CSE373: Data Structures & Algorithms

Lecture 23: Applications

Linda Shapiro

Spring 2016

Announcements

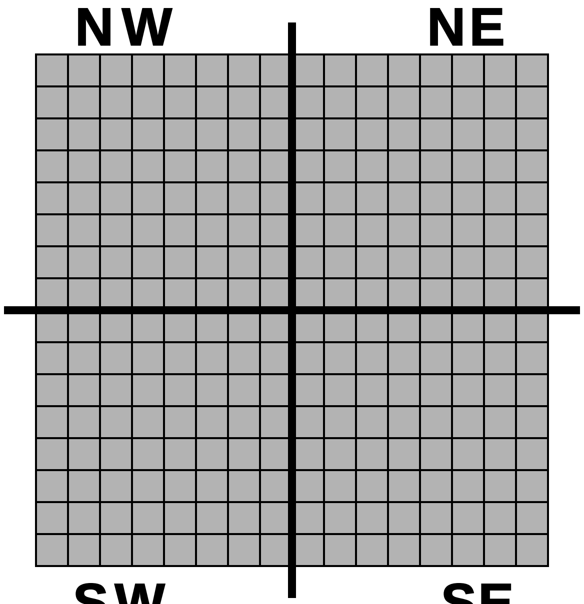
Other Data Structures and Algorithms

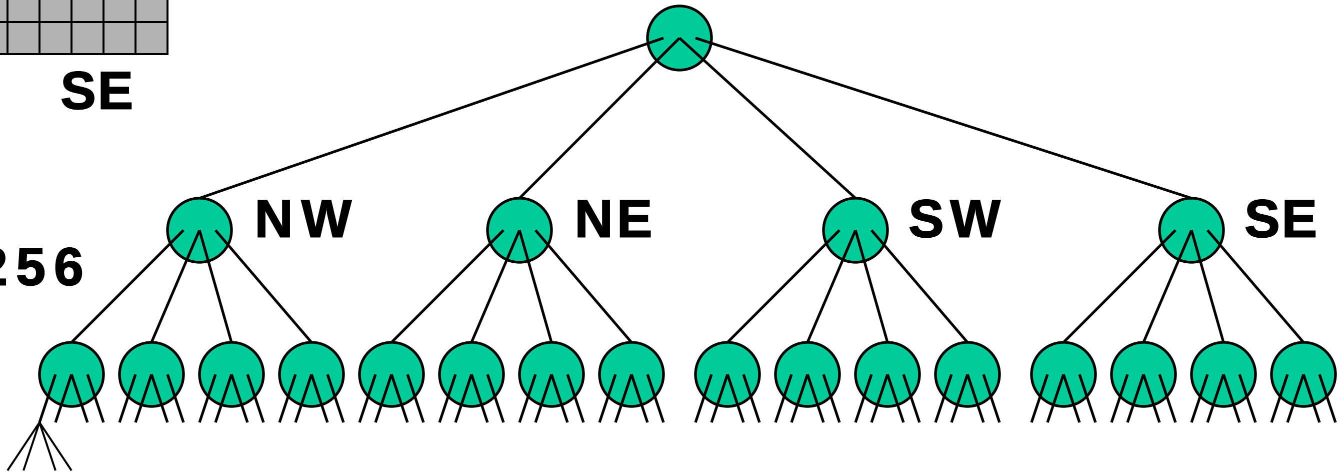
- Quadtrees: used in spatial applications like geography and image processing
- Octrees: used in vision and graphics
- Image pyramids: used in image processing and computer vision
- Backtracking search: used in AI and vision
- Graph matching: used in AI and vision
- Neural nets and deep learning

Quadtrees

- Finkel and Bentley, 1974
- Lots of work by Hanan Samet, including a book
- Raster structure: divides space, not objects
- Form of *block coding:* compact storage of a large 2-dimensional array
- Vector versions exist too

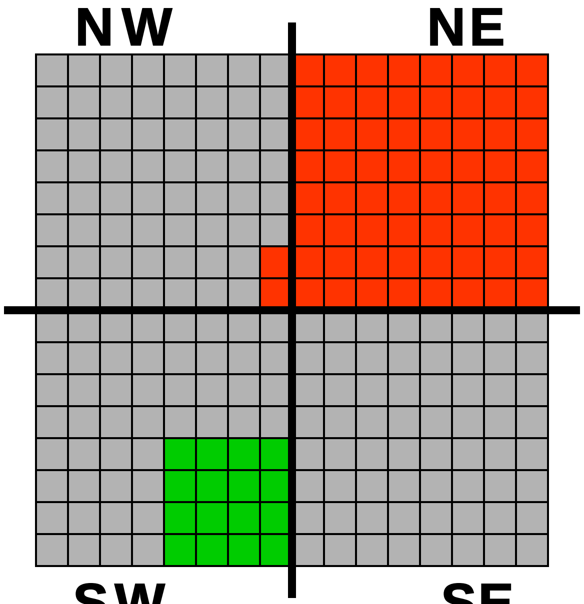
Quadtrees, the idea



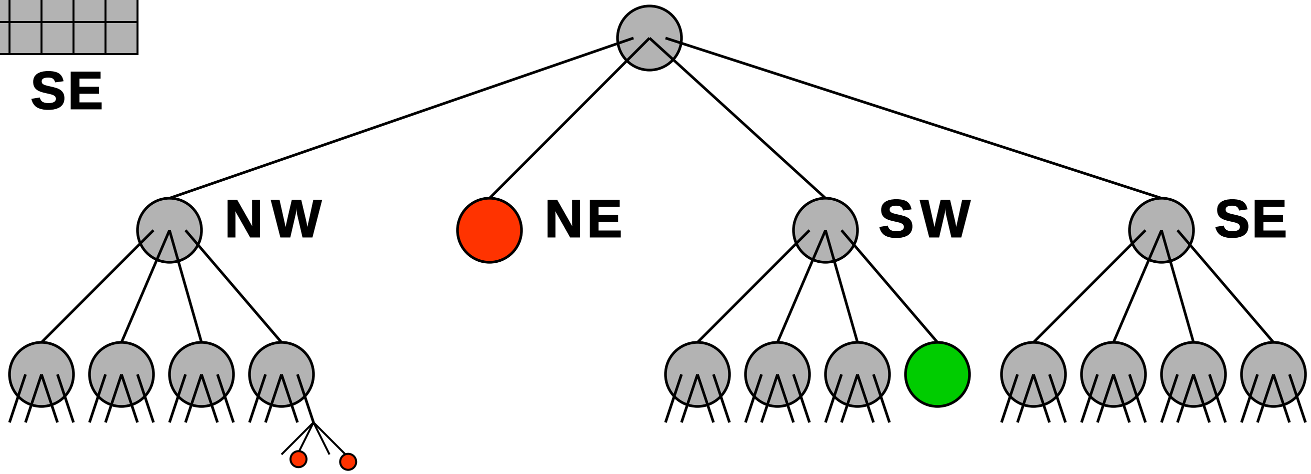


1, 4, 16, 64, 256 nodes

Quadtrees, the idea



Choropleth raster map



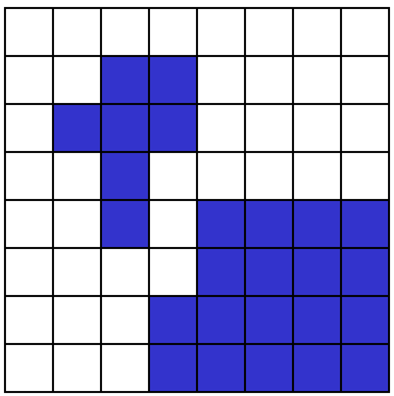
Quadtrees

- Grid with 2^k times 2^k pixels
- Depth is k +1
- Internal nodes always have 4 children
- Internal nodes represent a non-homogeneous region
- Leaves represent a homogeneous region and store the common value (or name)

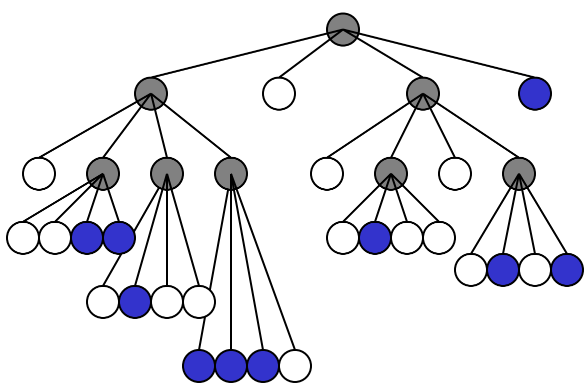
Quadtree complexity theorem

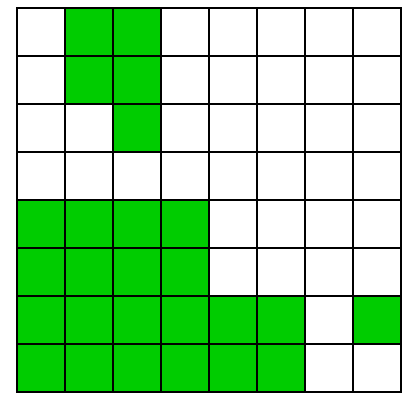
- A subdivision with boundary length r pixels in a grid of 2^k times 2^k gives a quadtree with $O(k \cdot r)$ nodes.
- Idea: two adjacent, different pixels "cost" at most 2 paths in the quadtree.

Overlay with quadtrees

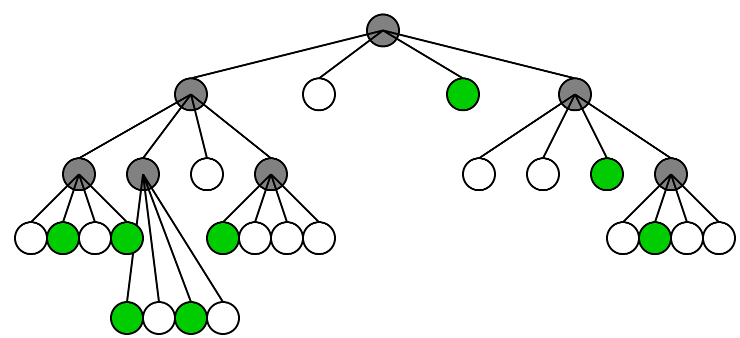


Water

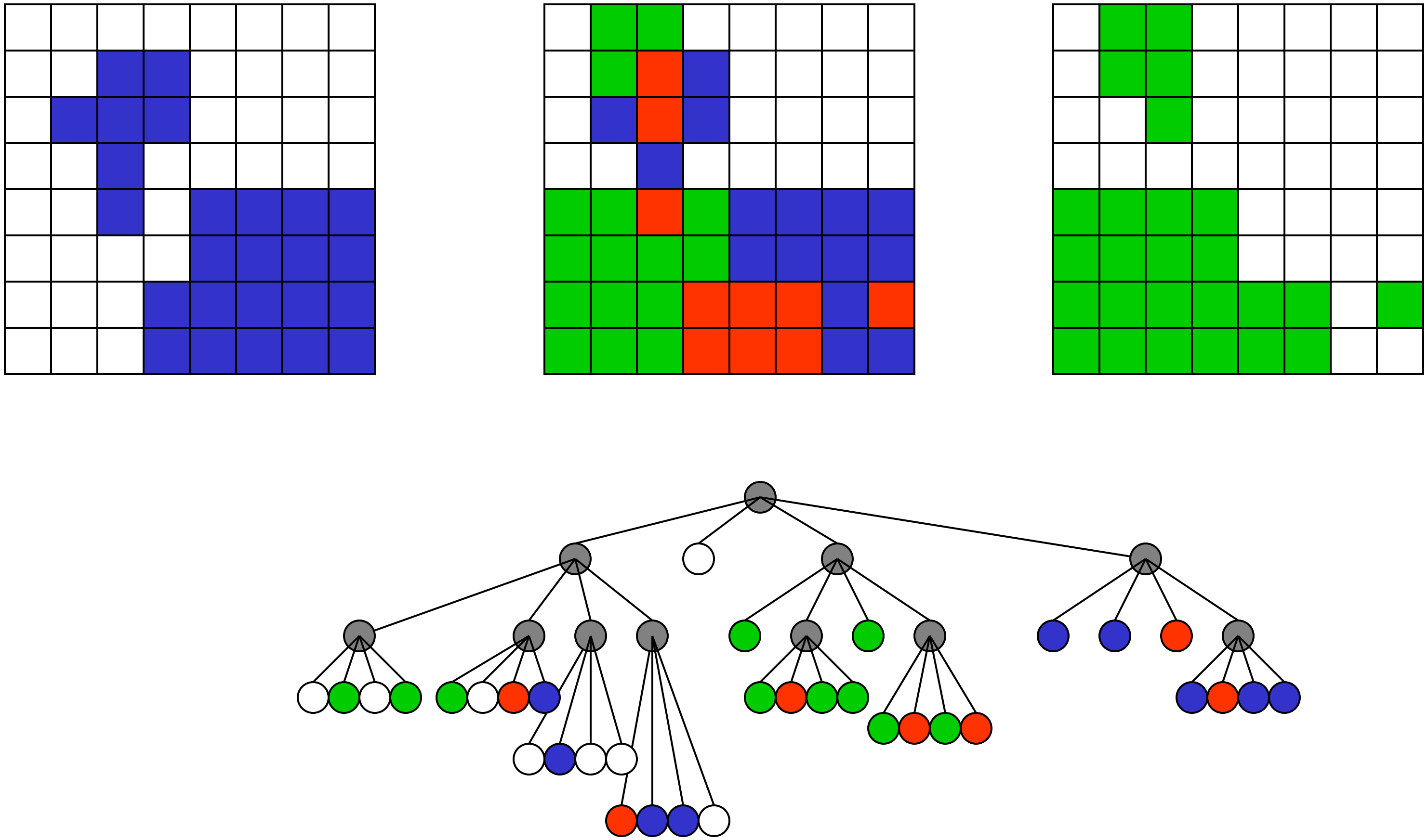




Acid rain with PH below 4.5

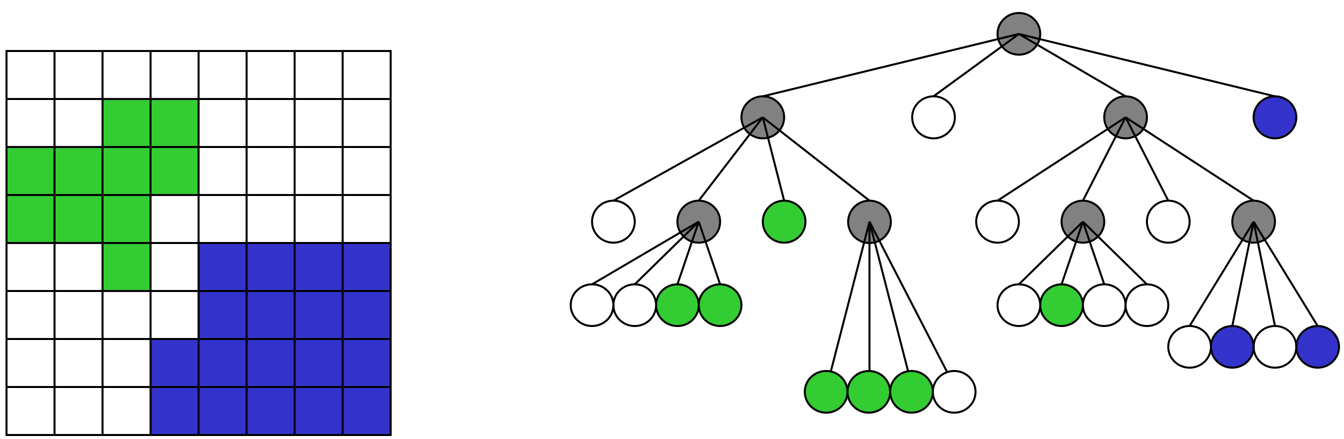


Result of overlay



Various queries

- Point location: trivial
- Windowing: descend into subtree(s) that intersect query window
- Traversal boundary polygon: up and down in the quadtree



