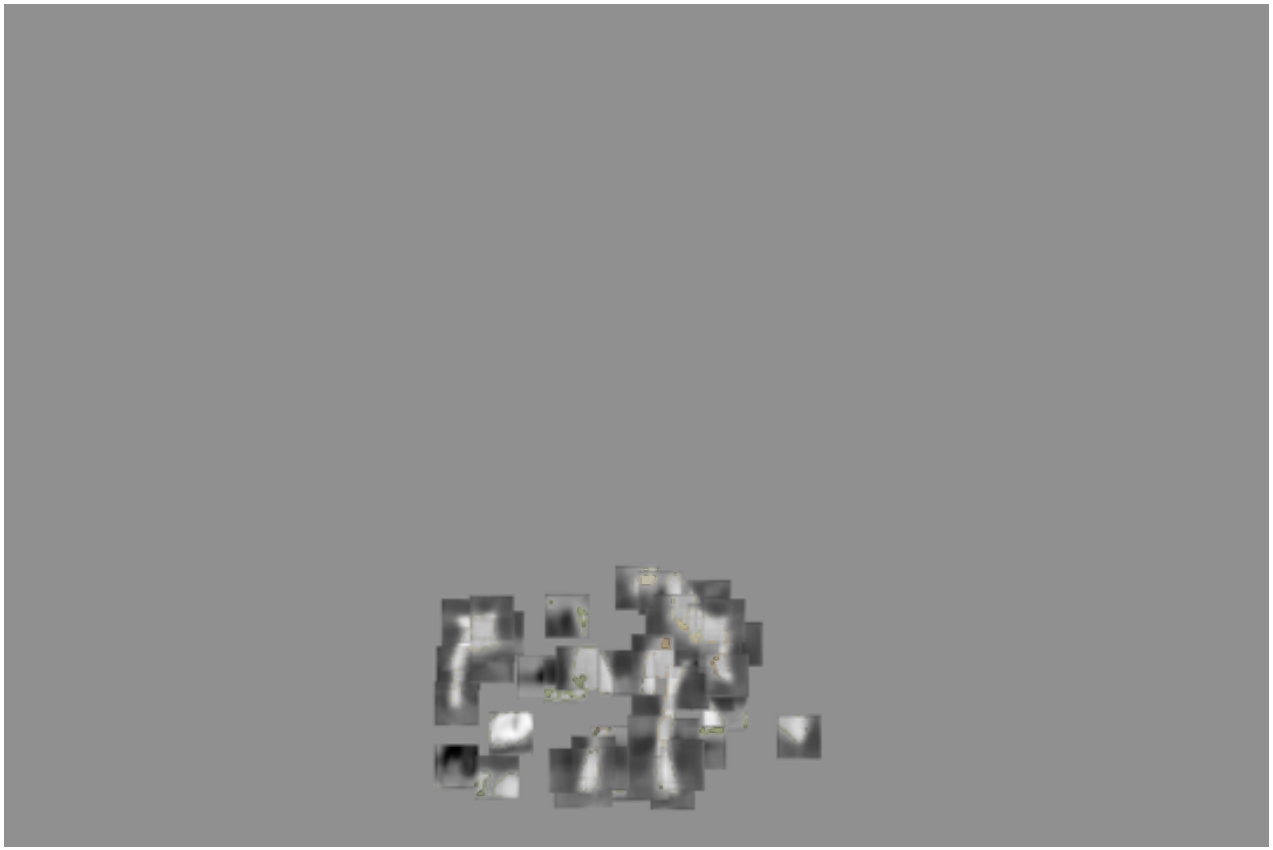
Example



1st hypothesis

[Source: B. Leibe, credit: R. Urtasun]

