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

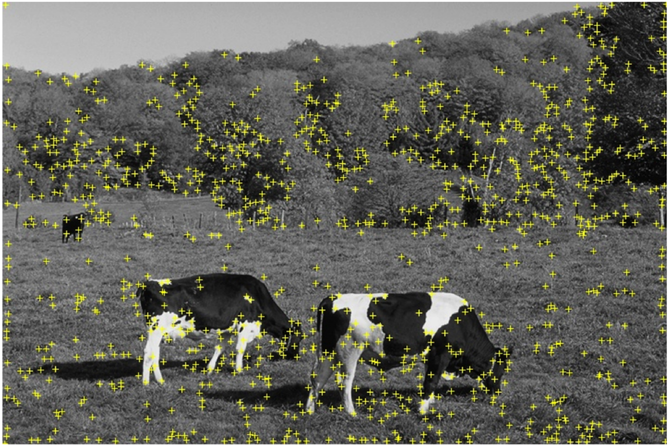
Type of Approaches

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

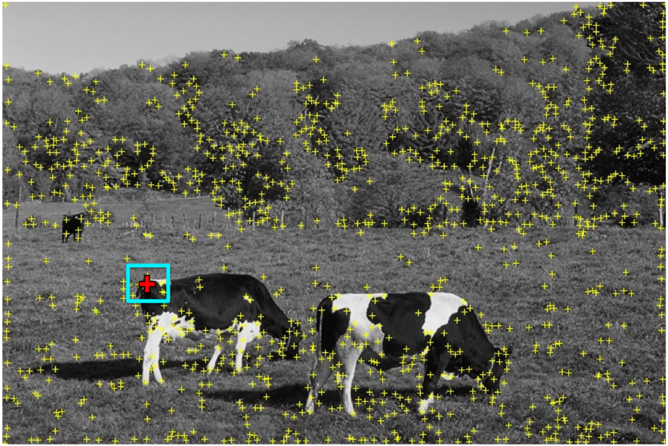
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Interest Point Based Approaches

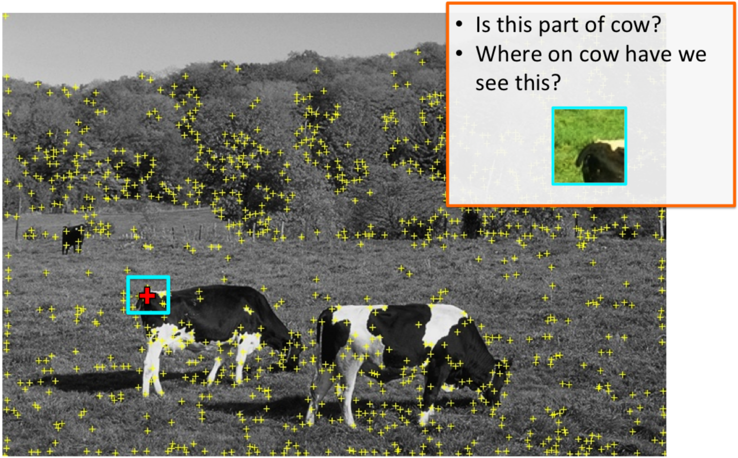
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Interest Point Based Approaches

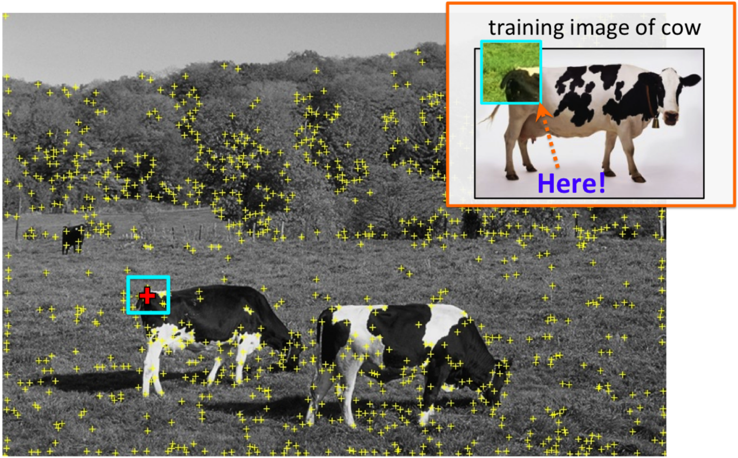
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



Interest points

Interest Point Based Approaches

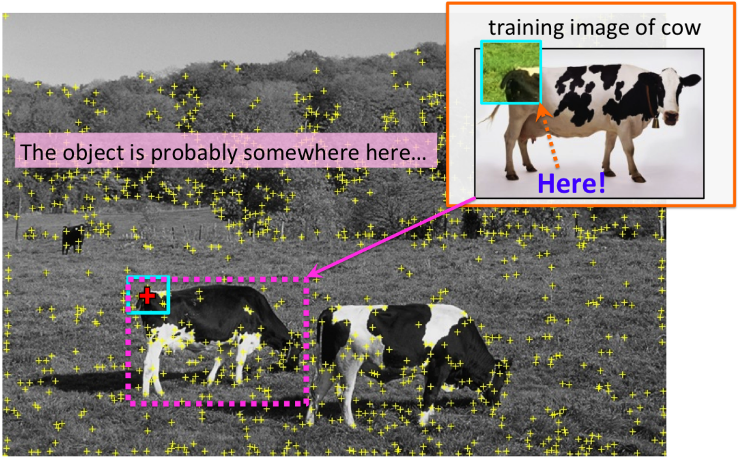
- Compute interest points (e.g., Harris corner detector is a popular choice)
- Vote for where the object could be given the content around interest points



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



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

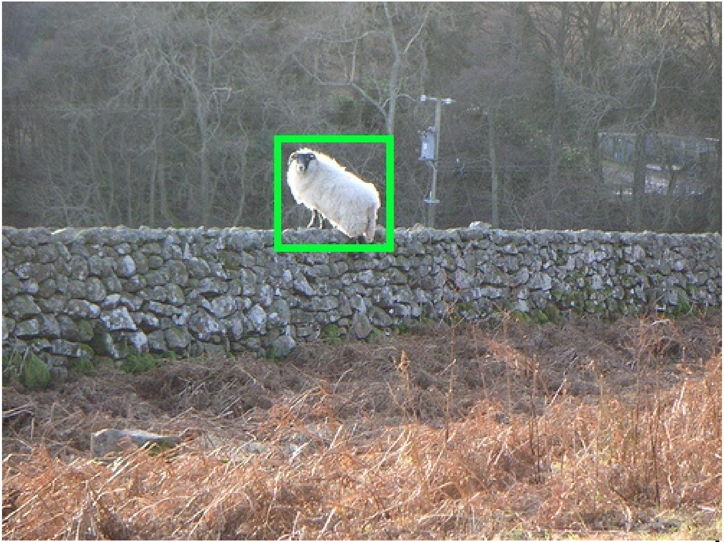
- Slide window and ask a classifier: "Is sheep in window or not?"

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

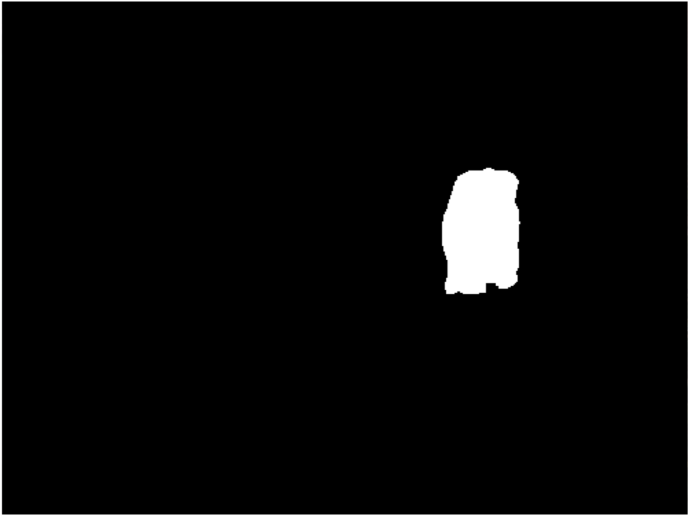
- Generate **many** different regions

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

- 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 $\leftarrow$ **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

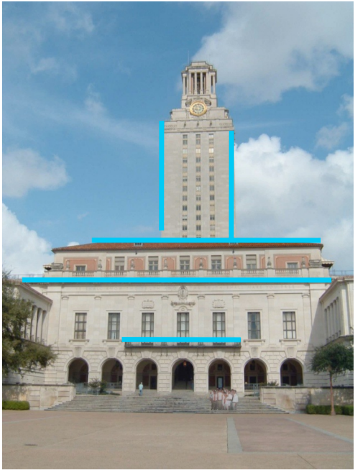
Robust Object Detection with Interleaved Categorization and Segmentation

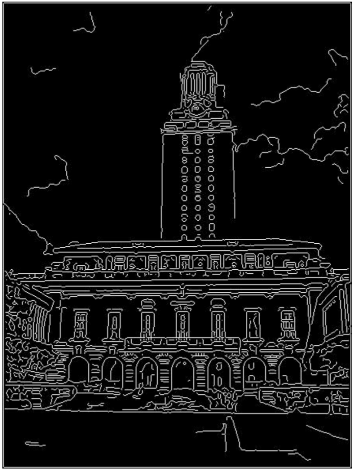
IJCV, 2008

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]