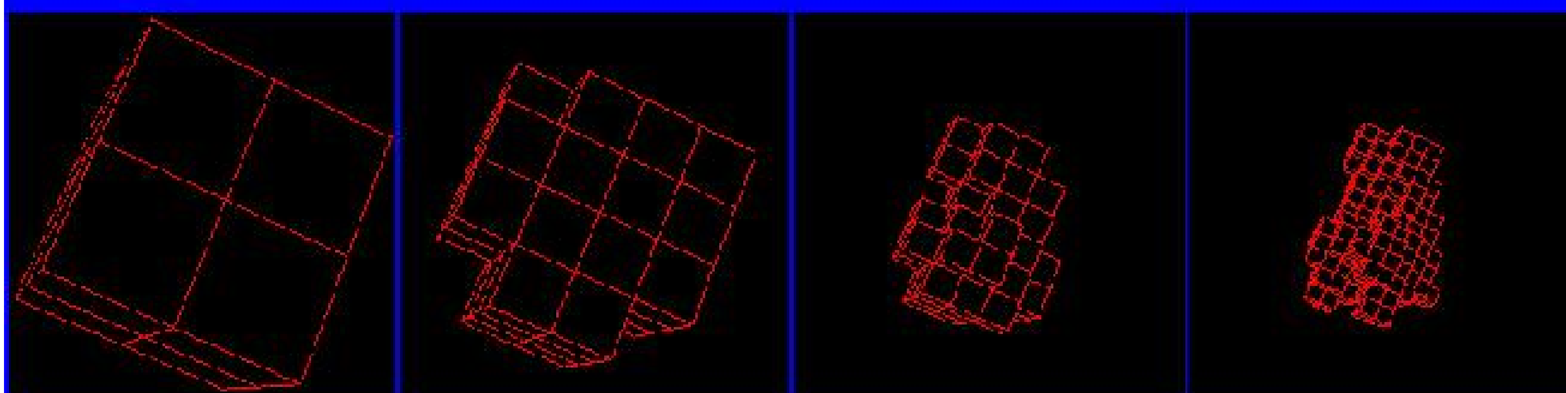
Octrees

- Like quadtrees, but for 3D applications.
- Breaks 3D space into octants
- Useful in graphics for representing 3D objects at different resolutions

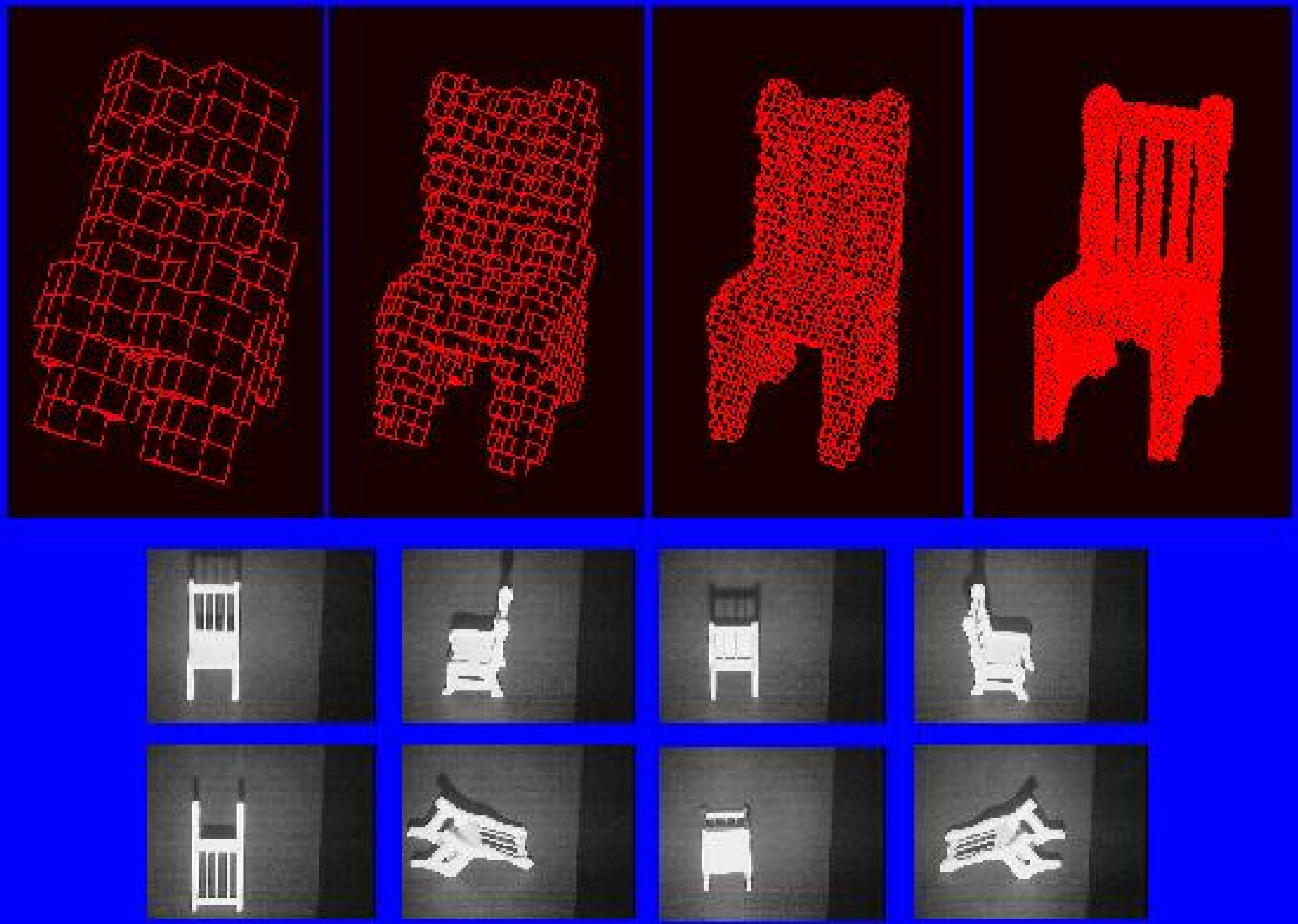
Hierarchical space carving

- Big cubes => fast, poor results
- Small cubes => slow, more accurate results
- Combination = octrees

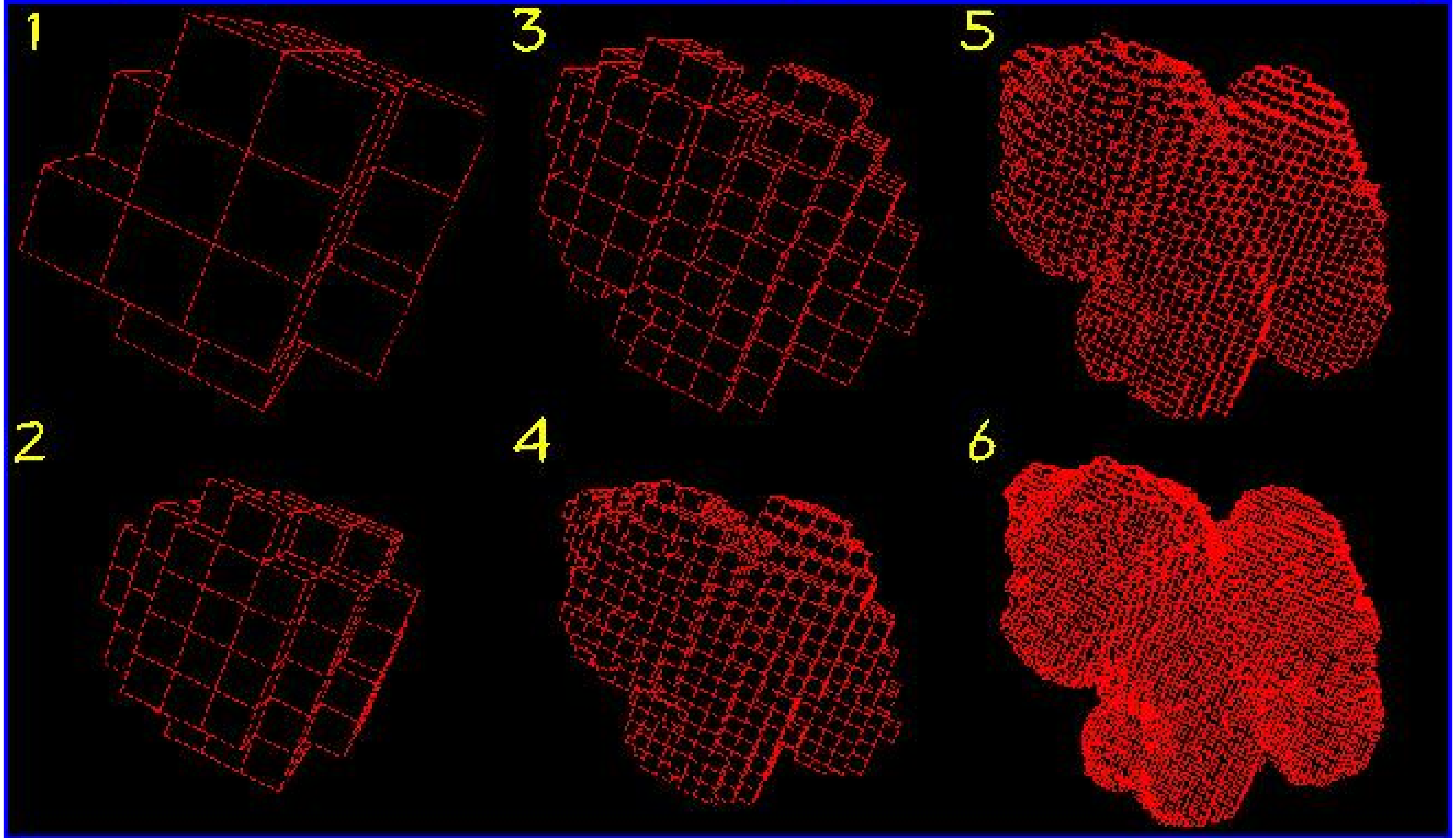
RULES:
- cube's out => done
- cube's in => done
- else => recurse