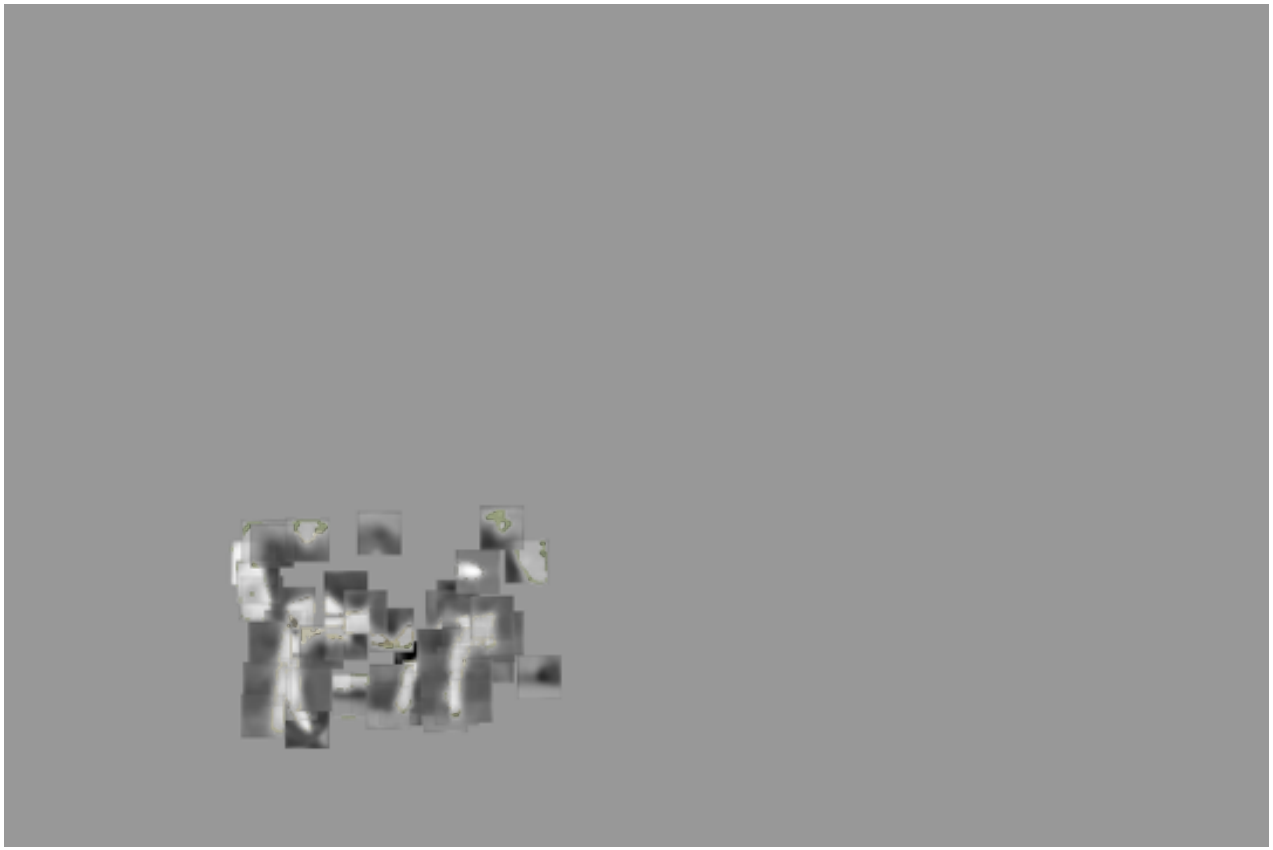
Example



2nd hypothesis

[Source: B. Leibe, credit: R. Urtasun]

Example



3rd hypothesis

[Source: B. Leibe, credit: R. Urtasun]

Scale Invariant Voting

Scale-invariant feature selection

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

Generate scale votes

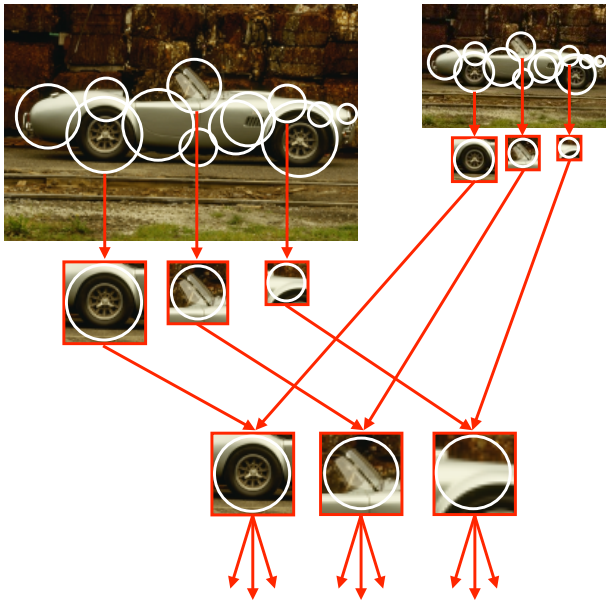
- Scale as 3rd dimension in voting space

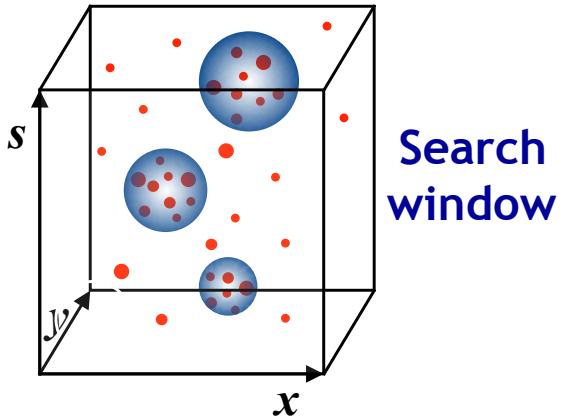
$$
\begin{aligned}
x_{vote} &= x_{img} - x_{occ}(s_{img}/s_{occ}) \\
y_{vote} &= y_{img} - y_{occ}(s_{img}/s_{occ}) \\
s_{vote} &= s_{img}/s_{occ}
\end{aligned}
$$

- Search for maxima in 3D voting space

[Source: B. Leibe, credit: R. Urtasun]