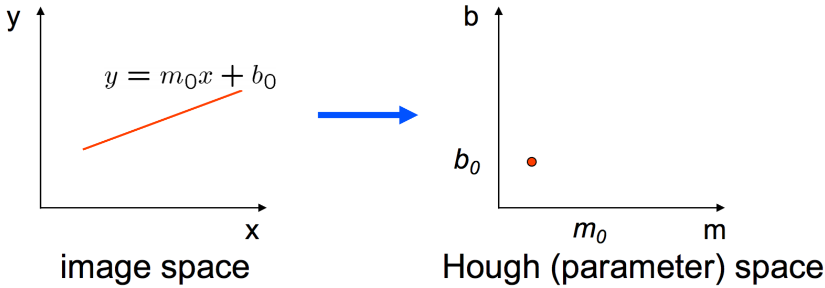
Hough Transform

- 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

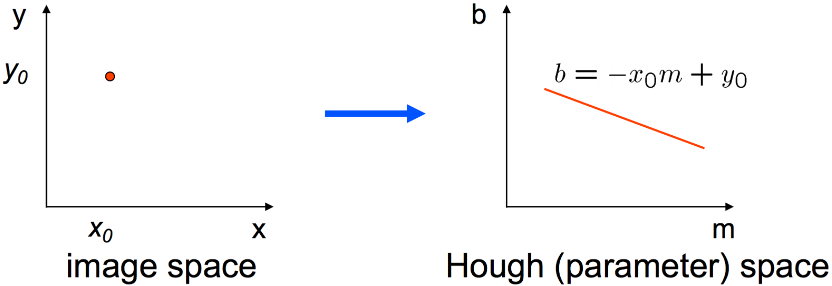
- Hough space: parameter space



- 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]

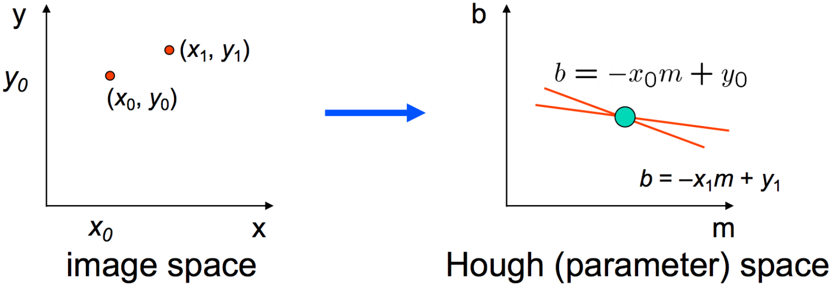
- Hough space: parameter space



- 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 space: parameter space

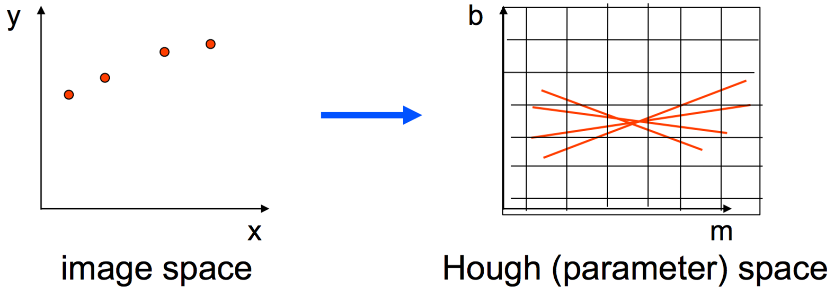


- Two points: Each point corresponds to a line in the Hough space
- A point where these two lines meet defines a line in the image!

[Source: S. Seitz]

Hough Transform: Line Detection

- Hough space: parameter space

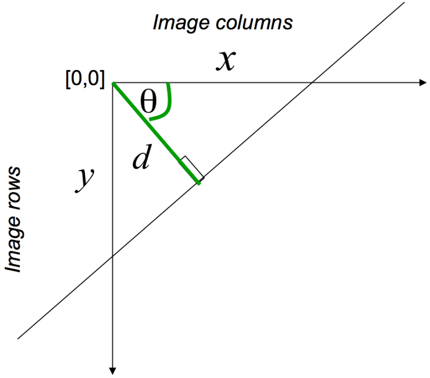


- Vote with each image point
- Find peaks in Hough space. Each peak is a line in the image.

[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

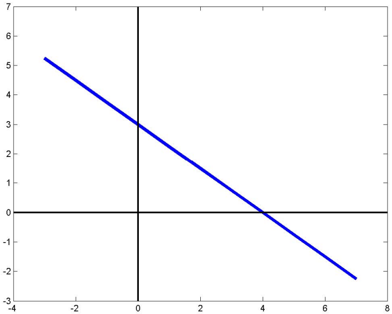
$$x \cos\theta - y \sin\theta = d$$

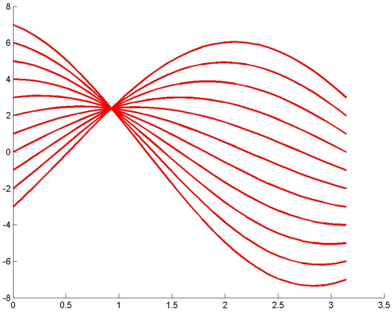
Point in image space → sinusoid segment in Hough space

[Source: S. Seitz]

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





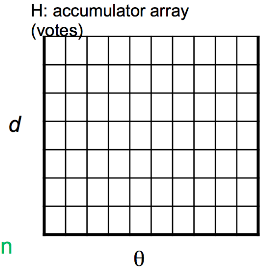
[Source: M. Kazhdan, slide credit: R. Urtasun]

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

[Source: S. Seitz]

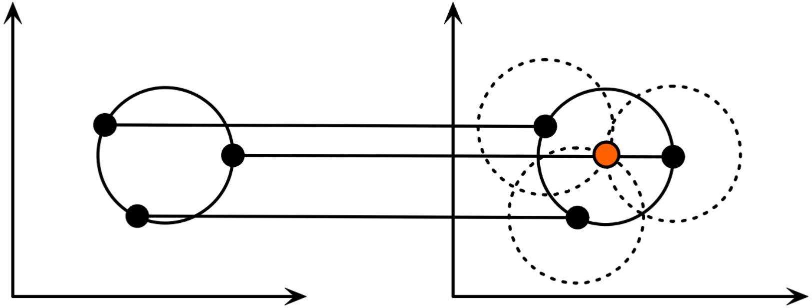
Hough Transform: Circle Detection

- 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



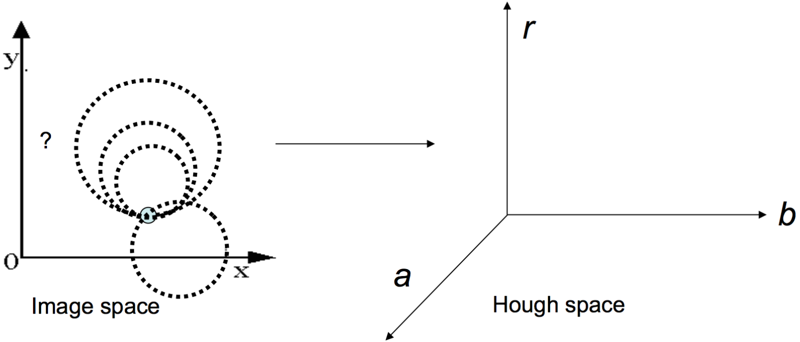
Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

[Source: H. Rhody]

Hough Transform: Circle Detection

What if we don't know r?

- Hough space: ?

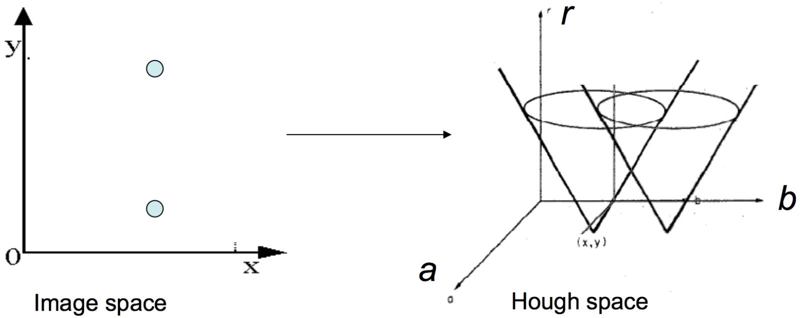


[Source: K. Grauman]

Hough Transform: Circle Detection

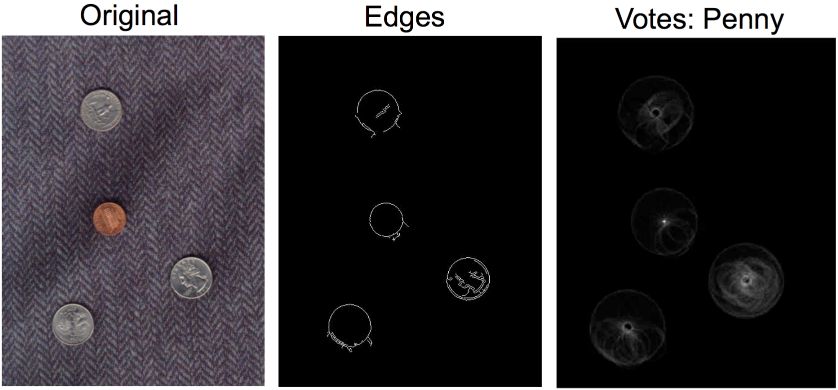
What if we don't know r?