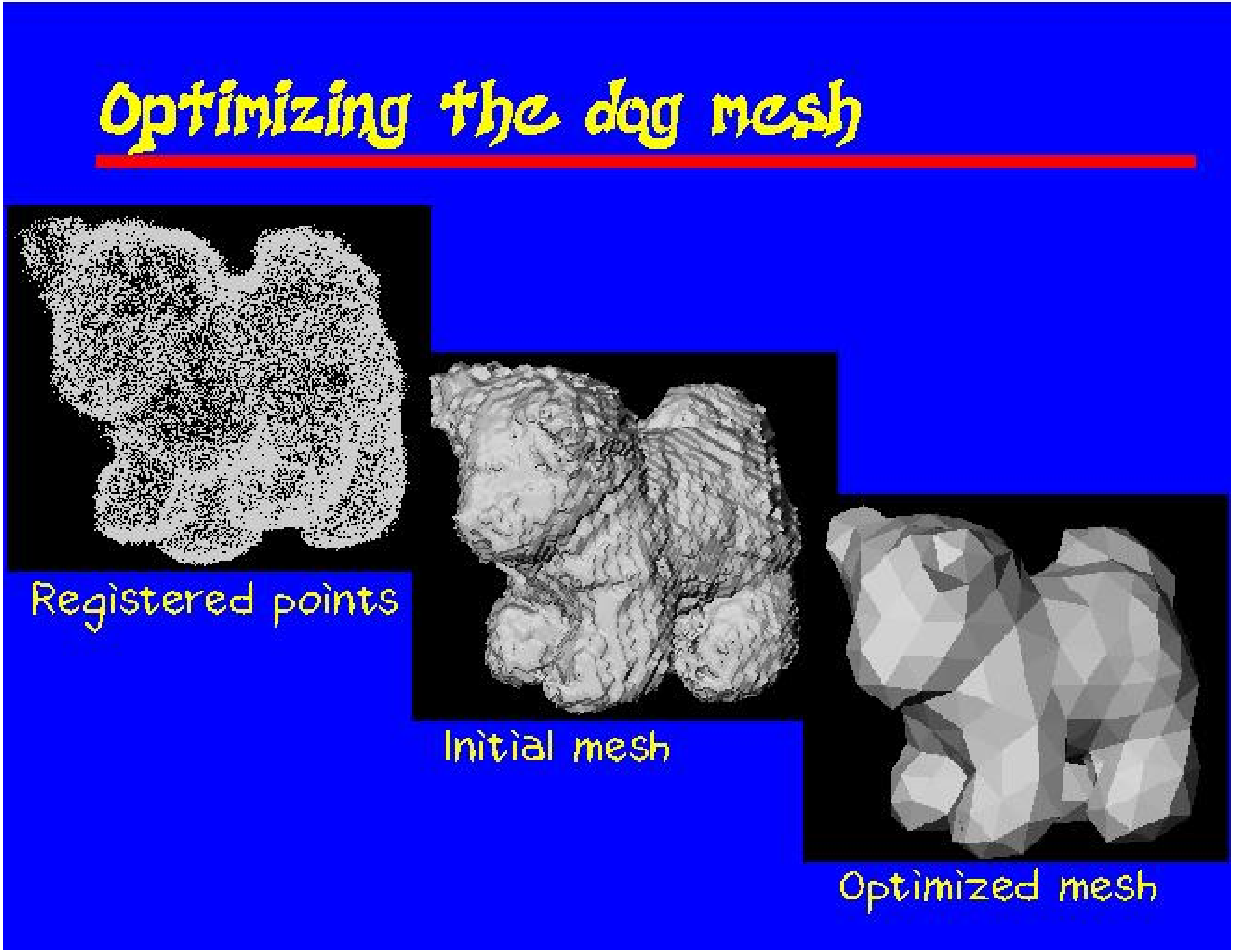


The rest of the chair

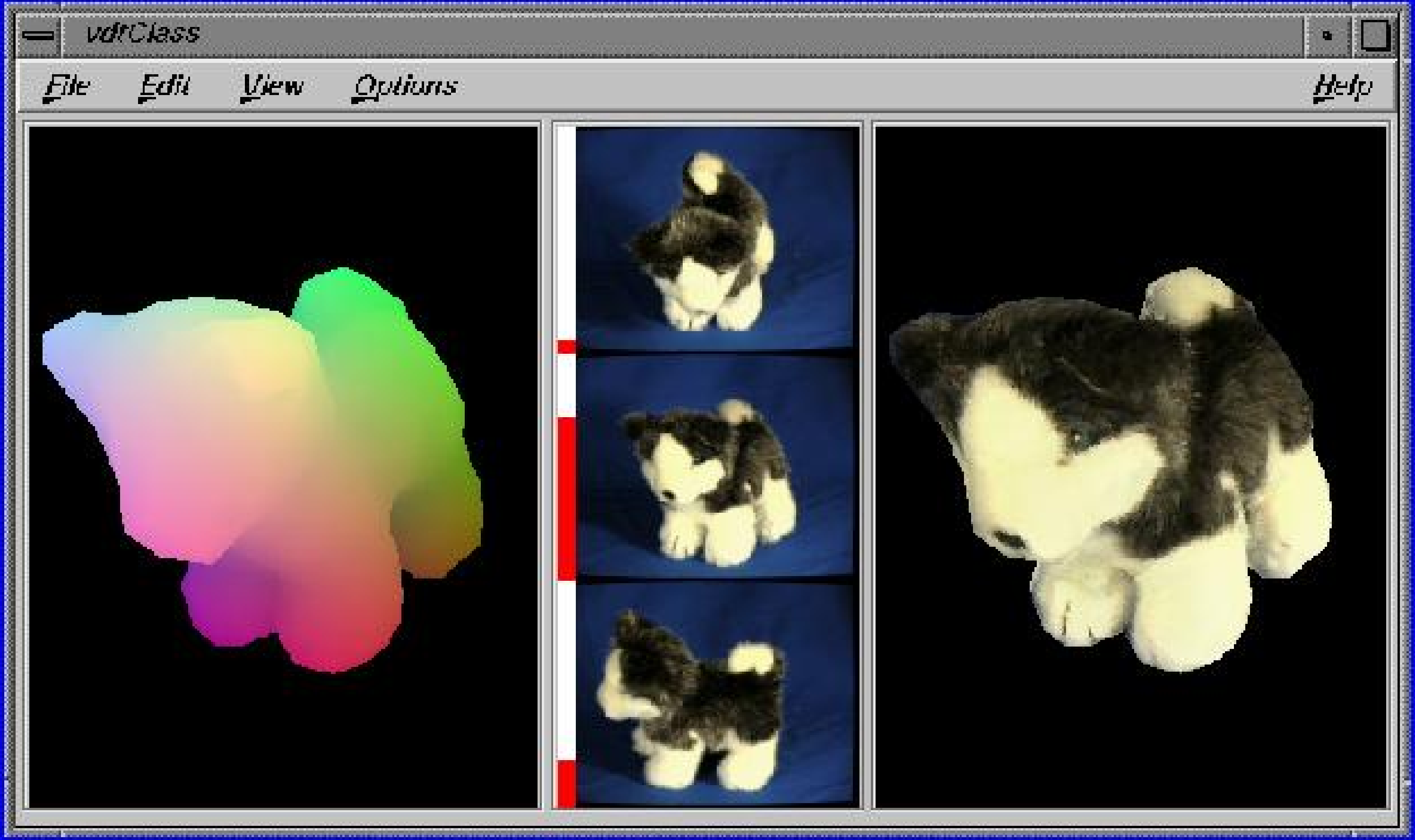


Same for a husky pup




Optimizing the dog mesh
Registered points
Initial mesh
Optimized mesh

Our viewer



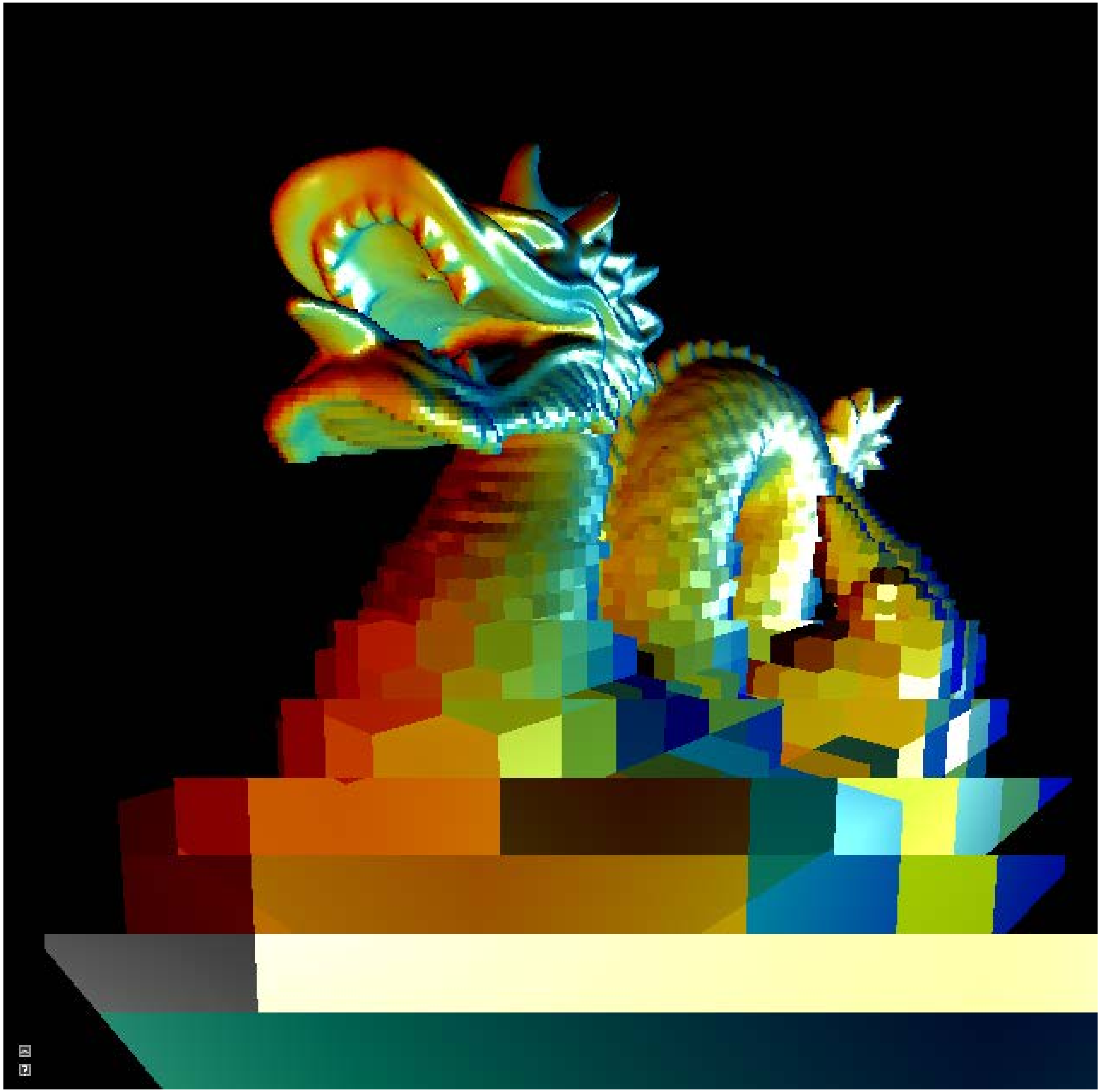
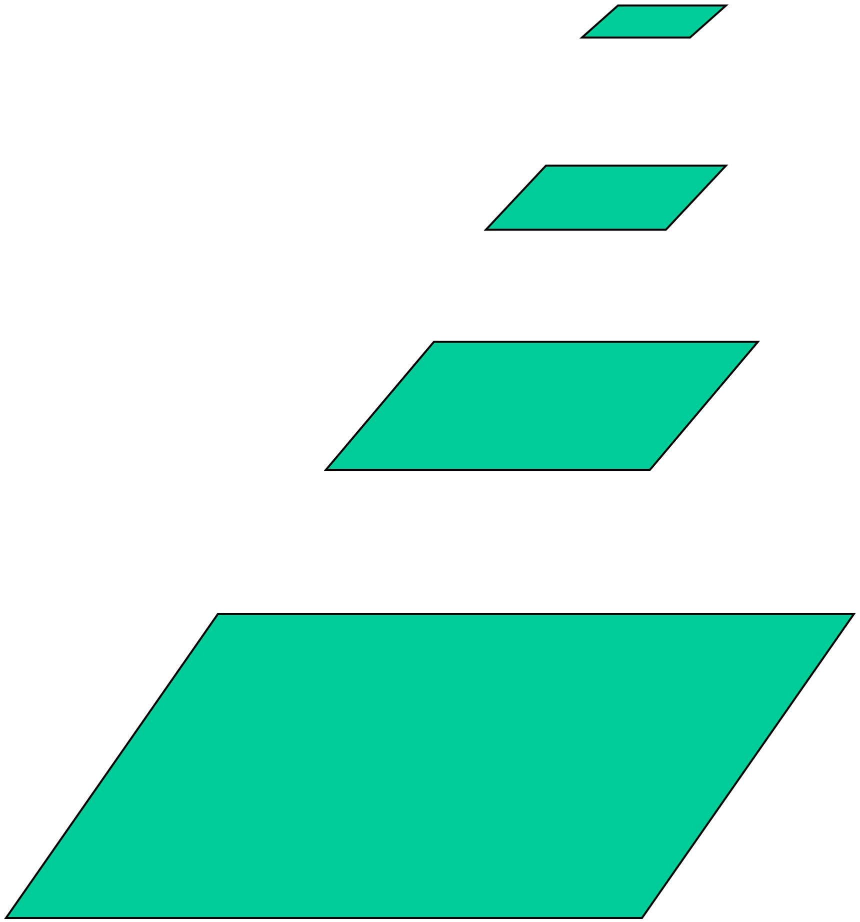


Image Pyramids



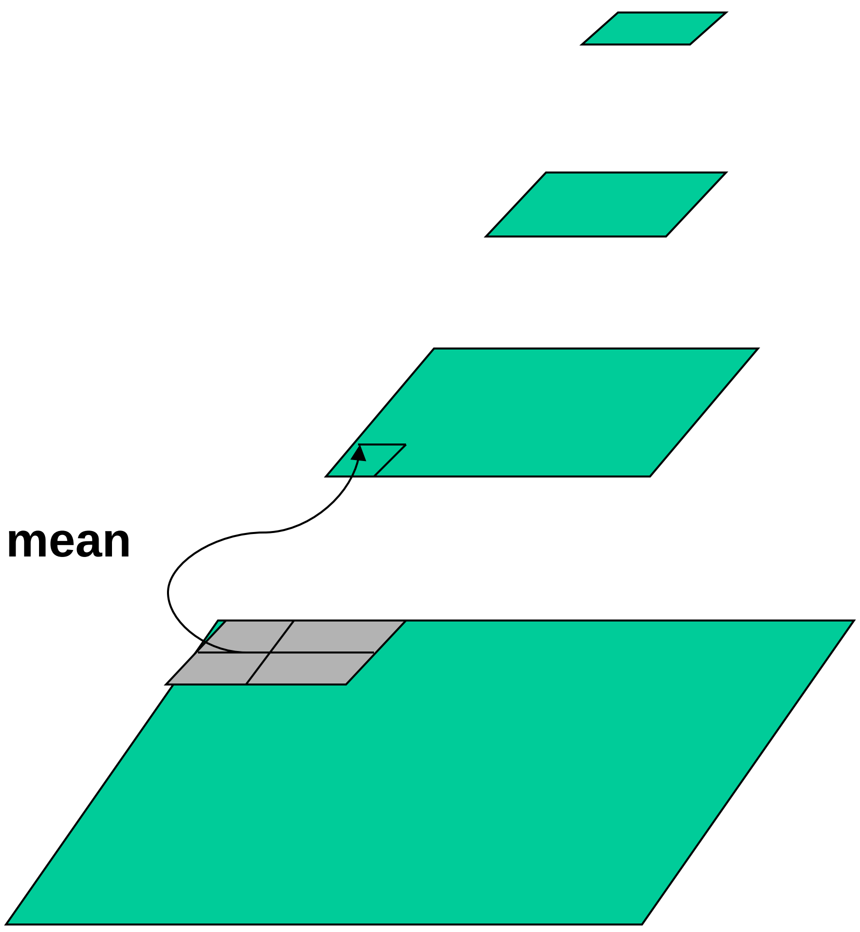
And so on.

3rd level is derived from the 2nd level according to the same funtion

2nd level is derived from the original image according to some function

Bottom level is the original image.

Mean Pyramid



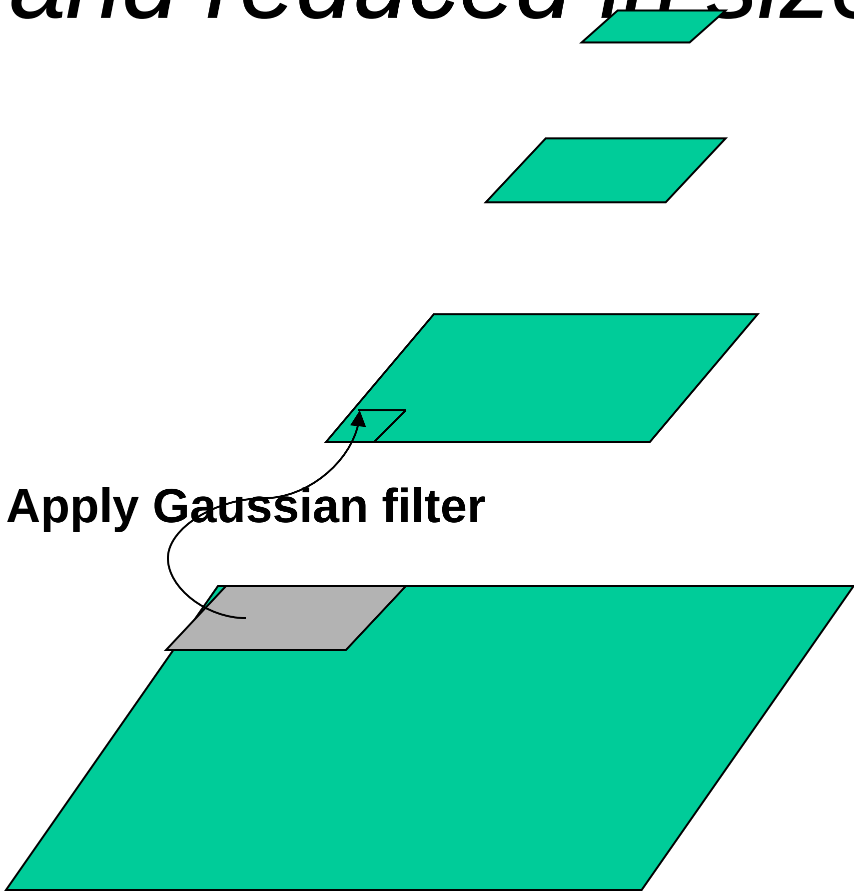
And so on.

At 3rd level, each pixel is the mean of 4 pixels in the 2nd level.

At 2nd level, each pixel is the mean of 4 pixels in the original image.

Bottom level is the original image.

Gaussian Pyramid
At each level, image is smoothed and reduced in size.

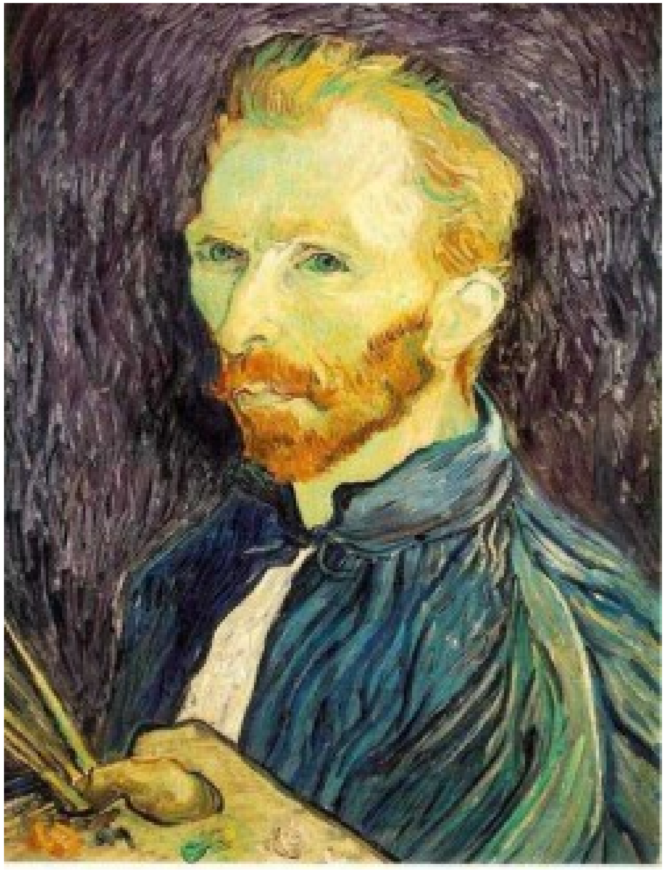


And so on.

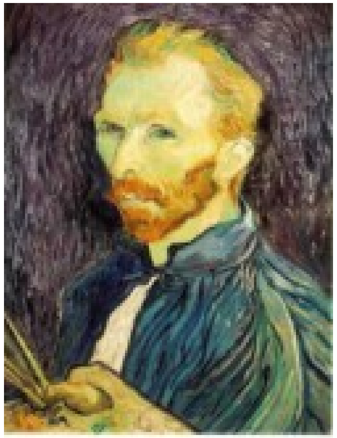
At 2nd level, each pixel is the result of applying a Gaussian mask to the first level and then subsampling to reduce the size.

Bottom level is the original image.

Example: Subsampling with Gaussian pre-filtering



Gaussian 1/2



G 1/4

G 1/8

Backtracking Search in AI/Vision

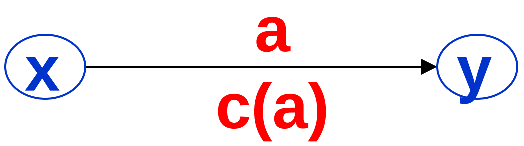
- Start at the root of a search tree at a “state”
- Generate children of that state
- For each child
  - If the child is the goal, done
  - If the child does not satisfy the constraints of the problem, ignore it and keep going in this loop
  - Else call the search recursively for this child
- Return

This is called **backtracking,** because if it goes through all children of a node and finds no solution, it returns to the parent and continues with the children of that parent.