Scale Invariant Voting



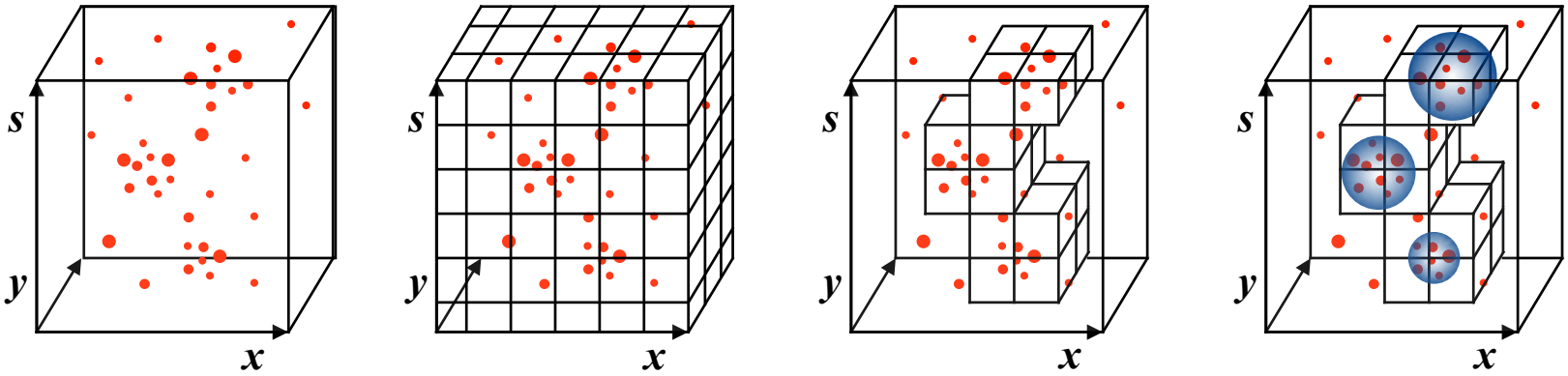


[Slide credit: B. Leibe]

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.

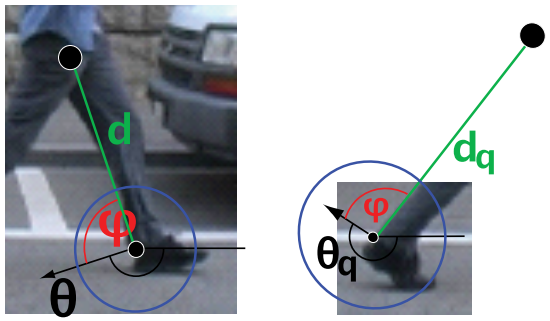


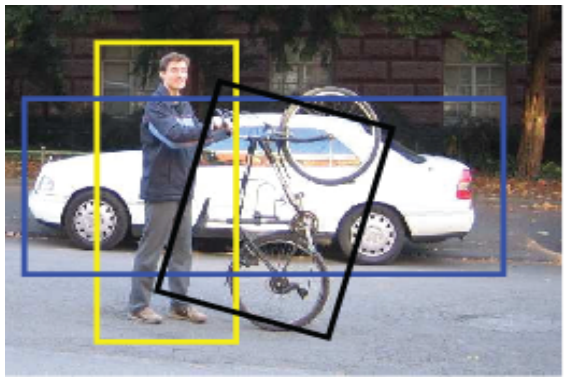
Scale votes | **Binned accum. array** | **Candidate maxima** | **Refinement (Mean-Shift)**

[Source: B. Leibe, credit: R. Urtasun]

Extension: Rotation-Invariant Detection

- Polar instead of Cartesian voting scheme
- Recognize objects under image-plane rotations
- Possibility to share parts between articulations
- But also increases false positive detections





[Source: B. Leibe, credit: R. Urtasun]