- Hough space: conics



[Source: K. Grauman]

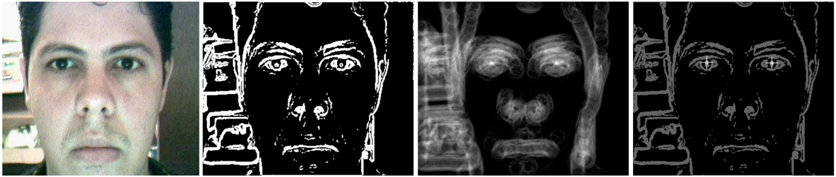
- Find the coins



[Source: K. Grauman]

Hough Transform: Circle Detection

- Iris detection

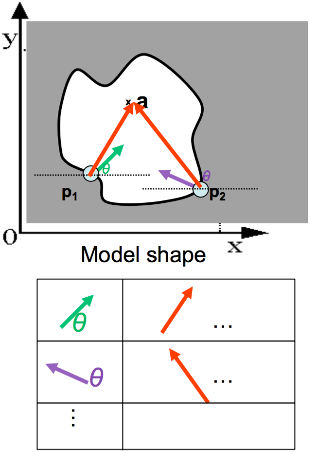


Gradient+threshold | Hough space (fixed radius) | Max detections

[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.

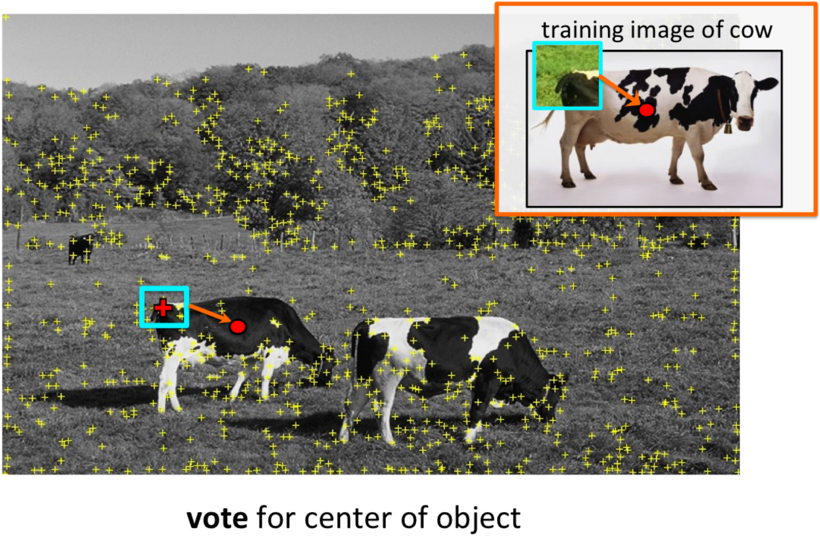
Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980

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

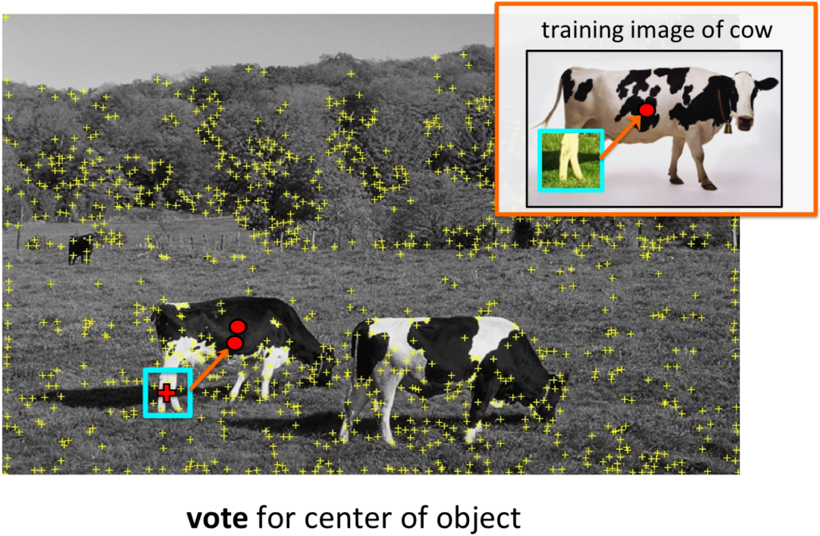
Implicit Shape Model: Basic Idea

- Vote for object center



Implicit Shape Model: Basic Idea

- Vote for object center



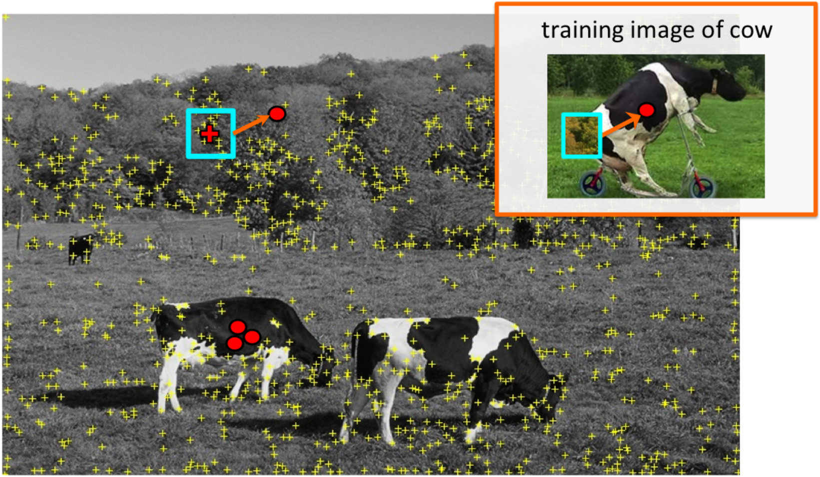
Implicit Shape Model: Basic Idea

- Vote for object center

vote for center of object

Implicit Shape Model: Basic Idea

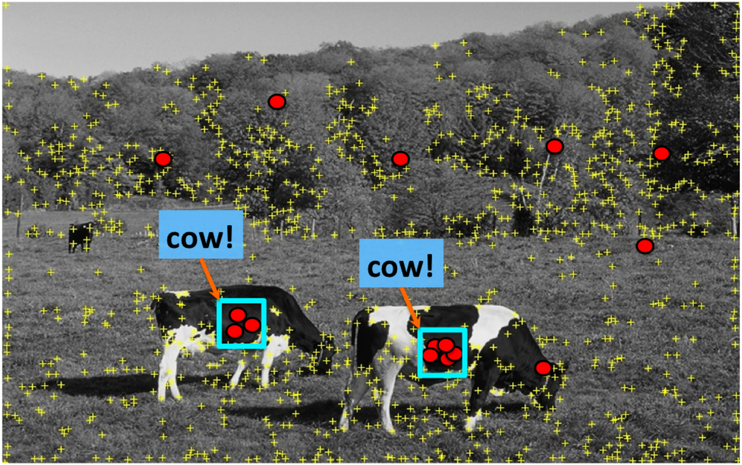
- Vote for object center



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

Implicit Shape Model: Basic Idea

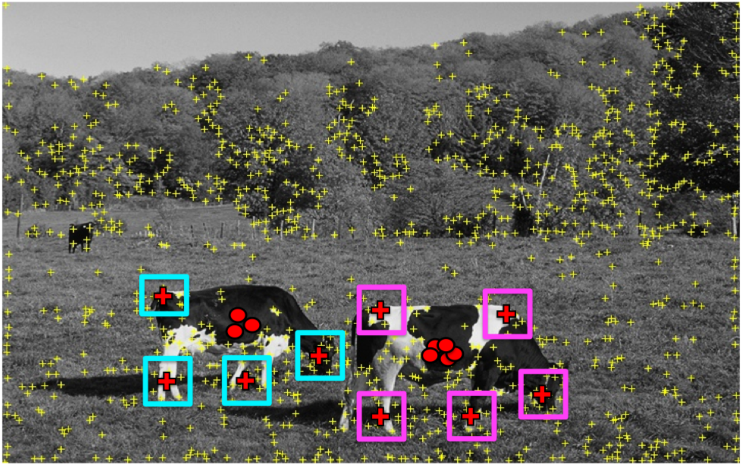
- Vote for object center



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

Implicit Shape Model: Basic Idea

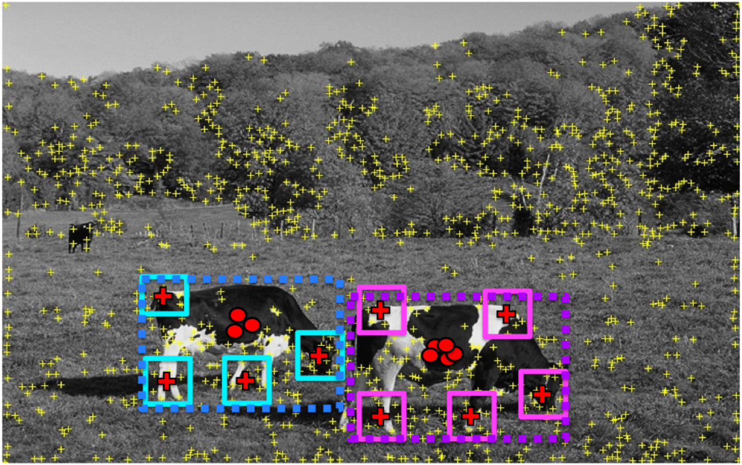
- Find the patches that produced the peak



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.