Formal State-Space Model

Problem = (S, s, A, f, g, c)

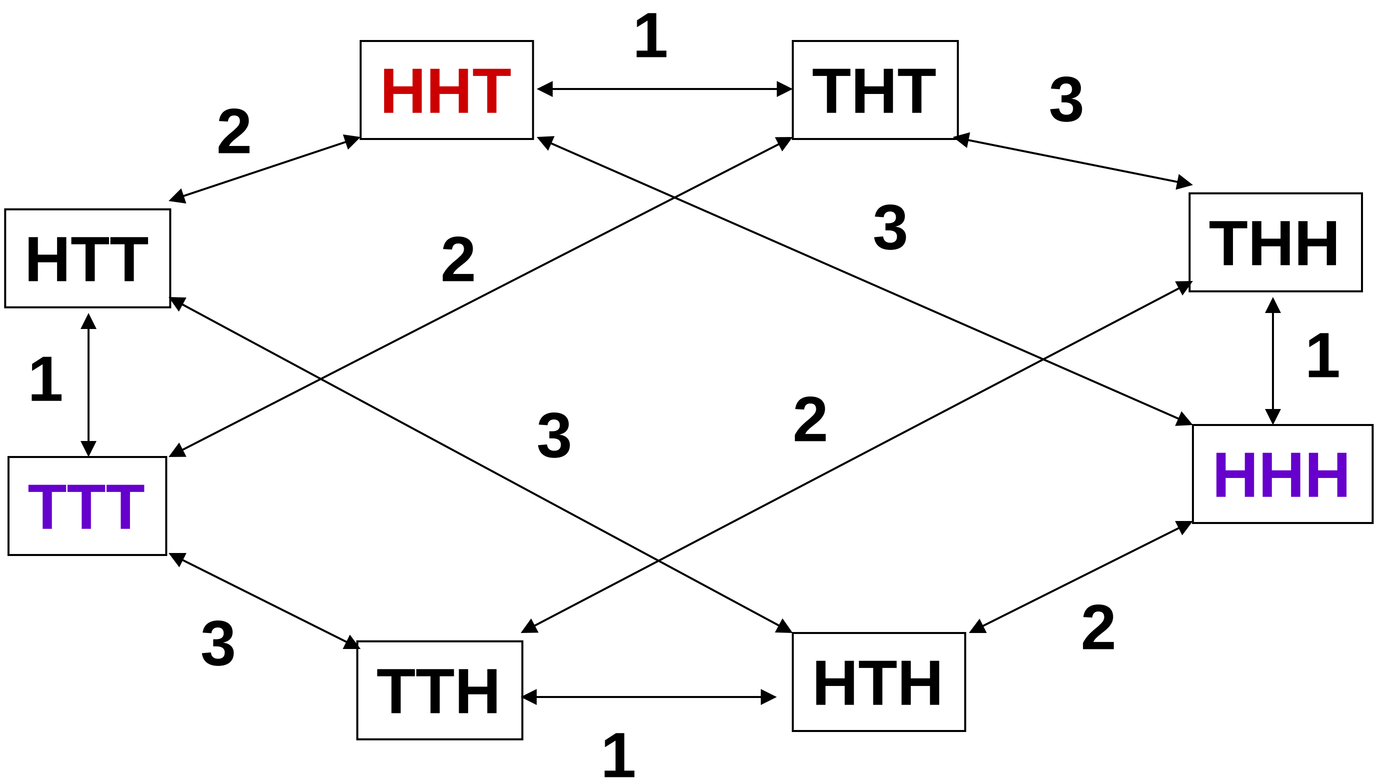
S = state space
s = initial state
A = set of actions
f = state change function
g = goal test function
c = cost function



3 Coins Problem
A Very Small State Space Problem

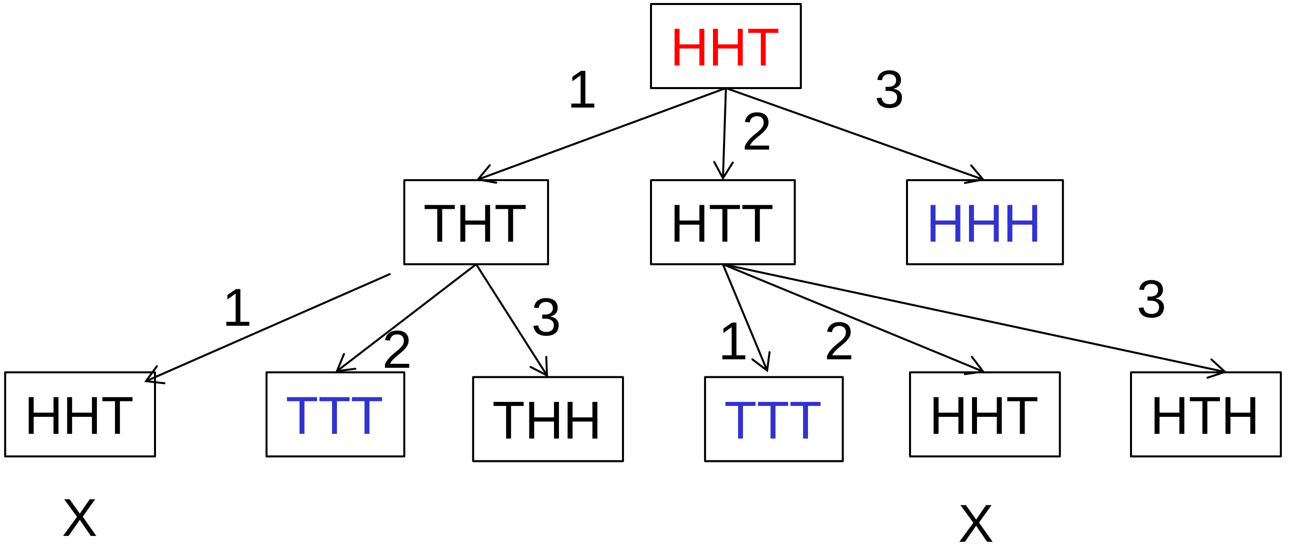
- There are 3 (distinct) coins: coin1, coin2, coin3.
- The initial state is H H T
- The legal operations are to turn over exactly one coin.
  - 1 (flip coin1), 2 (flip coin2), 3 (flip coin3)
- There are two goal states: H H H
  T T T

State-Space Graph



- **What are some solutions?**

Partial Search Tree



The 8-Puzzle Problem

one initial state

1	2	3
8	B	4
7	6	5

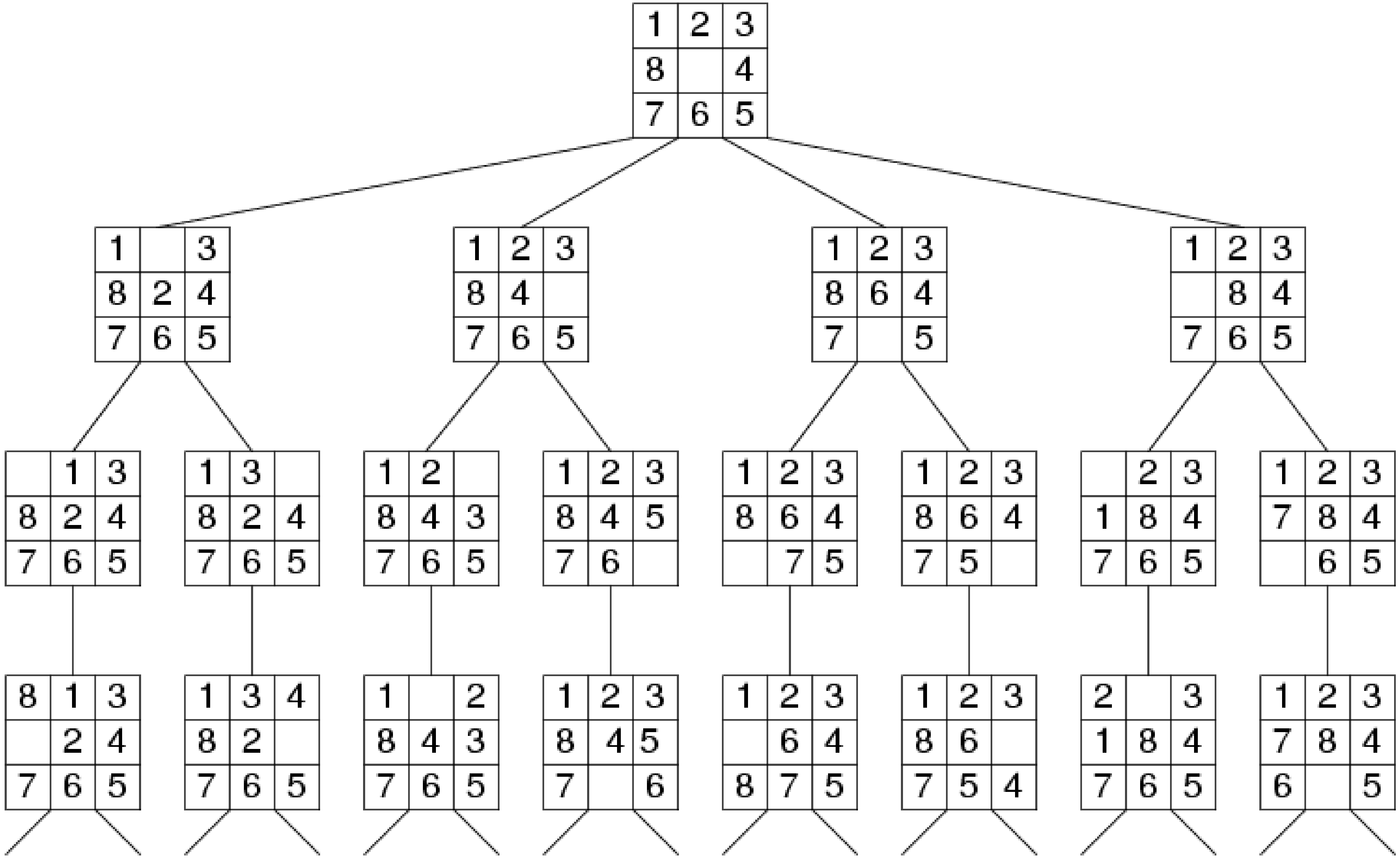
goal state

B	1	2
3	4	5
6	7	8

B=blank

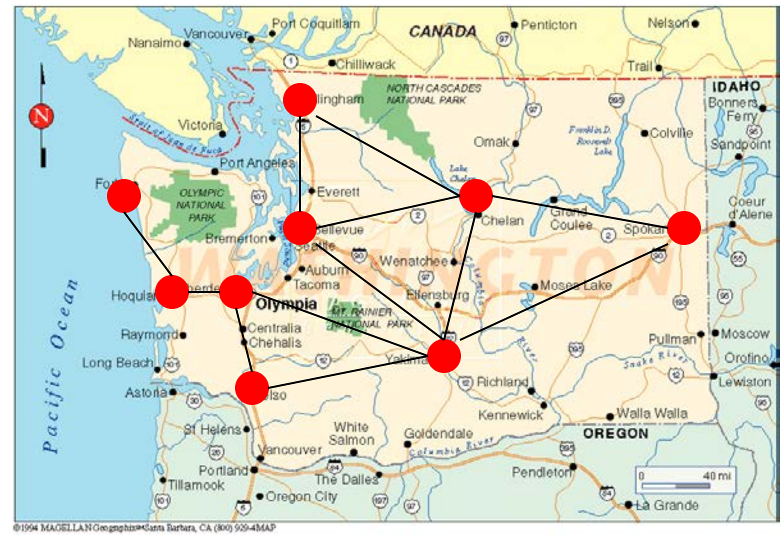
1. **What data structure easily represents a state?**
2. **How many possible states are there?**
3. **What is the complexity of the search?**

Search Tree Example: Fragment of 8-Puzzle Problem Space



Example: Route Planning

Find the shortest route from the starting city to the goal city given roads and distances.



- Input:
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state (test)
- Output:

The travelling salesman problem (TSP) asks the following question:
Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

Search in AI

- Search in Data Structures
  - You’re given an existent tree.
  - You search it in different orders.
  - It resides in memory.
- Search in Artificial Intelligence
  - The tree does not exist.
  - You have to generate it as you go.
  - For realistic problems, it does not fit in memory.

Search Strategies (Ch 3)

- Uninformed Search

  The search is blind, only the order of search is important.

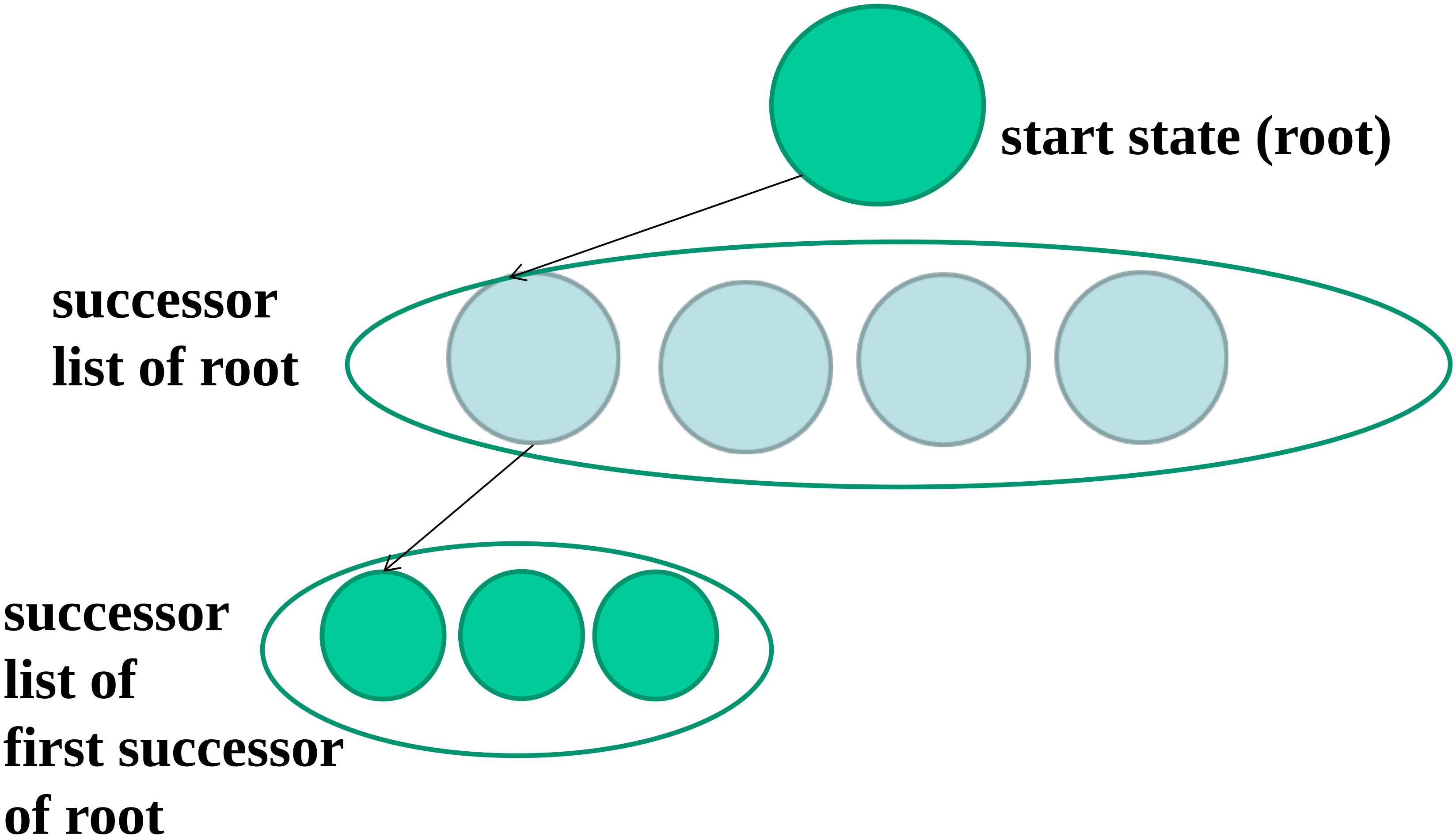
- Informed Search

  The search uses a heuristic function to estimate the goodness of each state.