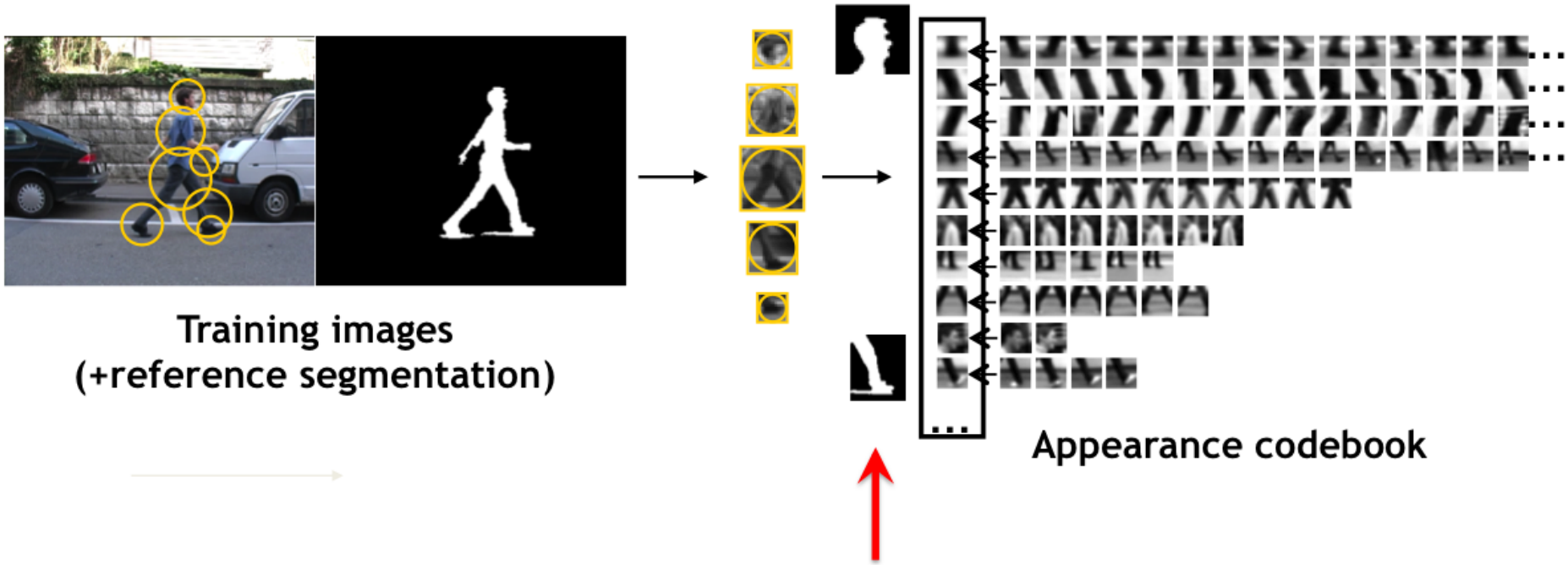
Sometimes it's Necessary

Figure from [Mikolajczyk et al., CVPR'06]

B. Leibe

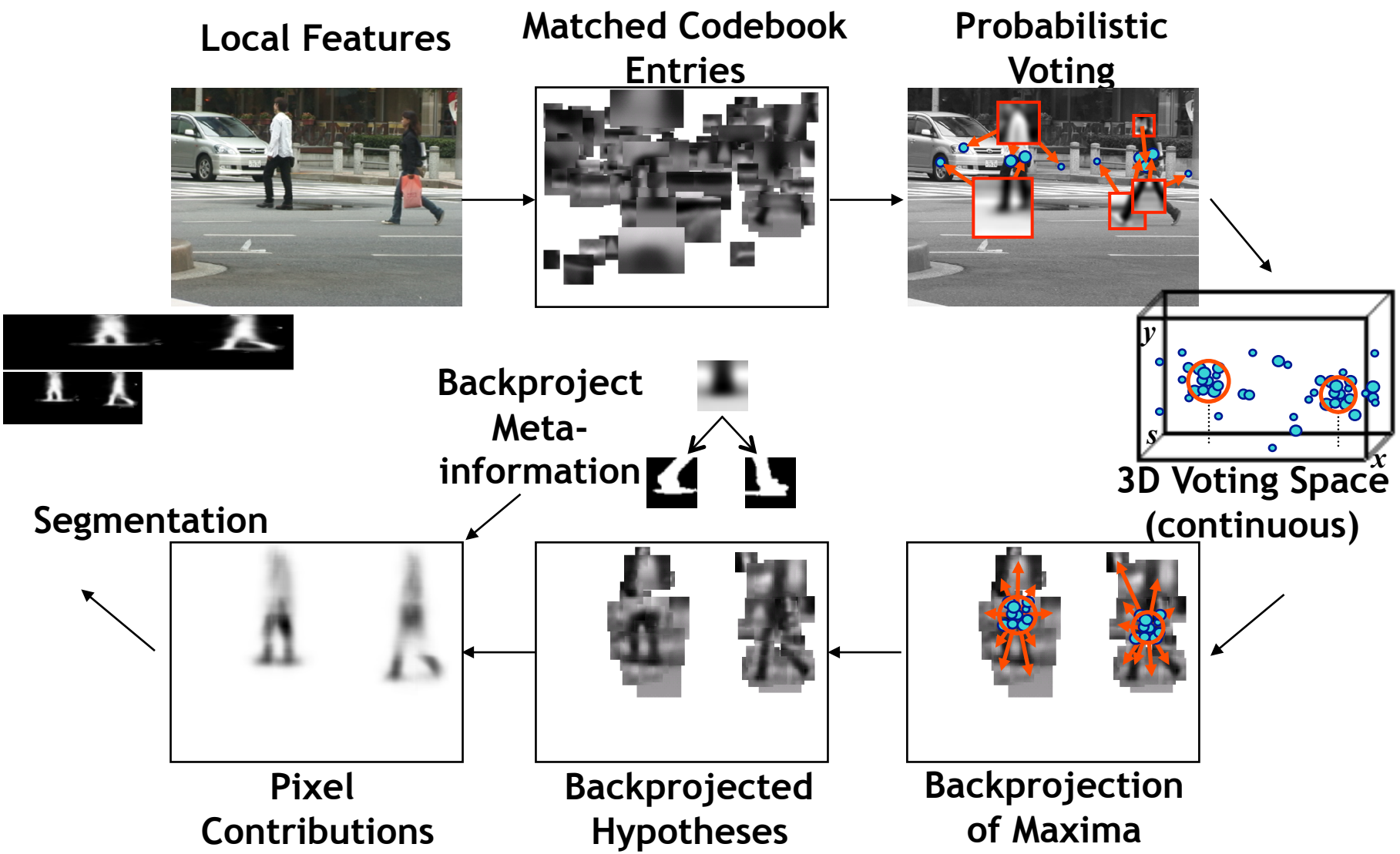
[Source: B. Leibe, credit: R. Urtasun]