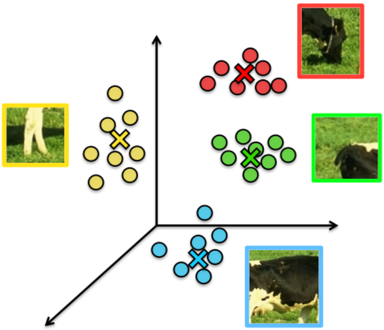
Implicit Shape Model: Basic Idea

- 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

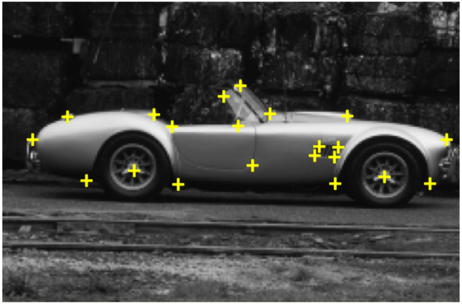
- Training: Getting the vocabulary

training image

Implicit Shape Model: Basic Idea

- Find interest points in each training image

training image

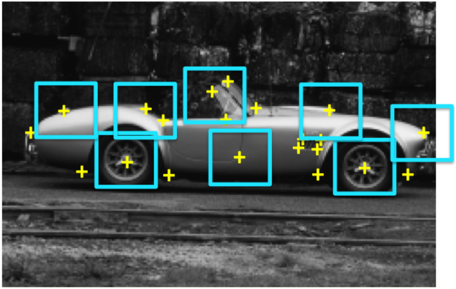


detect interest points (e.g. Harris)

Implicit Shape Model: Basic Idea

- Collect patches around each interest point

training image

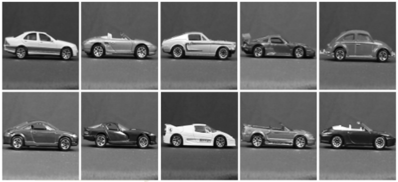


extract an image patch around each interest point

Implicit Shape Model: Basic Idea

- Collect patches across all training examples

training images

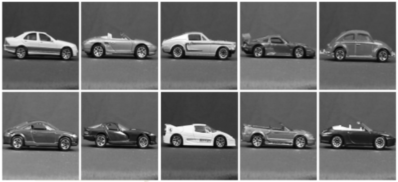


collect all patches

Implicit Shape Model: Basic Idea

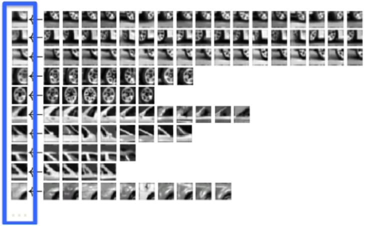
- Cluster the patches to get a small set of "representative" patches

training images



collect all patches

visual codebook

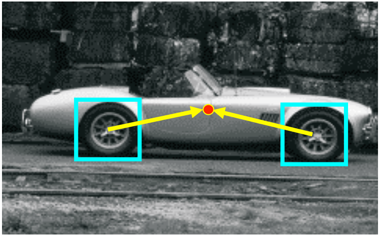


clusters

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

Implicit Shape Model: Training

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



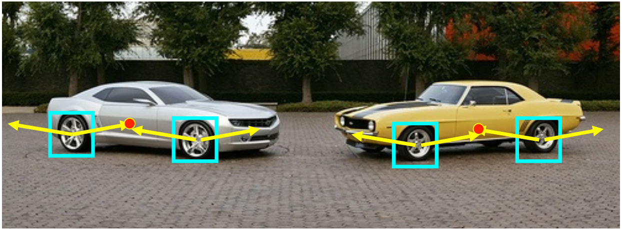
Training image

Visual codeword with displacement vectors

[Leibe et al. IJCV 2008]

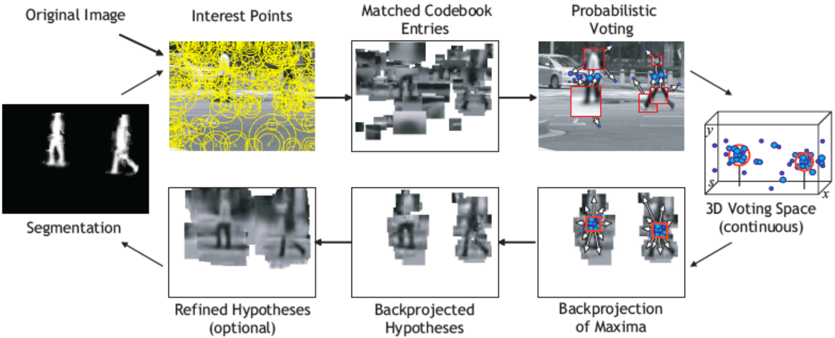
Implicit Shape Model: Test

- 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

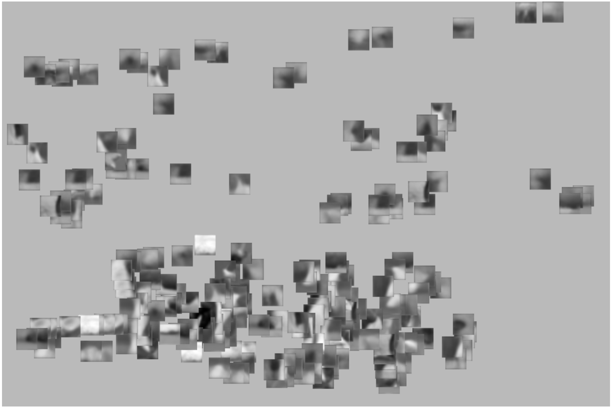
[Source: B. Leibe, credit: R. Urtasun]

Example

Interest points

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

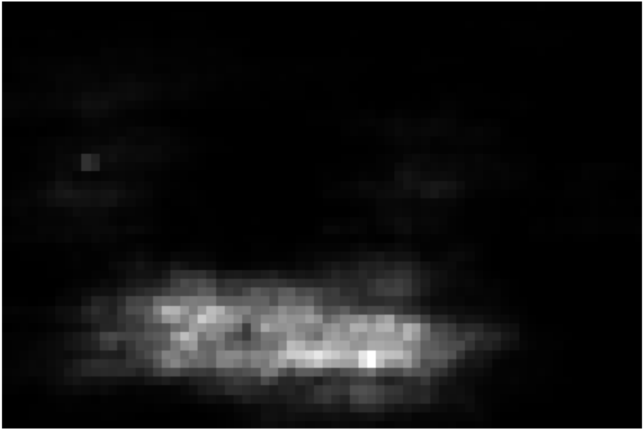
Example



Matched patches

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

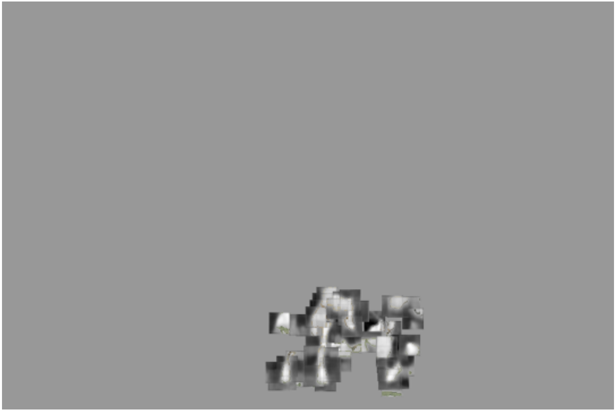
Example



Voting space

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

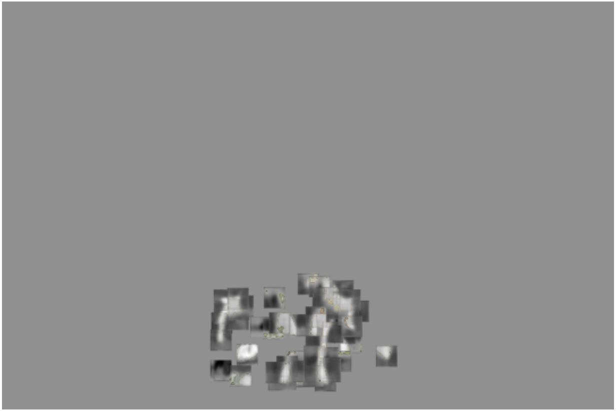
Example



1st hypothesis

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

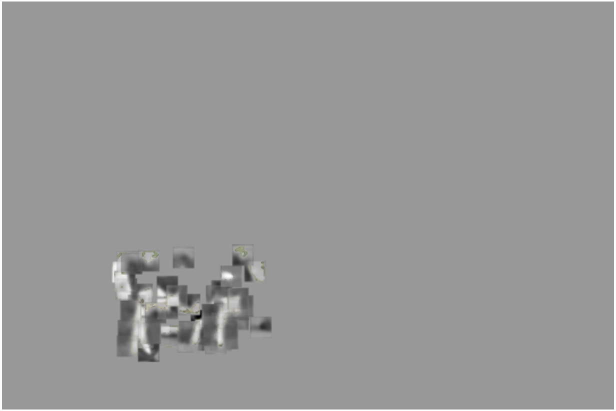
Example



2nd hypothesis

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

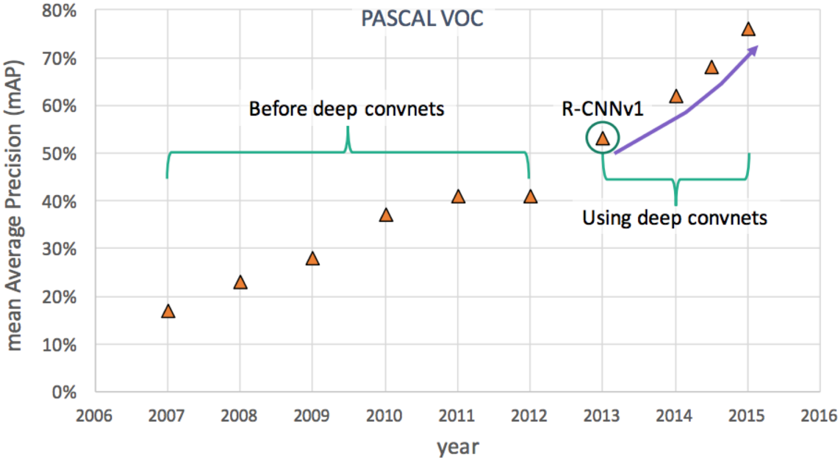
Example



3rd hypothesis

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

The CNN Era



[Slide credit: Renjie Liao]