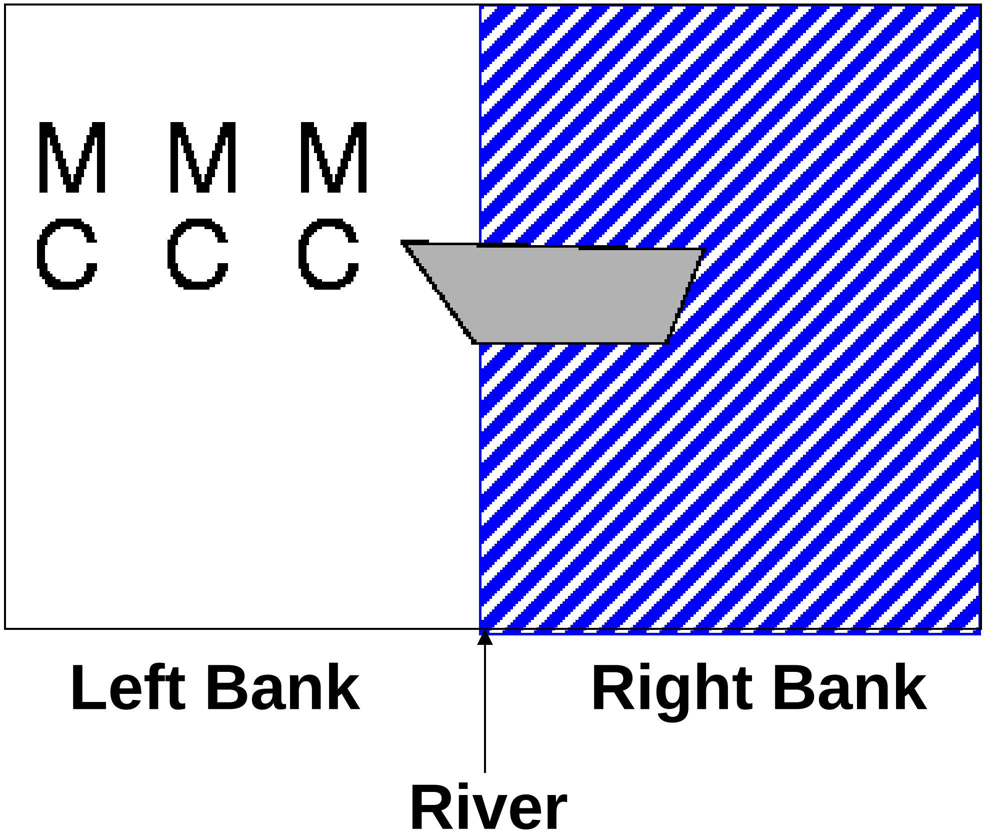
*Depth-First Search by Recursion**

- Search is a recursive procedure that is called with the start node and has arg s.
- It checks first if s is the goal.
- It also checks if s is illegal or too deep.
- If neither, it generates the list L of successors of its argument s.
- It iterates through list L, calling itself recursively for each state in L.

Depth-First Search by Recursion



Missionaries and Cannibals Problem



Missionary and Cannibals Notes

- Define your state as (M,C,S)
  - M: number of missionaries on left bank
  - C: number of cannibals on left bank
  - S: side of the river that the boat is on

- When the boat is moving, we are in between states. When it arrives, everyone gets out.

(3,3,L) → (3,1,R) **What action did I apply?**

What are all the actions?

- Left to right

1. MCR
2. MMR
3. ?
4. ?
5. ?

- Right to left

1. MCL
2. MML
3. ?
4. ?
5. ?

When is a state considered “DEAD”?

1. There are more cannibals than missionaries on the left bank. (Bunga-Bunga)
2. There are more cannibals than missionaries on the right bank. (Bunga-Bunga)
3. There is an ancestor state of this state that is exactly the same as this state. (Why?)

Same Ancestor State

Stack

(3,3,L)

(3,3,L)

(3,1,R)

(3,1,R)

(3,3,L)

X

Graph Matching

Input: 2 digraphs G1 = (V1,E1), G2 = (V2,E2)

Questions to ask:

1. **Are G1 and G2 isomorphic?**

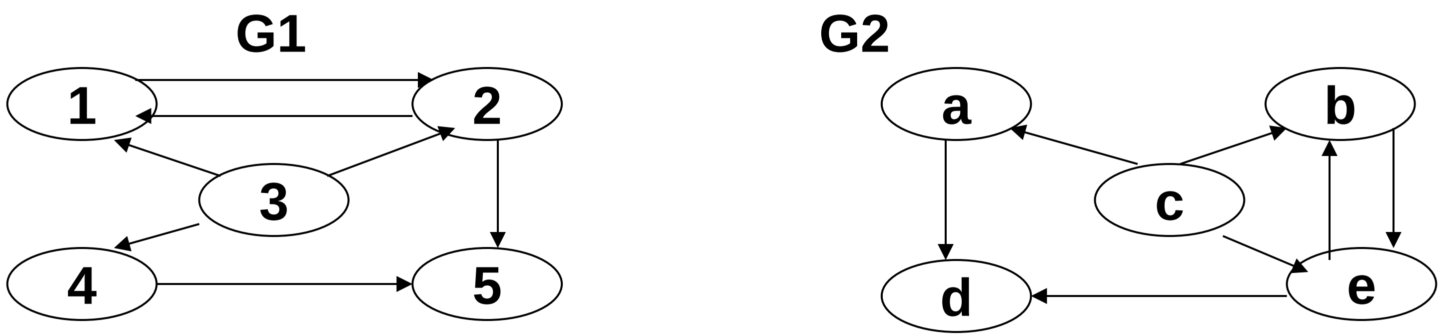
2. **Is G1 isomorphic to a subgraph of G2?**

3. **How similar is G1 to G2?**

4. **How similar is G1 to the most similar subgraph of G2?**

Isomorphism for Digraphs

G1 is isomorphic to G2 if there is a 1-1, onto mapping h: V1 → V2 such that (vi,vj) ∈ E1 iff (h(vi), h(vj)) ∈ E2.



Find an isomorphism h: {1,2,3,4,5} → {a,b,c,d,e}.
Check that the condition holds for every edge.

Answer: h(1)=b, h(2)=e, h(3)=c, h(4)=a, h(5)=d

Isomorphism for Digraphs

G1 is isomorphic to G2 if there is a 1-1, onto mapping h: V1 → V2 such that (vi,vj) ∈ E1 iff (h(vi), h(vj)) ∈ E2

Answer: h(1)=b, h(2)=e, h(3)=c, h(4)=a, h(5)=d
(1,2) ∈ E1 and (h(1),h(2))=(b,e) ∈ E2.
(2,1) ∈ E1 and (e,b) ∈ E2.
(2,5) ∈ E1 and (e,d) ∈ E2.
(3,1) ∈ E1 and (c,b) ∈ E2.
(3,2) ∈ E1 and (c,e) ∈ E2.
...

Subgraph Isomorphism for Digraphs

G1 is isomorphic to a subgraph of G2 if there is a 1-1 mapping $h: V1 \rightarrow V2$ such that $(v_i, v_j) \in E1 \Rightarrow (h(v_i), h(v_j)) \in E2$.

Isomorphism and subgraph isomorphism are defined similarly for undirected graphs.

In this case, when $(v_i, v_j) \in E1$, either (v_i, v_j) or (v_j, v_i) can be listed in E2, since they are equivalent and both mean $\{v_i, v_j\}$.

Subgraph Isomorphism for Graphs

G1 is isomorphic to a **subgraph** of **G2** if there is a 1-1 mapping $h: V1 \rightarrow V2$ such that $\{vi,vj\} \in E1 \Rightarrow \{h(vi), h(vj)\} \in E2.$

Because there are no directed edges, there are more possible mappings.

1	2	3	
c	b	d	
c	d	b	(shown on graph)
b	c	d	
b	d	c	
d	b	c	
d	c	b	

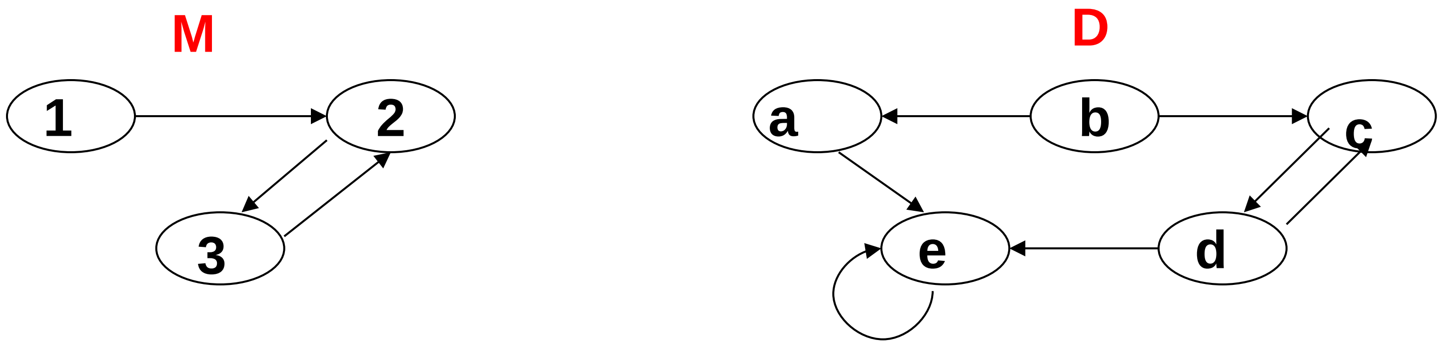
Graph Matching Algorithms: Subgraph Isomorphism for Digraph

Given model graph $M = (VM,EM)$
data graph $D = (VD,ED)$

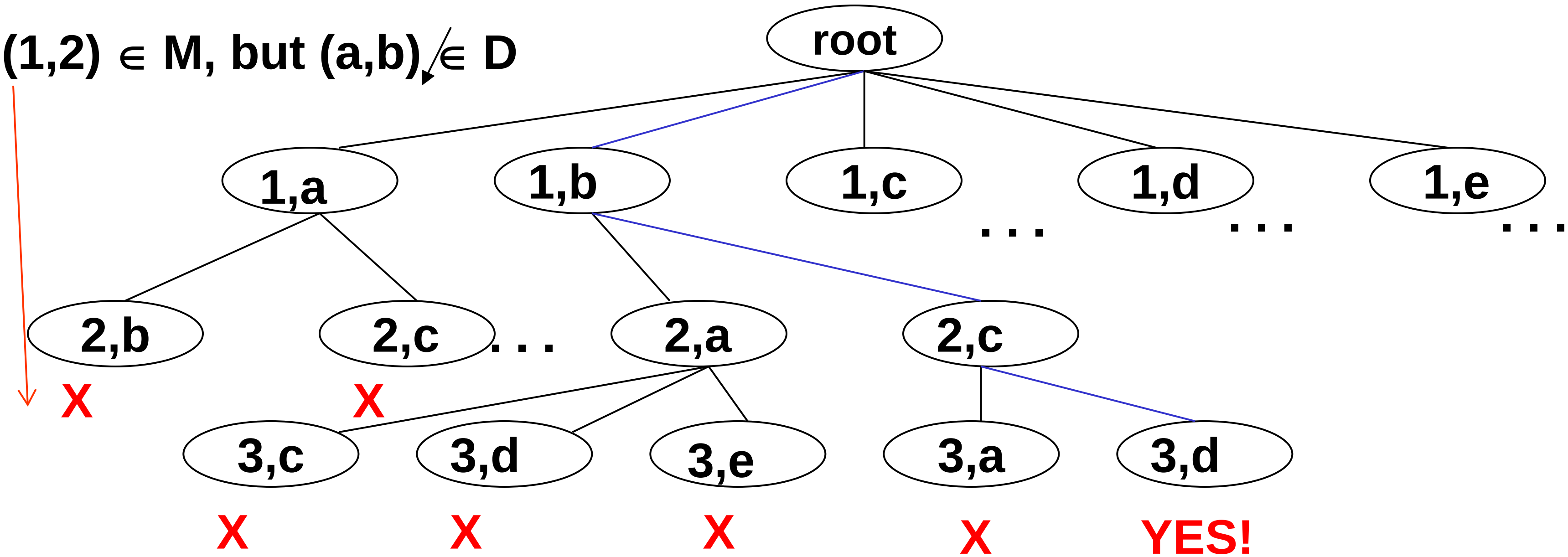
Find 1-1 mapping $h:VM \rightarrow VD$

satisfying $(vi,vj) \in EM \Rightarrow ((h(vi),h(vj)) \in ED$.

Method: Recursive Backtracking Tree Search (Order is depth first, leftmost child first.)

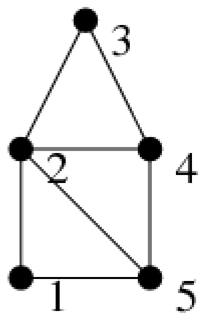


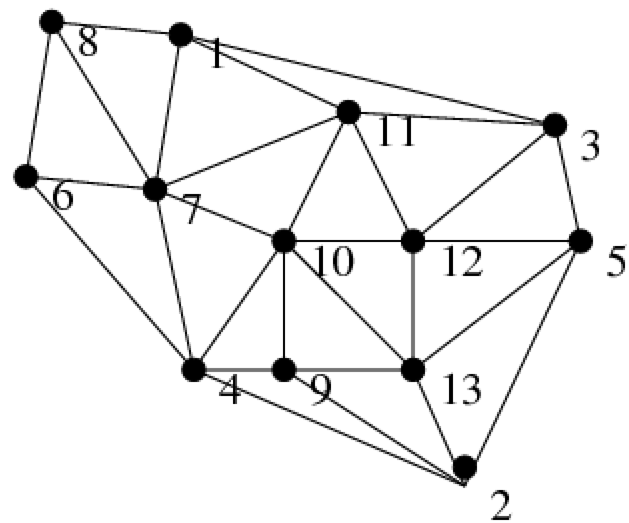
(1,2) ∈ M, but (a,b) ∉ D



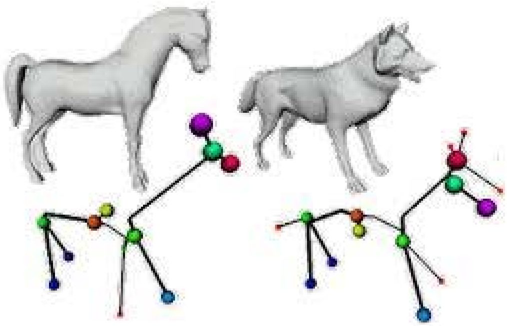
Application to Computer Vision

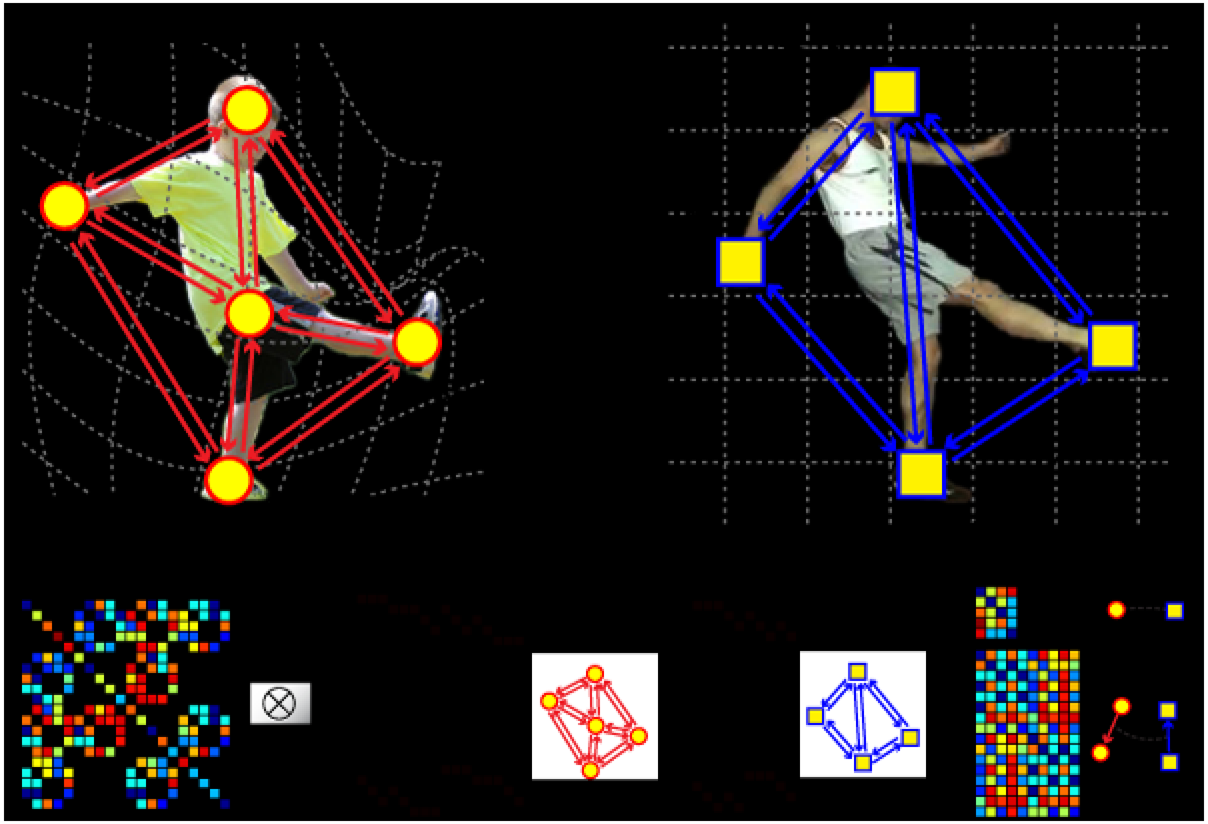
Find the house model in the image graph.