Recognition and Segmentation

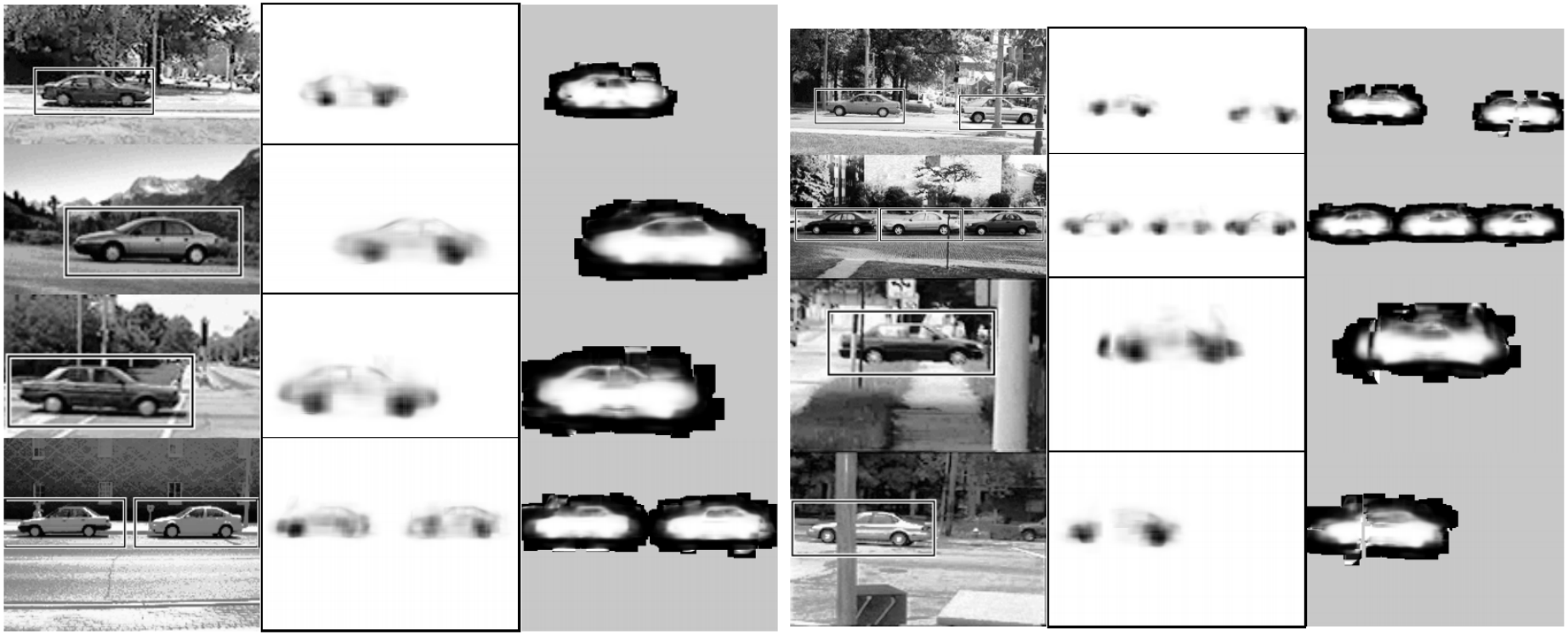


- Augment each visual word with meta-deta: for example, segmentation mask

Recognition and Segmentation



Results

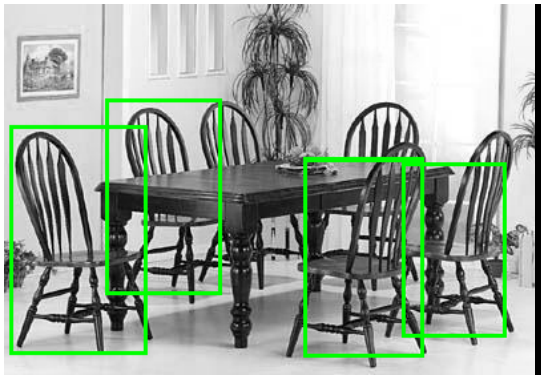


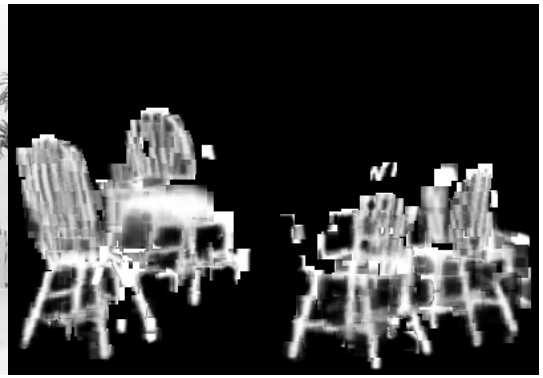
[Source: B. Leibe]

Results

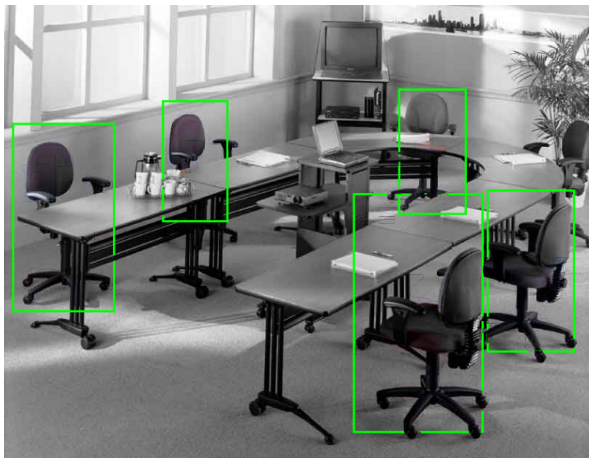
[Source: B. Leibe]