Deep Object Detection

Object Detection

Published: 09 Oct 2015 Category: deep_learning

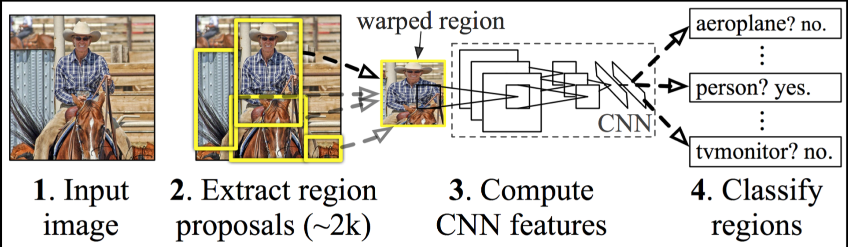
Jump to...

- Leaderboard
- Papers
  - R-CNN
  - MultiBox
  - SPP-Net
  - DeepID-Net
  - NoC
  - Fast R-CNN
  - DeepBox
  - MR-CNN
  - Faster R-CNN
  - YOLO
  - AttentionNet
  - DenseBox

- SSD
- Inside-Outside Net (ION)
- G-CNN
- HyperNet
- MultiPathNet
- CRAFT
- OHEM
- R-FCN
- MS-CNN
- PVANET
- GBD-Net
- StuffNet
- Feature Pyramid Network (FPN)
- YOLOv2
- DSSD

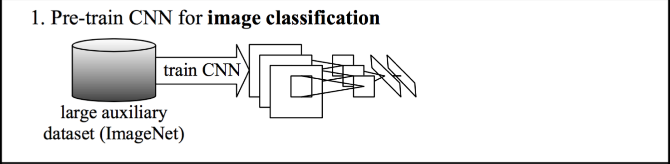
https://handong1587.github.io/deep_learning/2015/10/09/object-detection.html

RCNN: Regions with CNN Features

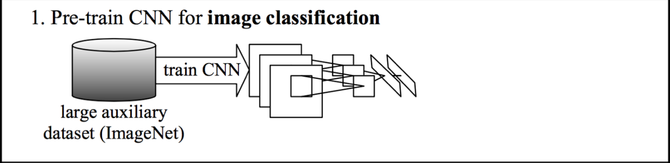


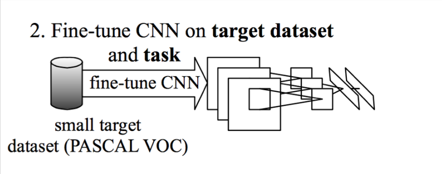
[Slide credit: Ross Girshick]

Training


1. Pre-train CNN for **image classification**
train CNN
large auxiliary
dataset (ImageNet)

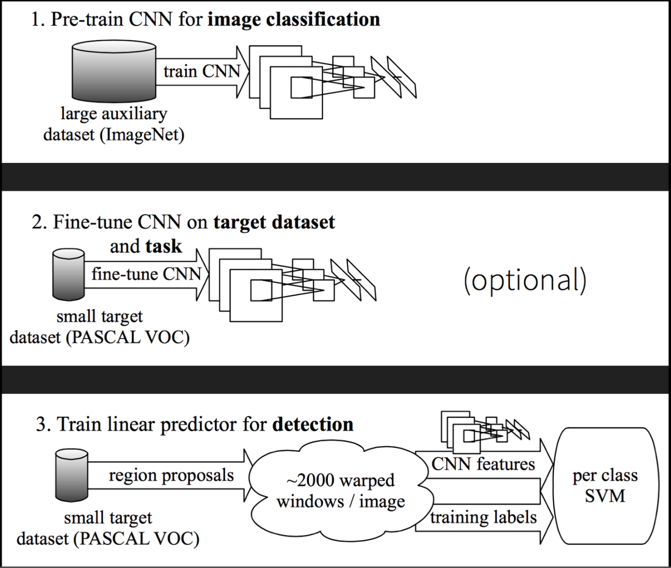
Training





(optional)

Training



(optional)

RCNN: Performance

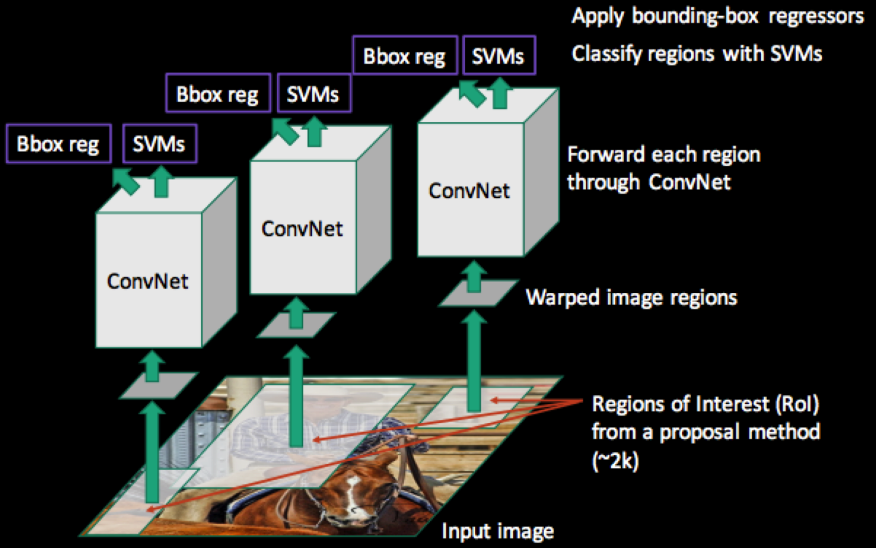
	VOC2007
DPM v5 (Girshick et al. 2011)	33.7%
Regionlets (Wang et al. 2013)	41.7%
R-CNN (AlexNet)	54.2%
R-CNN (AlexNet) + BB	58.5%
R-CNN (VGGNet)	62.2%
R-CNN (VGGNet) + BB	66.0%

RCNN: Performance

	VOC2007
DPM v5 (Girshick et al. 2011)	33.7%
Regionlets (Wang et al. 2013)	41.7%
R-CNN (AlexNet)	54.2%
R-CNN (AlexNet) + BB	58.5%
R-CNN (VGGNet)	62.2%
R-CNN (VGGNet) + BB	66.0%

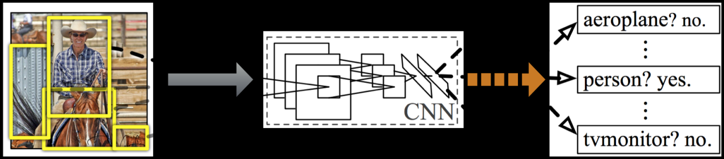
R-CNN (VGGNet)	Time
Train	84 hours
Test	47 s/im

RCNN Pipeline



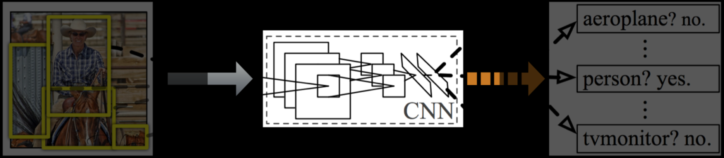
[Slide credit: Ross Girshick]

Object Detection



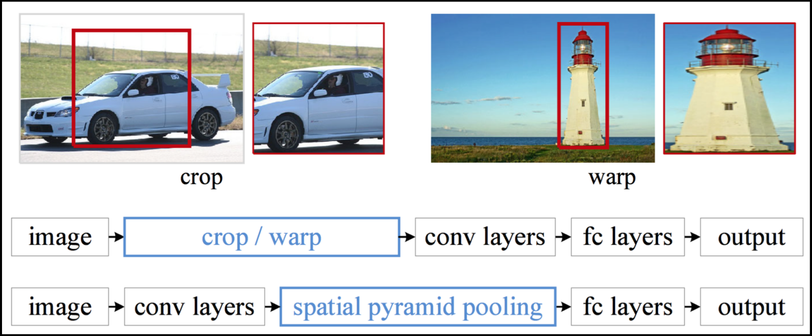
Getting Proposals Feature Extraction Classifier

Object Detection



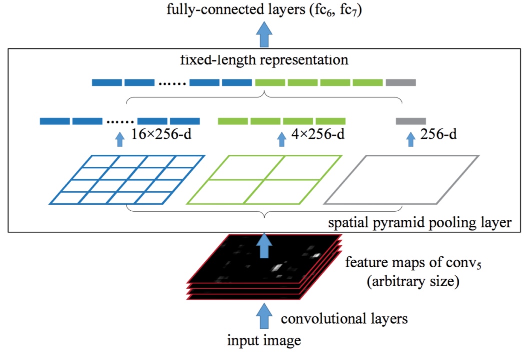
Getting Proposals **Feature Extraction** Classifier