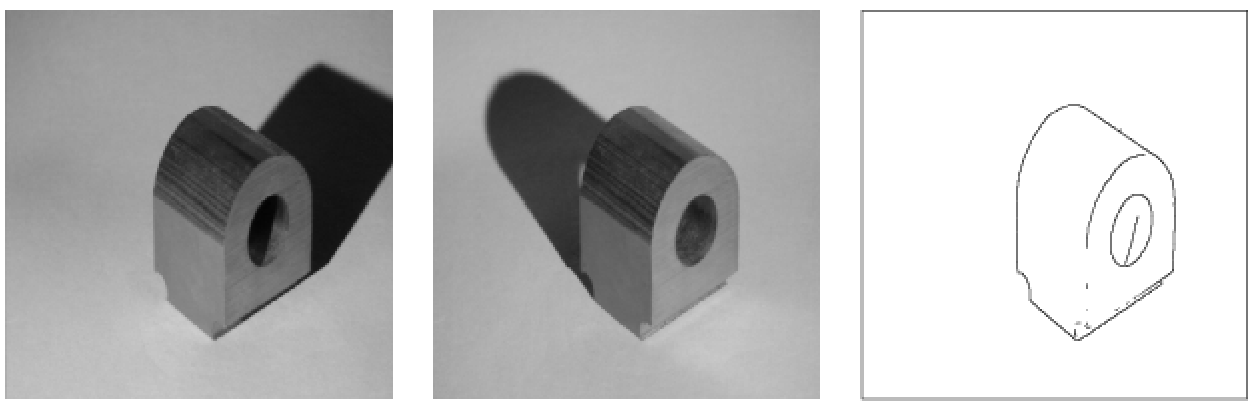
More Examples





RIO: Relational Indexing for Object Recognition

- **RIO worked with industrial parts that could have**
  - planar surfaces
  - cylindrical surfaces
  - threads



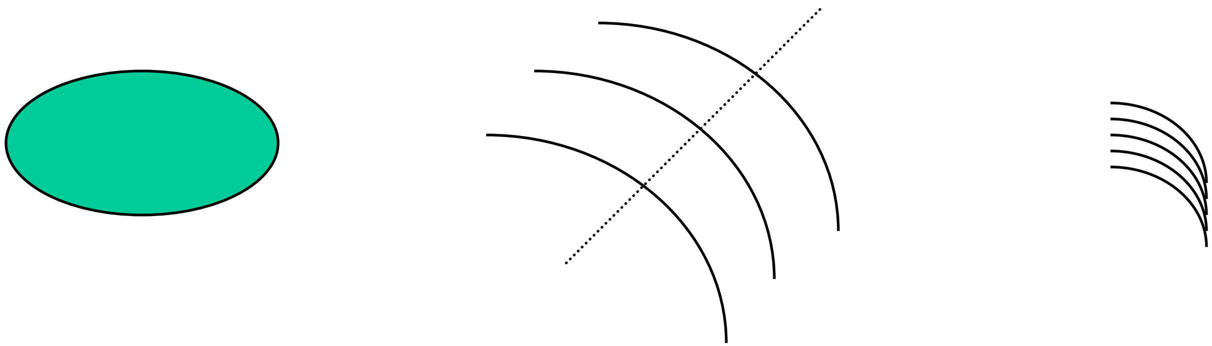
Object Representation in RIO

- **3D objects are represented by a 3D mesh and set of 2D view classes.**

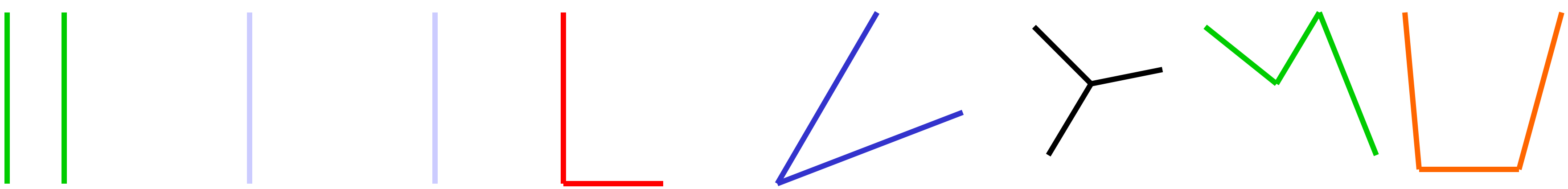
- **Each view class is represented by an attributed graph whose nodes are features and whose attributed edges are relationships.**

- **Graph matching is done through an indexing method.**

RIO Features



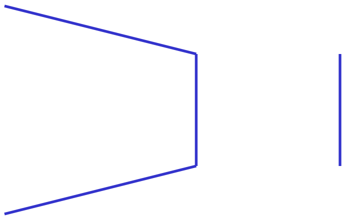
ellipses **coaxials** **coaxials-multi**



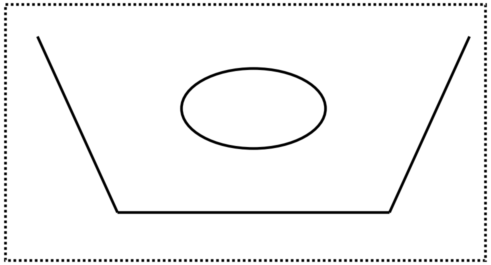
parallel lines **junctions** **triples**

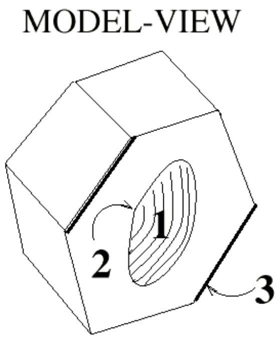
close and far **L** **V** **Y** **Z** **U**

RIO Relationships

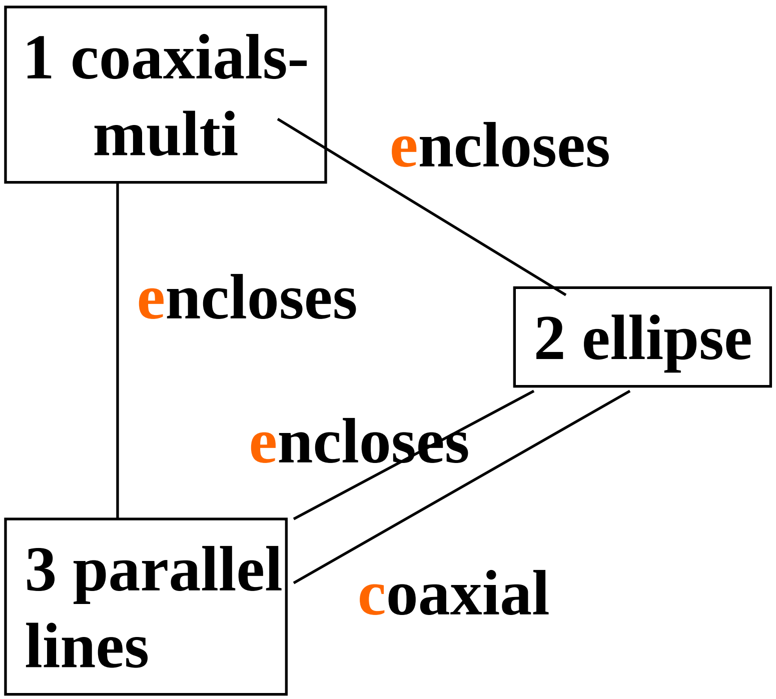


- **share one arc**
- **share one line**
- **share two lines**
- **coaxial**
- **close at extremal points**
- **bounding box encloses / enclosed by**





Graph Representation



This is just a piece of the whole graph.

Relational Indexing for Recognition

Preprocessing (off-line) Phase

for each model view M_i in the database

- **encode each 2-graph of M_i to produce an index**
- **store M_i and associated information in the indexed bin of a hash table H**

Matching (on-line) phase

1. **Construct a relational (2-graph) description D for the scene**

2. **For each 2-graph G of D**

   - **encode it, producing an index to access the hash table H**

   - **cast a vote for each M_i in the associated bin**

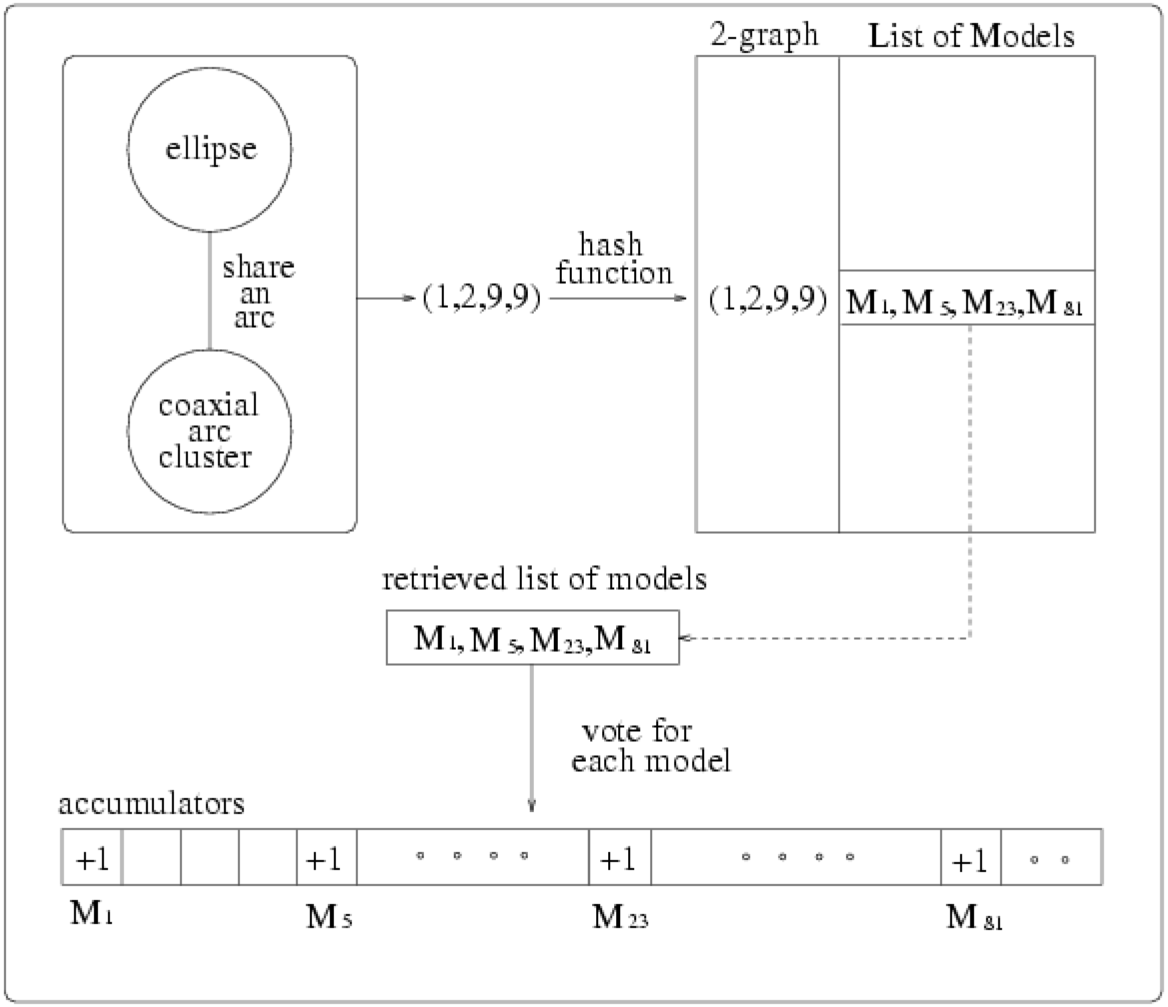
3. **Select the M_is with high votes as possible hypotheses**

4. **Verify or disprove via alignment, using the 3D meshes**

The Voting Process

hash table

two related features from an image



array of accumulators to vote for models

Verification

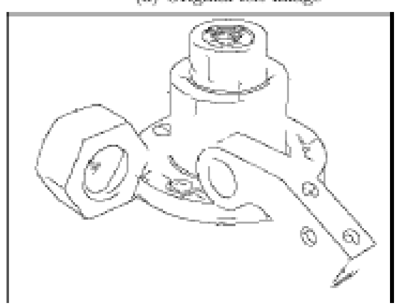
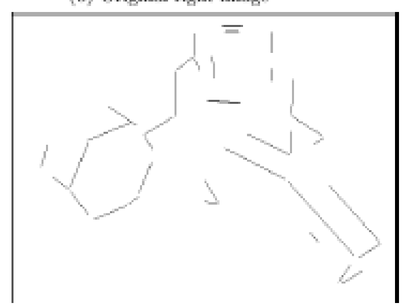
1. **The matched features of the hypothesized object are used to determine its pose. Pose is computed from correspondences between 2D and 3D points, lines, and circles.**

2. **The 3D mesh of the object is used to project all its features onto the image using perspective projection and hidden feature removal.**

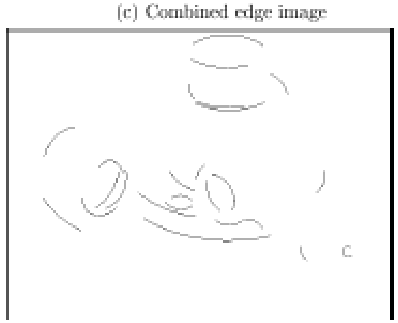
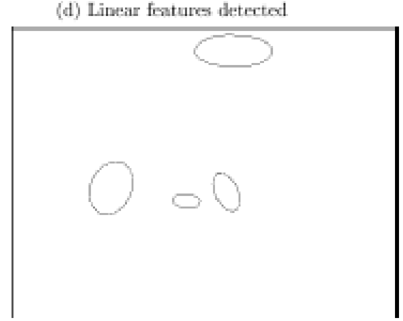
3. **A verification procedure checks how well the object features line up with edges on the image.**

Feature Extraction


(a) Original left image (b) Original right image

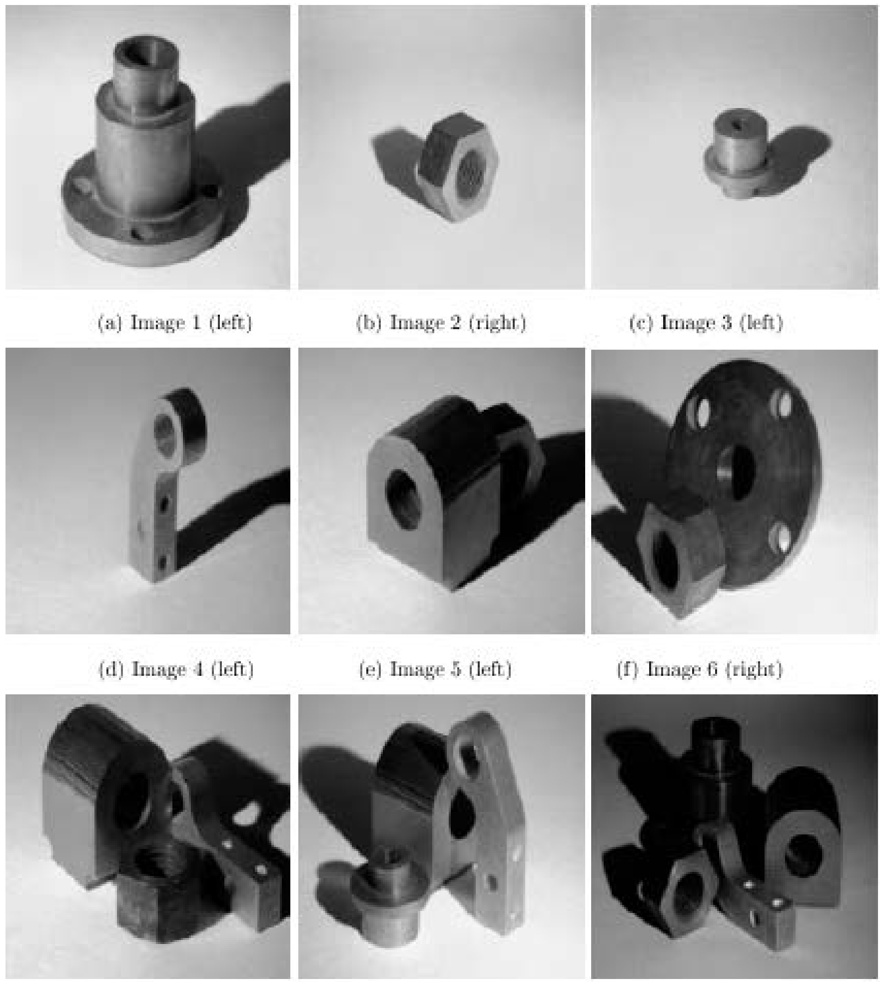



(c) Combined edge image (d) Linear features detected

(e) Circular arc features detected (f) Ellipses detected

Some Test Scenes

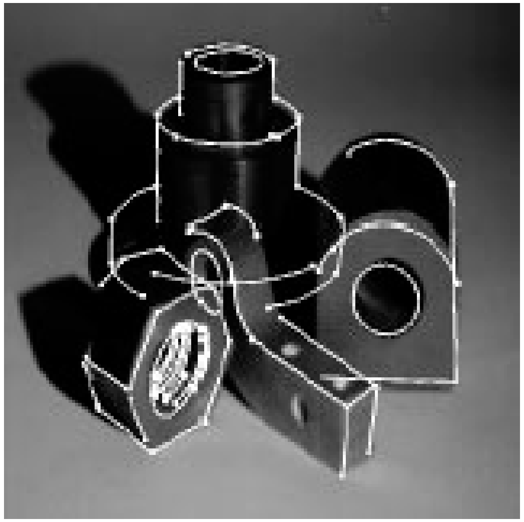


(a) Image 1 (left) (b) Image 2 (right) (c) Image 3 (left)