Results

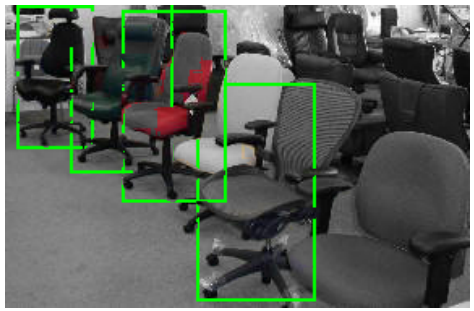


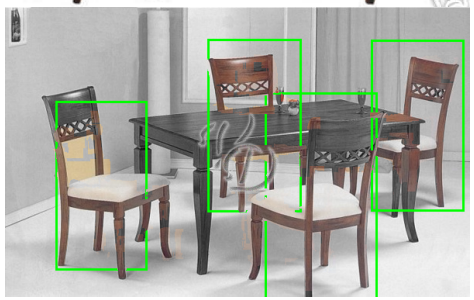


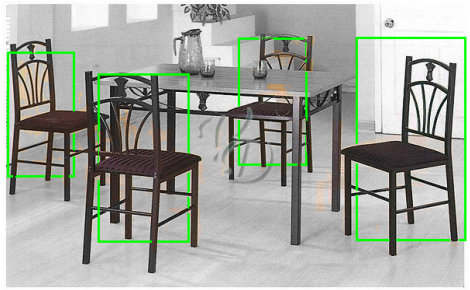
Dining room chairs



Office chairs





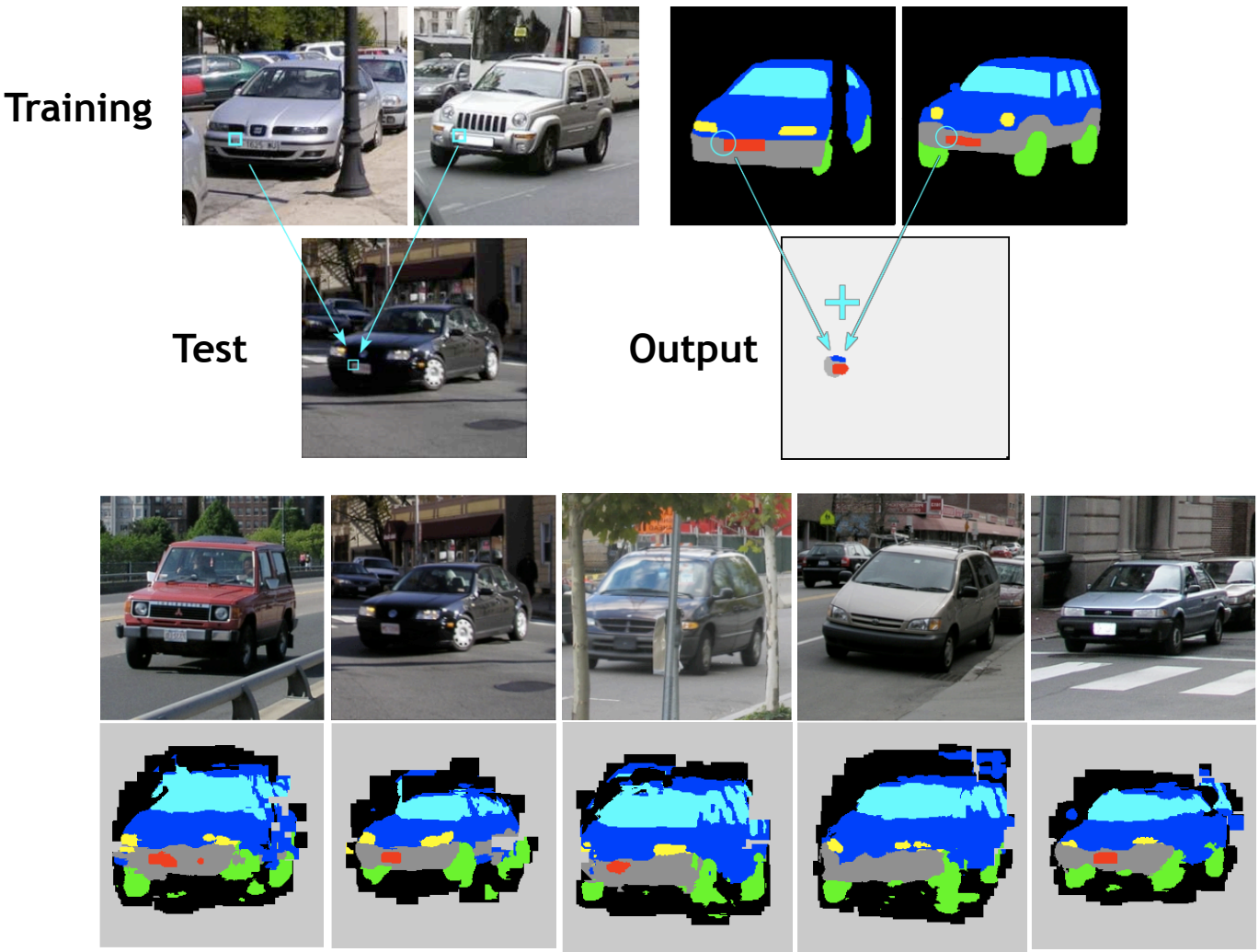


[Source: B. Leibe]