Spatial Pyramid Pooling



Slide credit: K. He, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition. ECCV2014

Spatial Pyramid Pooling

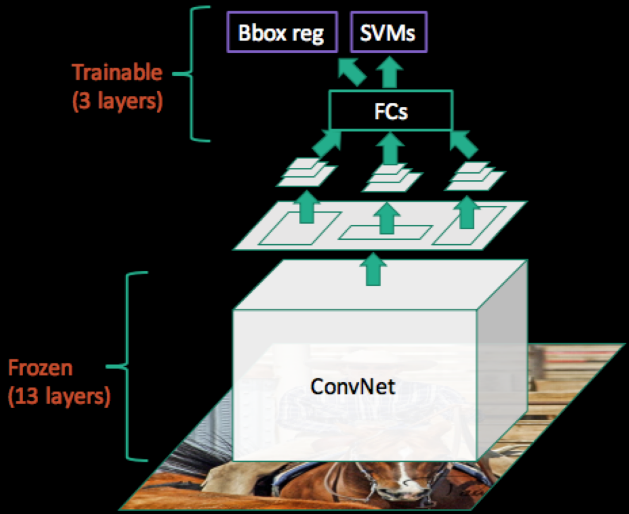



```
[pool3x3]        [pool2x2]        [pool1x1]
type=pool        type=pool        type=pool
pool=max         pool=max         pool=max
inputs=conv5     inputs=conv5     inputs=conv5
sizeX=5          sizeX=7          sizeX=13
stride=4         stride=6         stride=13

[fc6]
type=fc
outputs=4096
inputs=pool3x3,pool2x2,pool1x1
```

Slide credit: K. He, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition. ECCV2014

SPP-Net

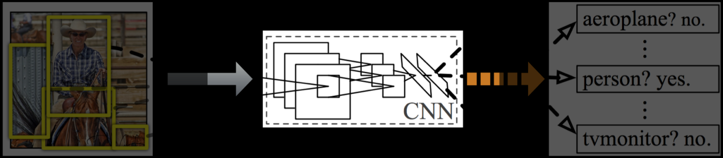


[Slide credit: Ross Girshick]

SPP-Net: Performance

	VOC2007	Speed
R-CNN (ZFNet)	59.2%	14.5 s/im
R-CNN (VGGNet)	66.0%	47.0 s/im
SPP (ZFNet)	59.2%	0.38 s/im
SPP (VGGNet)	63.1%	2.3 s/im

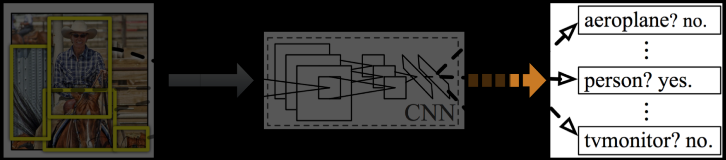
Object Detection



Getting Proposals **Feature Extraction** Classifier

SPP

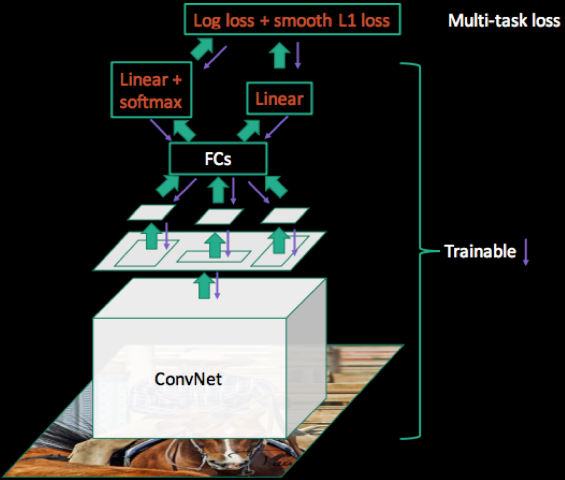
Object Detection



Getting Proposals Feature Extraction **Classifier**

Fast R-CNN

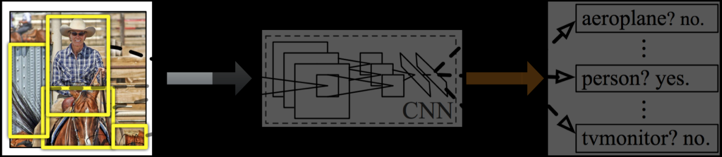
Totally end-to-end!



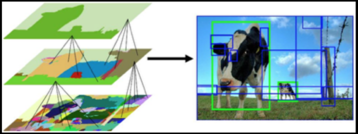
Fast R-CNN: Performance

	VOC2007
SPPNet BB	63.1%
R-CNN BB	66.0%
Fast RCNN	66.9%
Fast RCNN (07+12)	70.0%

Object Detection

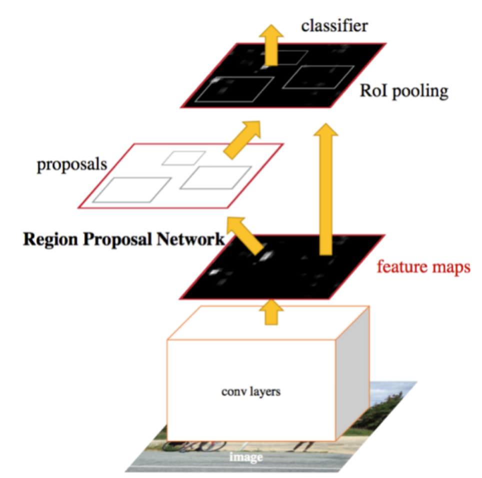


Getting Proposals Feature Extraction Classifier



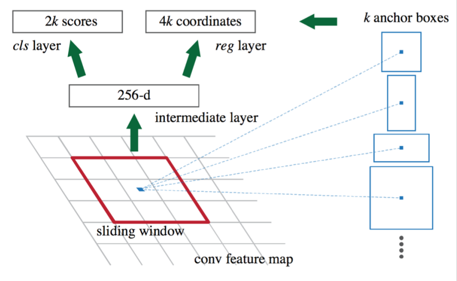
(e.g. selective search)

Faster R-CNN



Ren S, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NIPS2015

Region Proposal Network (RPN)



- Sliding window style
- Multi-scale predictions on fix-sized window for efficiency (take advantage of the large receptive field of CNN features)
- Same loss as R-CNN (cls+bbox)

anchor	128^2, 2:1	128^2, 1:1	128^2, 1:2	256^2, 2:1	256^2, 1:1	256^2, 1:2	512^2, 2:1	512^2, 1:1	512^2, 1:2
proposal	188×111	113×114	70×92	416×229	261×284	174×332	768×437	499×501	355×715

Region Proposal Network (RPN)

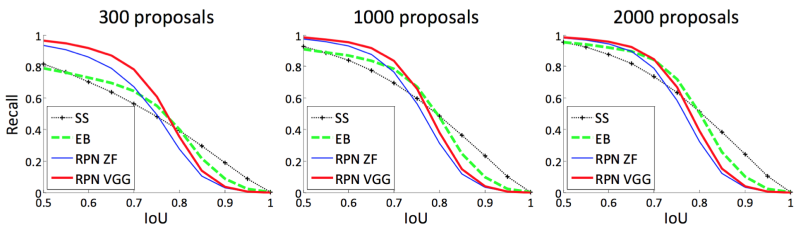


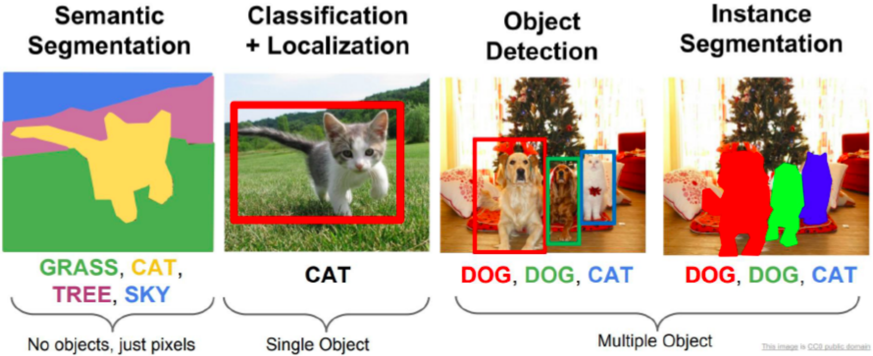
Figure 2: Recall *vs.* IoU overlap ratio on the PASCAL VOC 2007 test set.

Faster R-CNN: Performance

- Fewer and better proposals not only bring speed-up, but also detection performance boost.