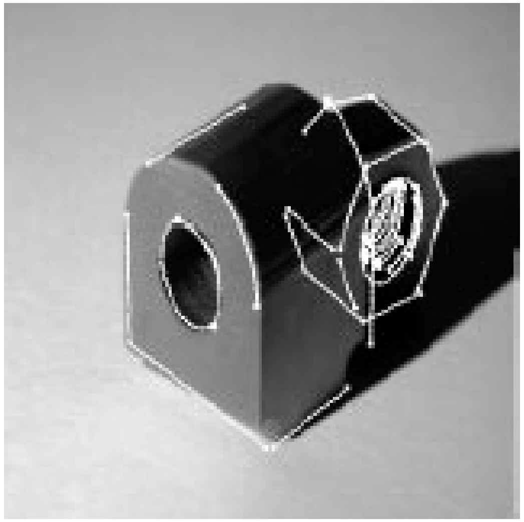
(d) Image 4 (left) (e) Image 5 (left) (f) Image 6 (right)

(g) Image 7 (left) (h) Image 8 (right) (i) Image 9 (right)

Sample Alignments
3D to 2D Perspective Projection



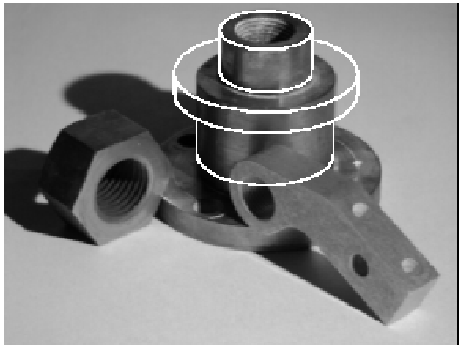
(a)

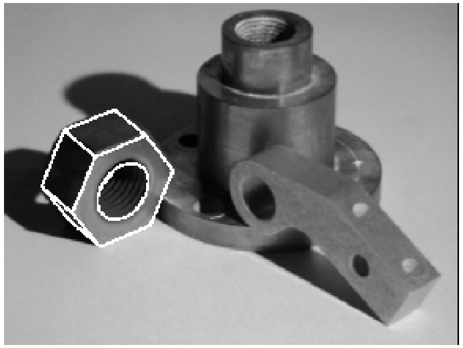


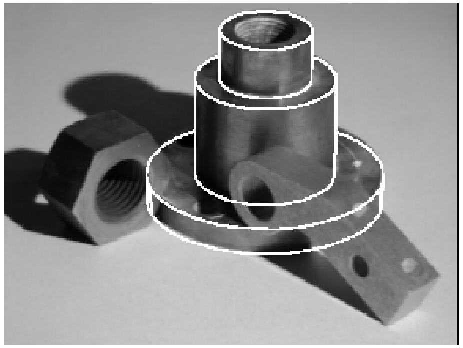
(b)

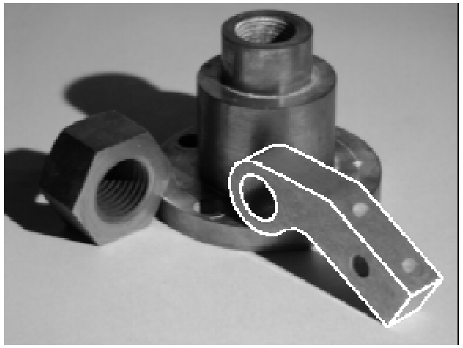
RIO Verifications

incorrect hypothesis



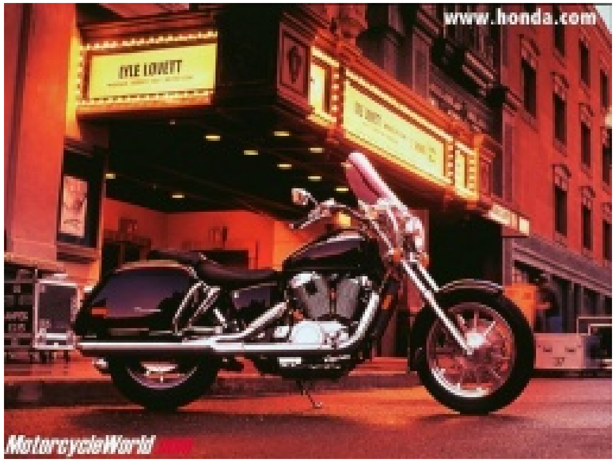




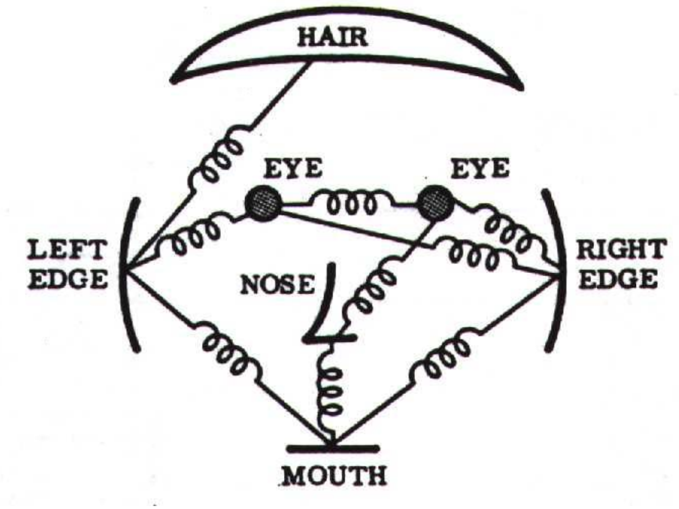


Fergus Object Recognition by Parts:

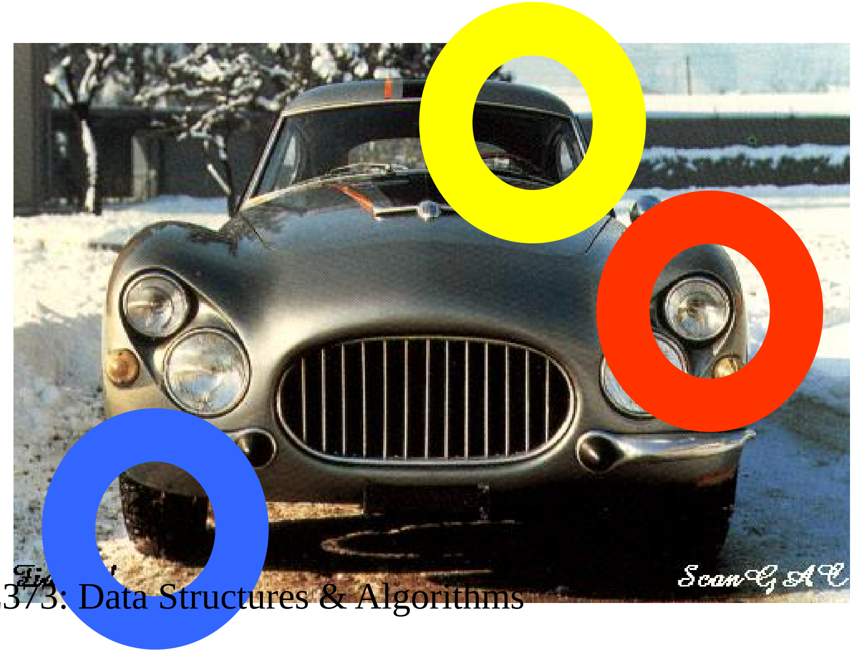
- Enable Computers to Recognize Different Categories of Objects in Images.



Model: Constellation Of Parts

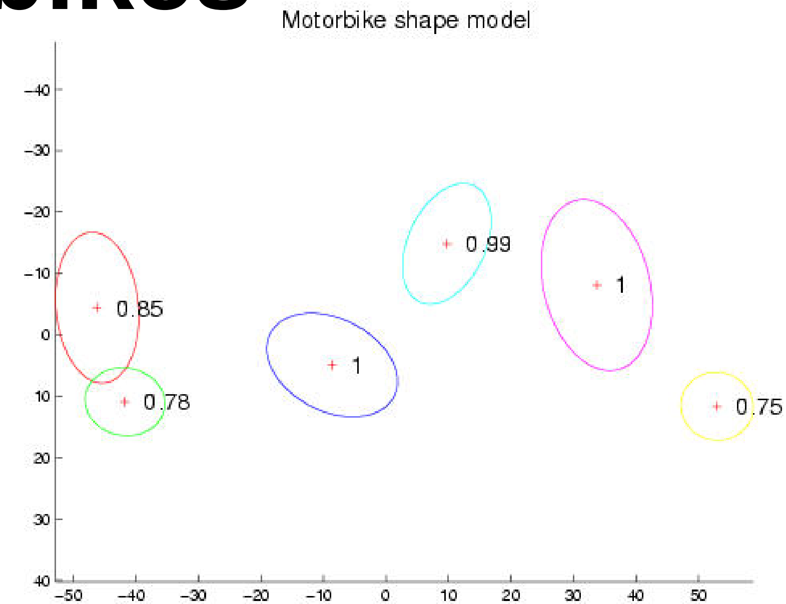


Fischler & Elschlager, 1973



Motorbikes

Part 1 – Det:5e–18
Part 2 – Det:8e–22
Part 3 – Det:6e–18
Part 4 – Det:1e–19
Part 5 – Det:3e–17
Part 6 – Det:4e–24
Background – Det:5e–19



Motorbike shape model
0.85
0.78
1
0.99
1
0.75

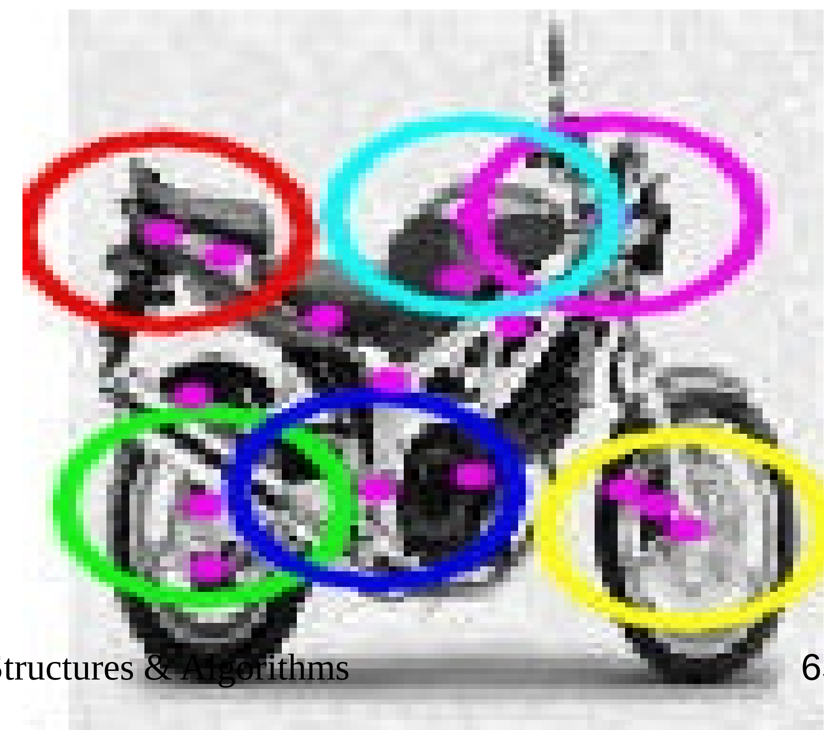
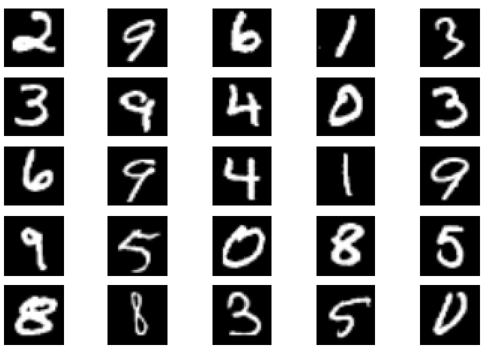


Image classification

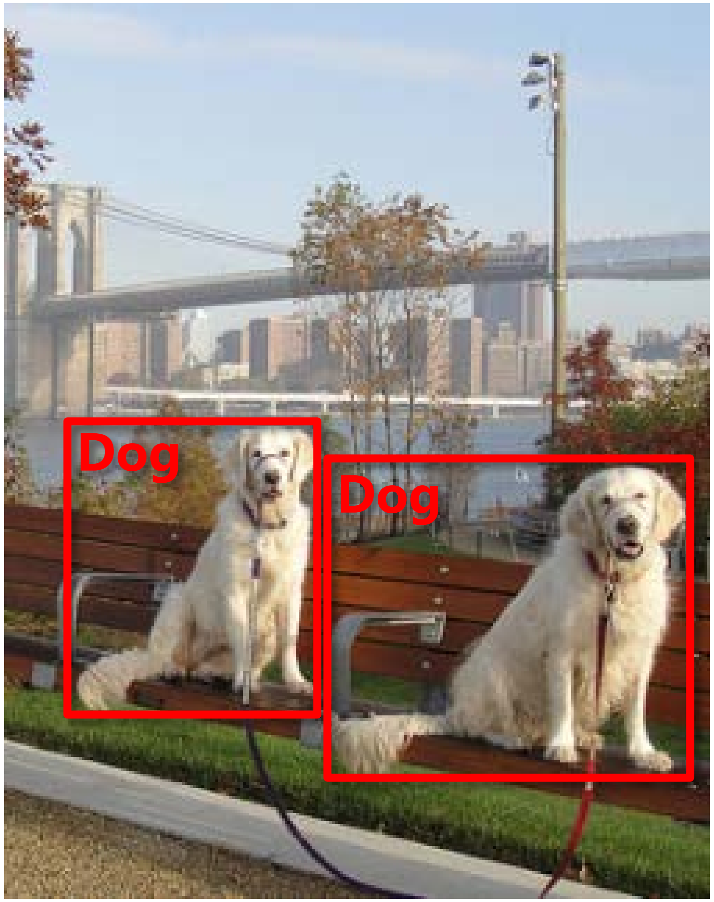
- K classes
- Task: assign correct class label to the whole image



Digit classification (MNIST)

Object recognition (Caltech-10

Classification vs. Detection



Generic categories

Can we detect people, chairs, horses, cars, dogs, buses, bottles, sheep …?

PASCAL Visual Object Categories (VOC) dataset

Quiz time

Warm up

This is an average image of which object class?

Warm up

pedestrian

A little harder

?

A little harder

?