Results

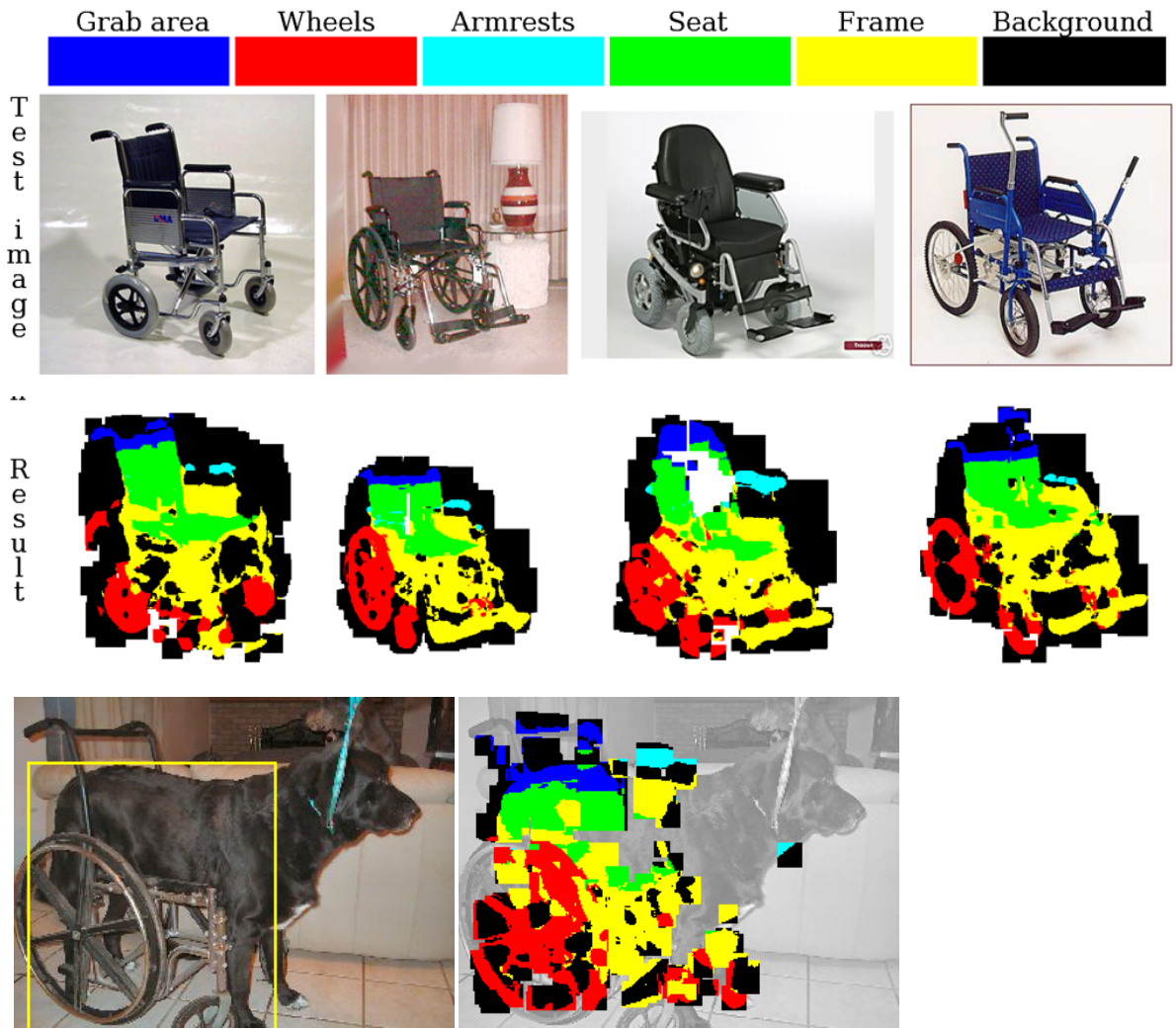
[Source: B. Leibe]

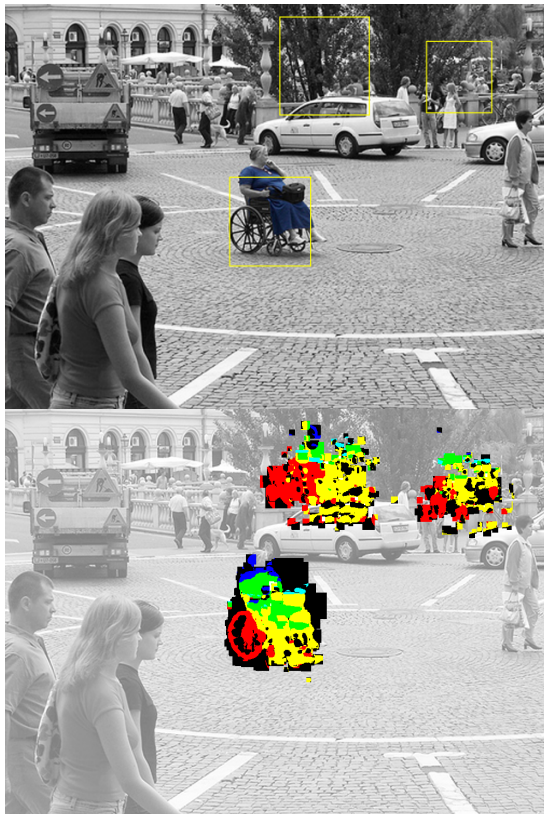
Inferring Other Information: Part Labels