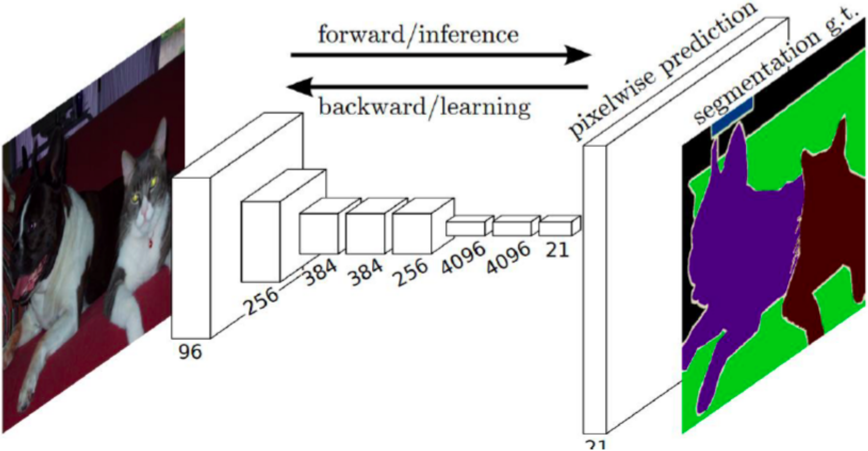
method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	**196**
RPN+VGG, shared	300	07+12	**73.2**	**196**

Recognition Tasks

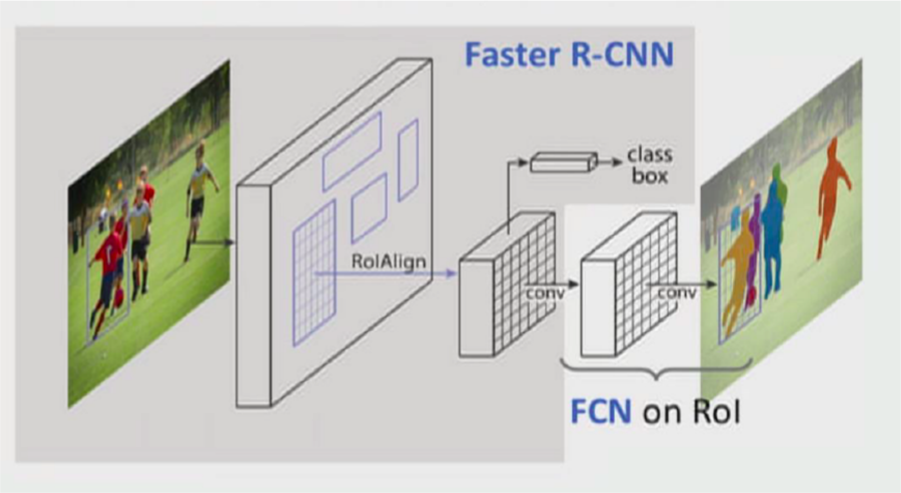


Mask-RCNN

Segmentation via FCN

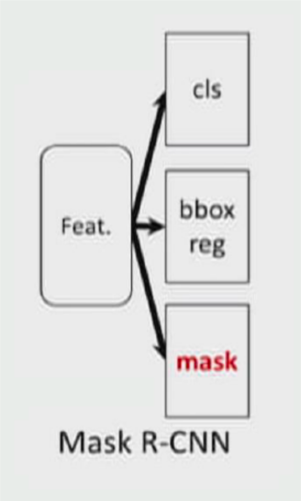


Mask-RCNN

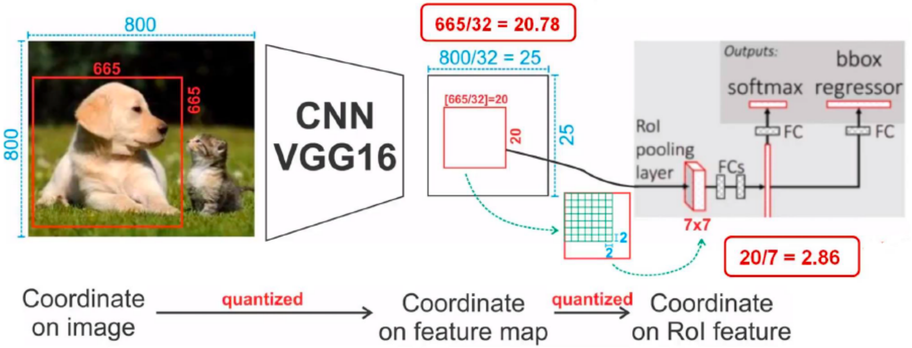


Mask-RCNN: Multiple Heads

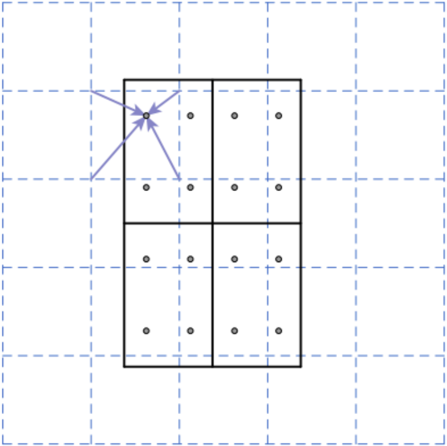
- Loss for each proposal is:
  $L = L\_cls + L\_box + L\_mask$