[Source: B. Leibe]

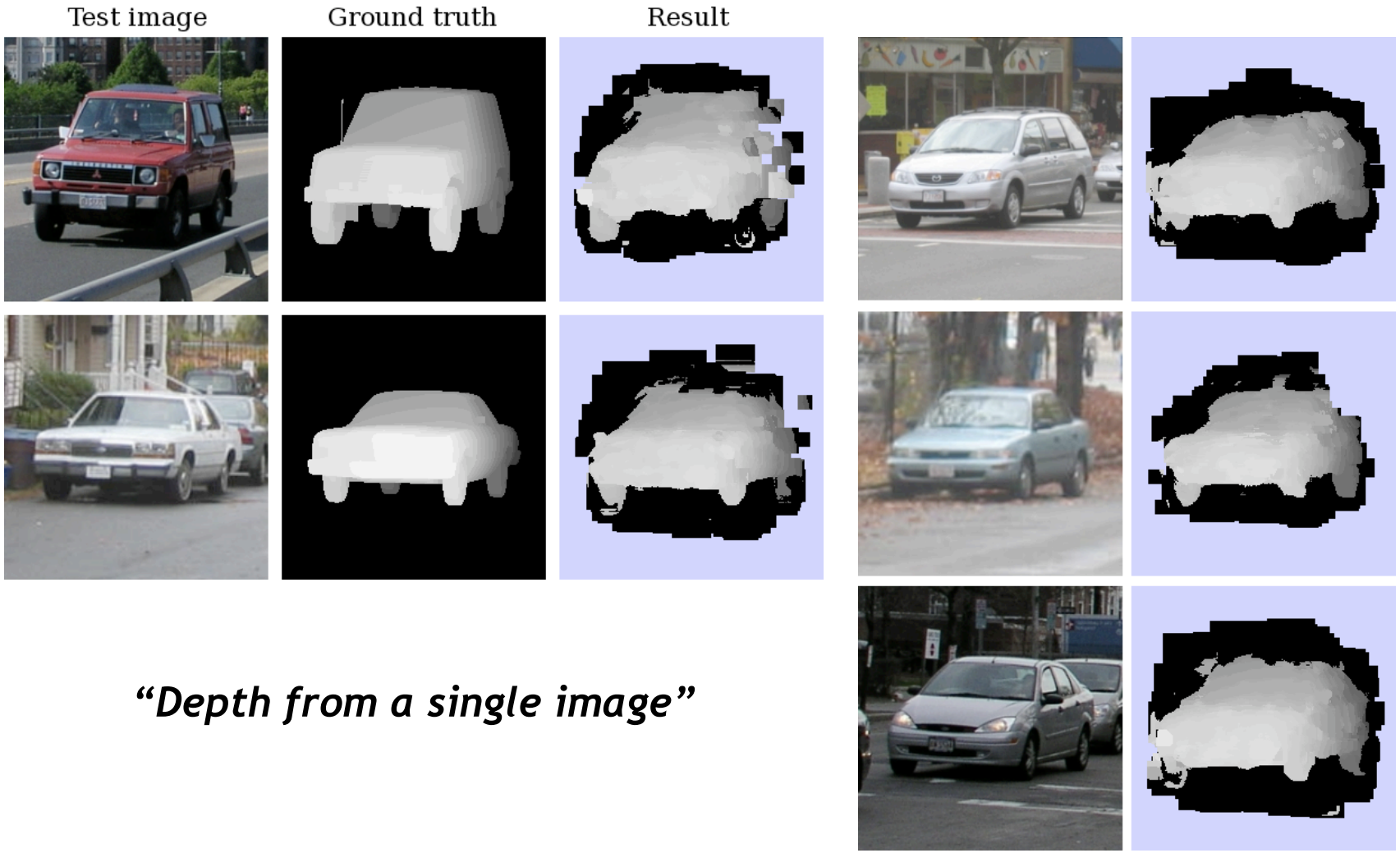
Inferring Other Information: Part Labels





[Source: B. Leibe]

Inferring Other Information: Depth



"Depth from a single image"

[Source: B. Leibe]

Conclusion

- Exploits a lot of parts (as many as interest points)
- Very simple Voting scheme: Generalized Hough Transform
- Works well, but not as well as Deformable Part-based Models with latent SVM training (next time)
- Extensions: train the weights discriminatively.
- Code, datasets & several pre-trained detectors available at `http://www.vision.ee.ethz.ch/bleibe/code`

[Source: B. Leibe, credit: R. Urtasun]