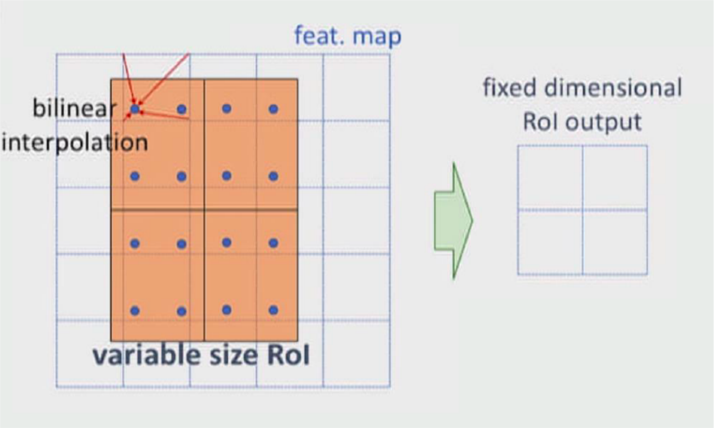
Mask-RCNN: RoI Align



Mask-RCNN: RoI Align



Mask-RCNN: RoI Align



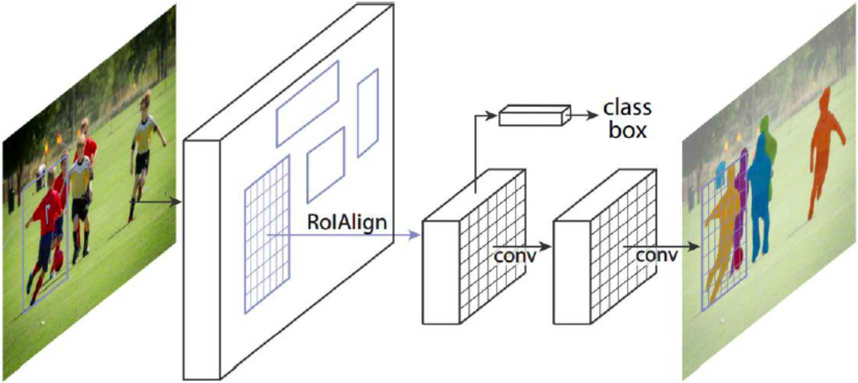
Mask-RCNN: RoI Align

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

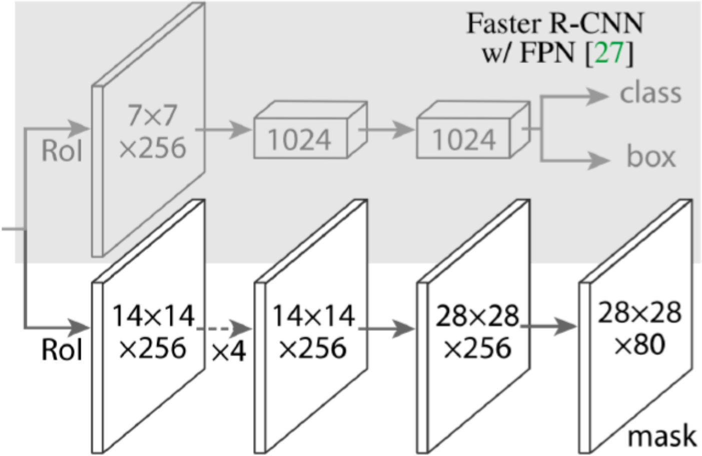
(a) RoIAlign (ResNet-50-C4) comparison

	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

Mask-RCNN: Mask Head



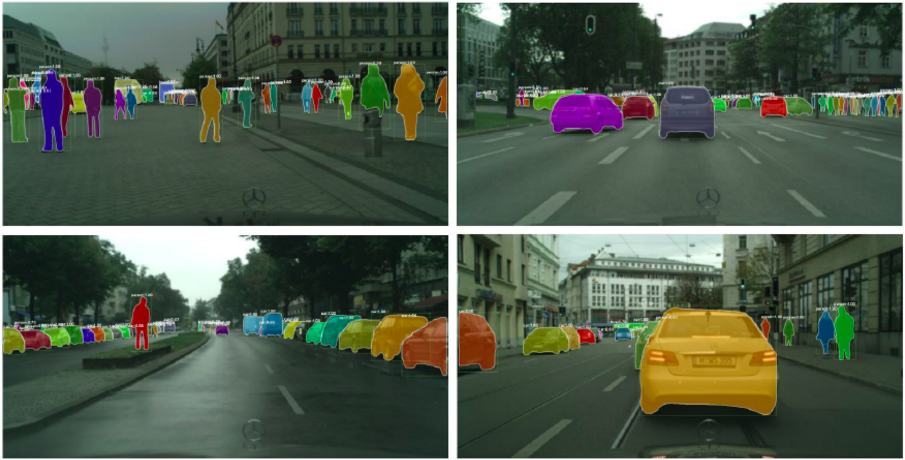
Mask-RCNN: Mask Head



Mask-RCNN: Detection Results

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

Mask-RCNN: Instance Segmentation

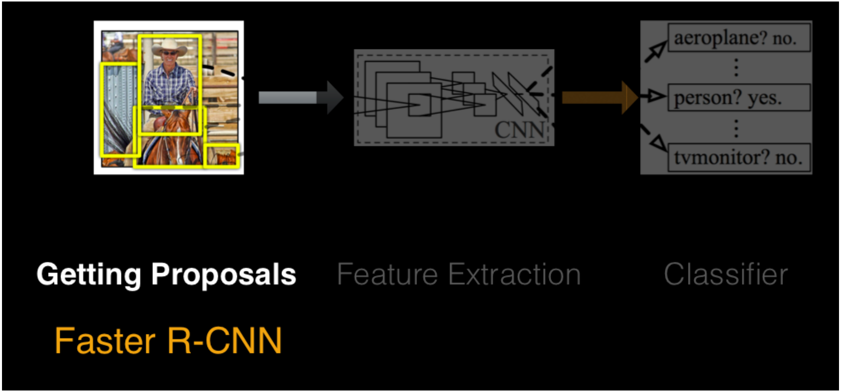


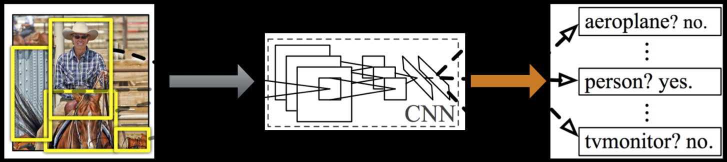
Mask-RCNN: Instance Segmentation

	training data	AP [val]	AP	AP_{50}	person	rider	car	truck	bus	train	mcycle	bicycle
InstanceCut [23]	fine + coarse	15.8	13.0	27.9	10.0	8.0	23.7	14.0	19.5	15.2	9.3	4.7
DWT [4]	fine	19.8	15.6	30.0	15.1	11.7	32.9	17.1	20.4	15.0	7.9	4.9
SAIS [17]	fine	-	17.4	36.7	14.6	12.9	35.7	16.0	23.2	19.0	10.3	7.8
DIN [3]	fine + coarse	-	20.0	38.8	16.5	16.7	25.7	20.6	30.0	23.4	17.1	10.1
SGN [29]	fine + coarse	29.2	25.0	44.9	21.8	20.1	39.4	24.8	33.2	30.8	17.7	12.4
Mask R-CNN	fine	31.5	26.2	49.9	30.5	23.7	46.9	22.8	32.2	18.6	19.1	16.0
Mask R-CNN	fine + COCO	**36.4**	**32.0**	**58.1**	**34.8**	**27.0**	**49.1**	**30.1**	**40.9**	**30.9**	**24.1**	**18.7**

Mask-RCNN: Pose

Object Detection





Getting Proposals | Feature Extraction | Classifier

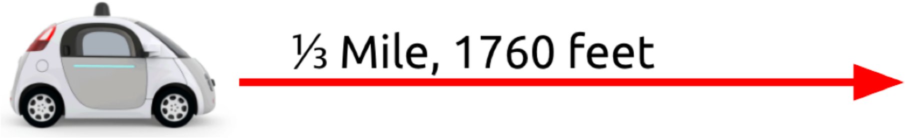
Faster R-CNN | SPP | Fast R-CNN

66.0% —> 73.2%

47 s/im —> 0.2 s/im

Car Example

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img



[Slide credit: Joseph Chet Redmon]

Car Example

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img

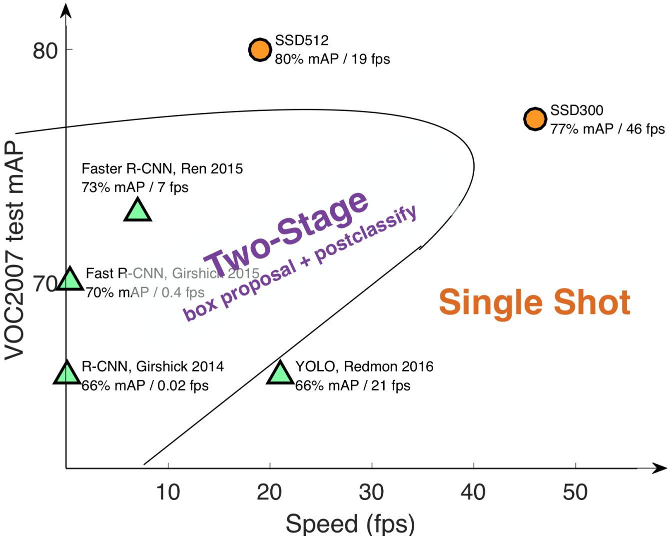
[Slide credit: Joseph Chet Redmon]

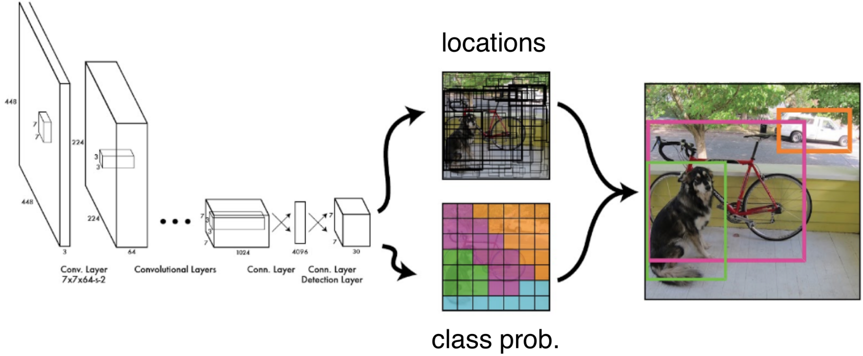
Car Example

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img

[Slide credit: Joseph Chet Redmon]

Real Time Object Detection?



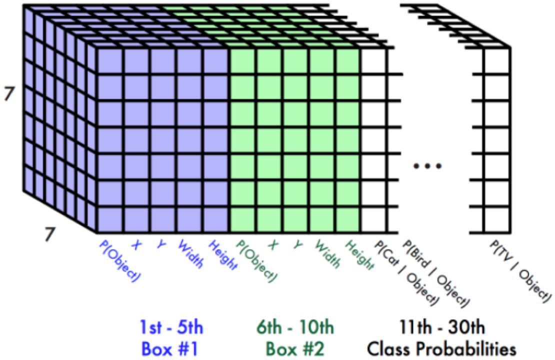


[Slide credit: Redmon J et al. You only look once: Unified, real-time object detection. CVPR'16]

YOLO: Output Parametrization

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities



For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

[Slide credit: Redmon J et al. You only look once: Unified, real-time object detection. CVPR'16]

YOLO Limitations

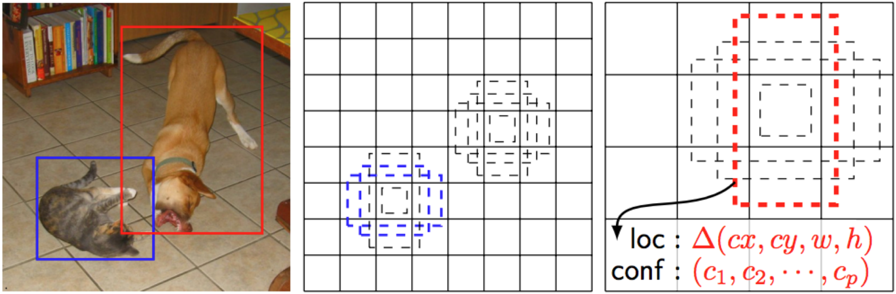
- Small objects
- Objects with different shapes and sizes
- Occluded objects

SSD: Single Shot MultiBox Detector

[Slide credit: Wei Liu et al., SSD: Single Shot MultiBox Detector, ECCV'16]

SSD: Single Shot MultiBox Detector

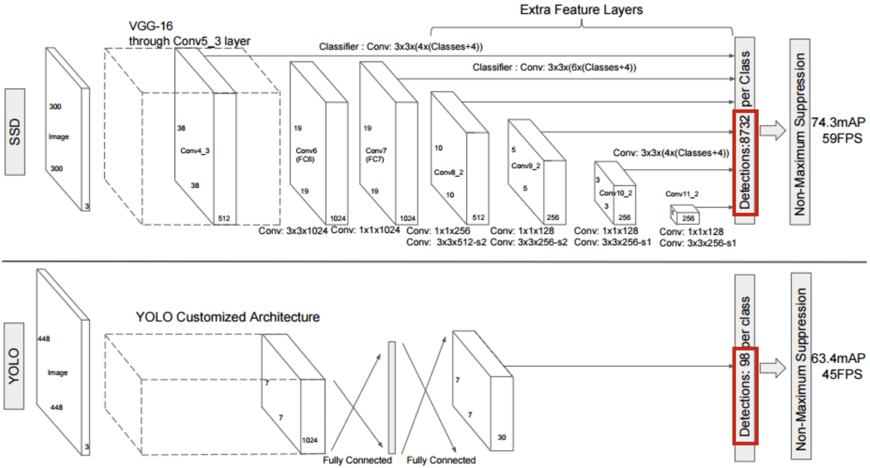
- SSD: YOLO + default box shape + multi-scale



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

[Slide credit: Wei L, et al. SSD: Single Shot MultiBox Detector. ECCV'16]

SSD: Single Shot MultiBox Detector



[Slide credit: Wei L, et al. SSD: Single Shot MultiBox Detector. ECCV'16]

Thank you and good luck!