Hint: airplane, bicycle, bus, car, cat, chair, cow, dog, dining table

A little harder

bicycle (PASCAL)

A little harder, yet

?

A little harder, yet

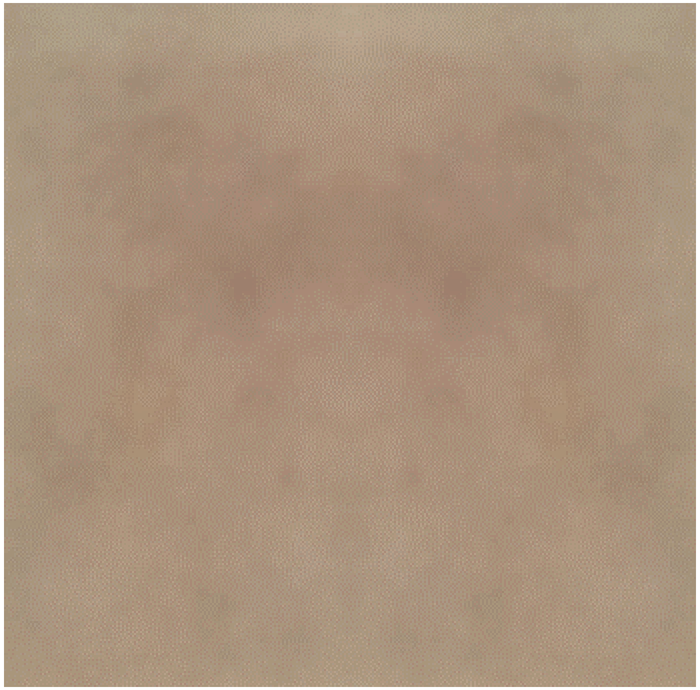
?

Hint: white blob on a green background

A little harder, yet

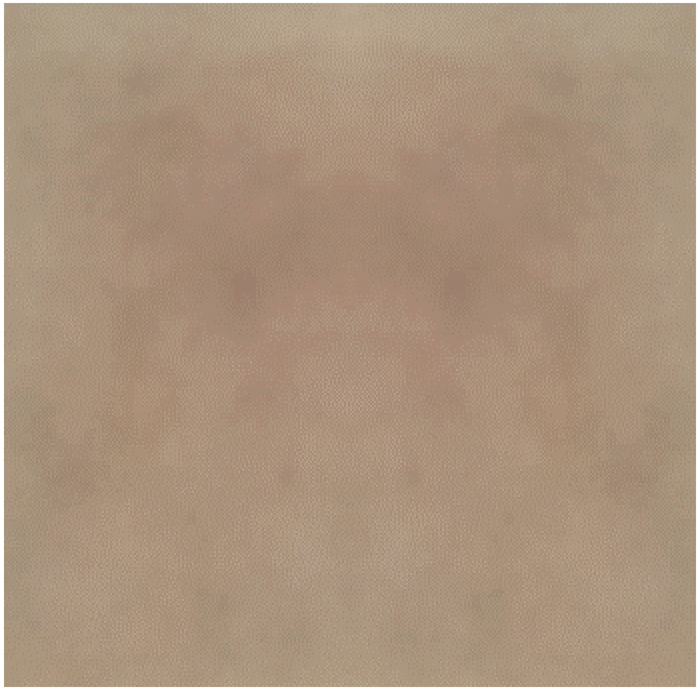
sheep (PASCAL)

Impossible?



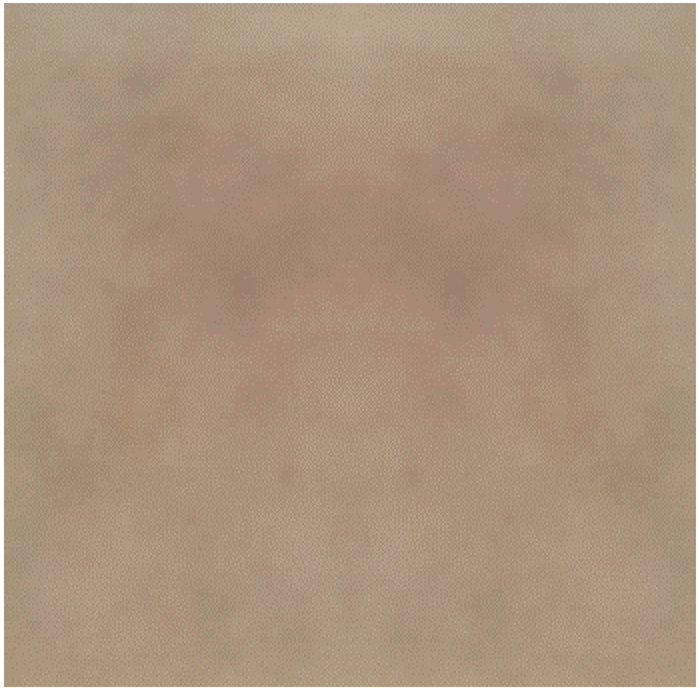
?

Impossible?



dog (PASCAL)

Impossible?



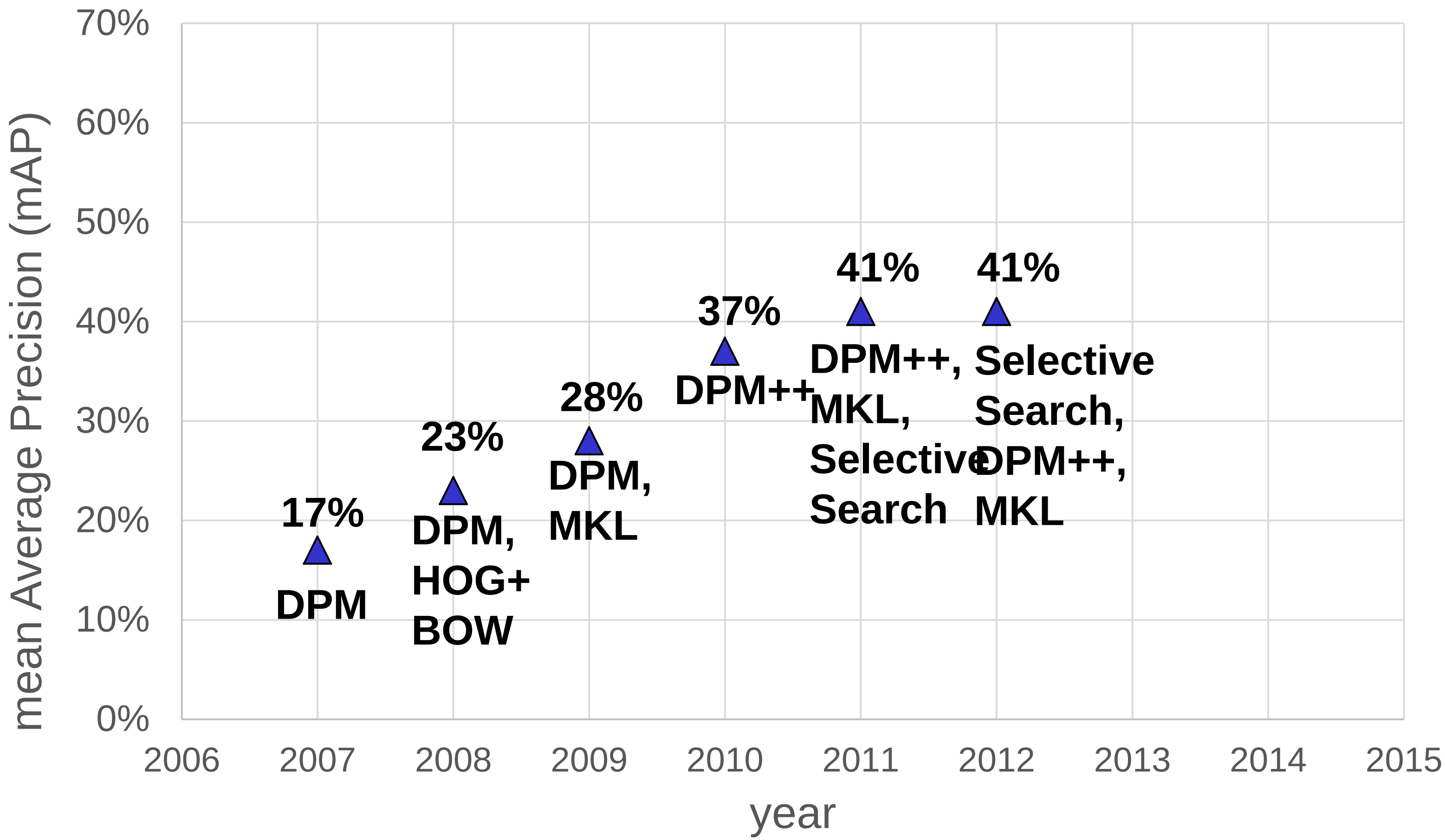
dog (PASCAL)

Why does the mean look like this?

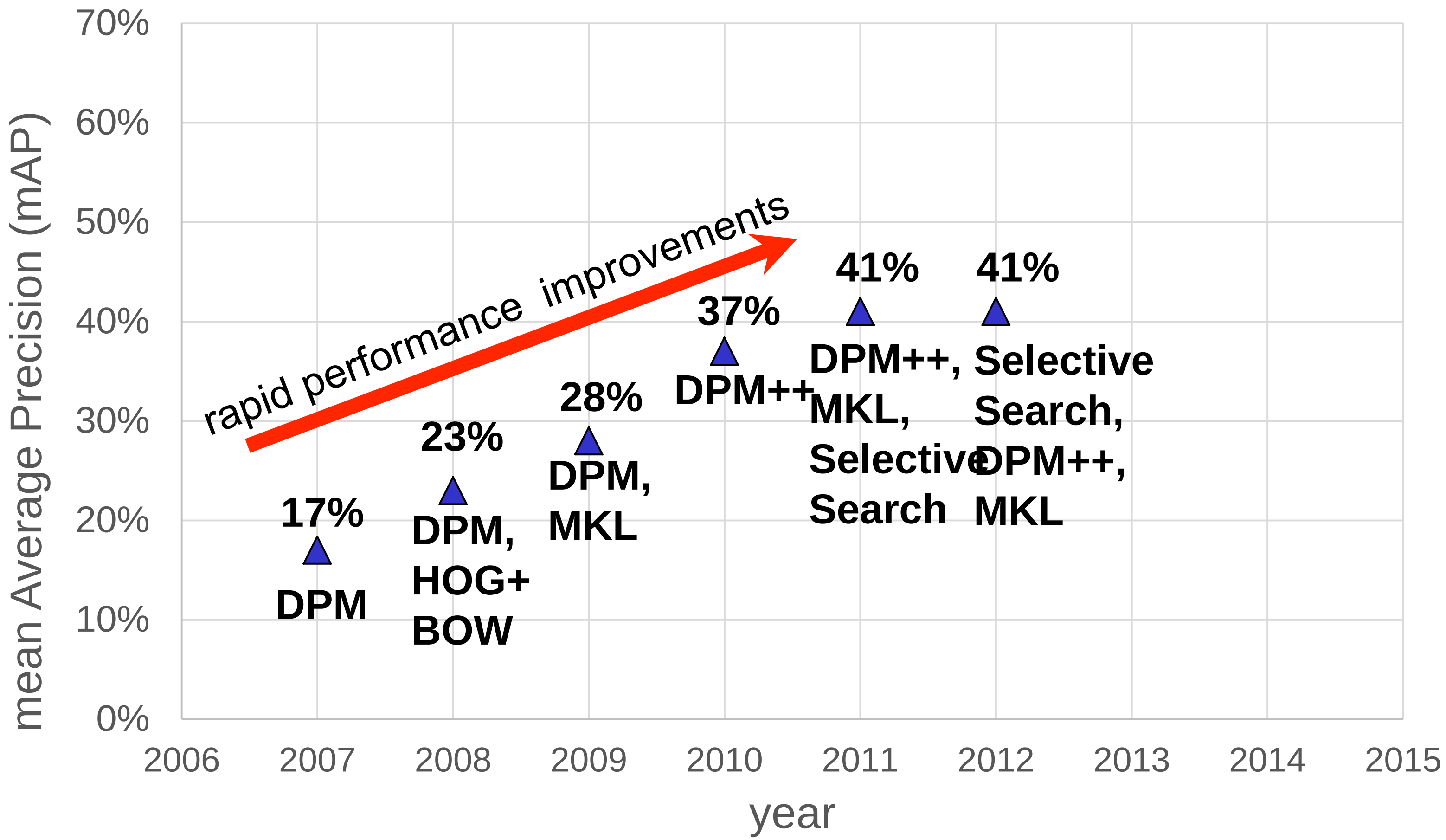
There's no alignment between the examples!

How do we combat this?

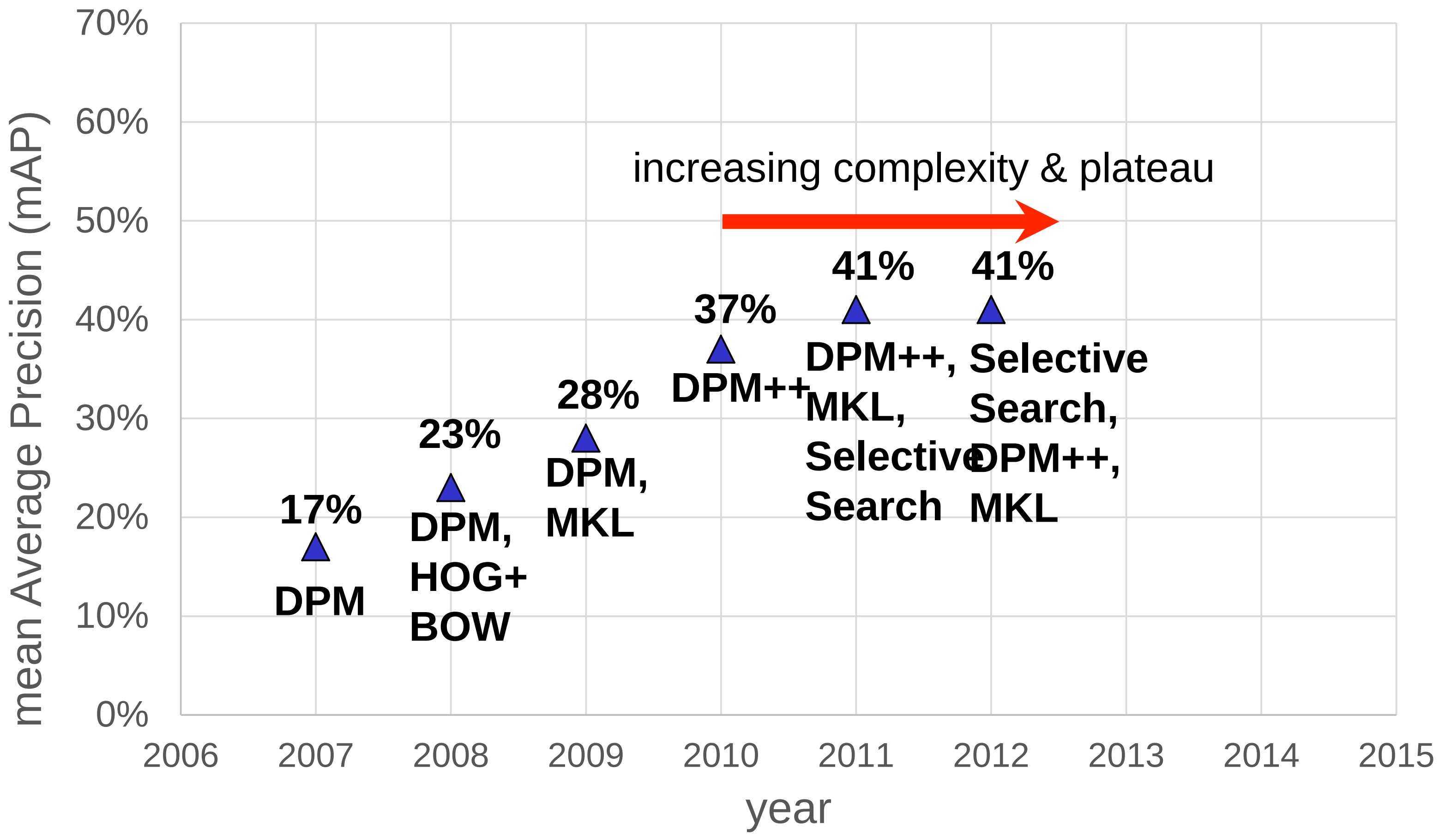
PASCAL VOC detection history



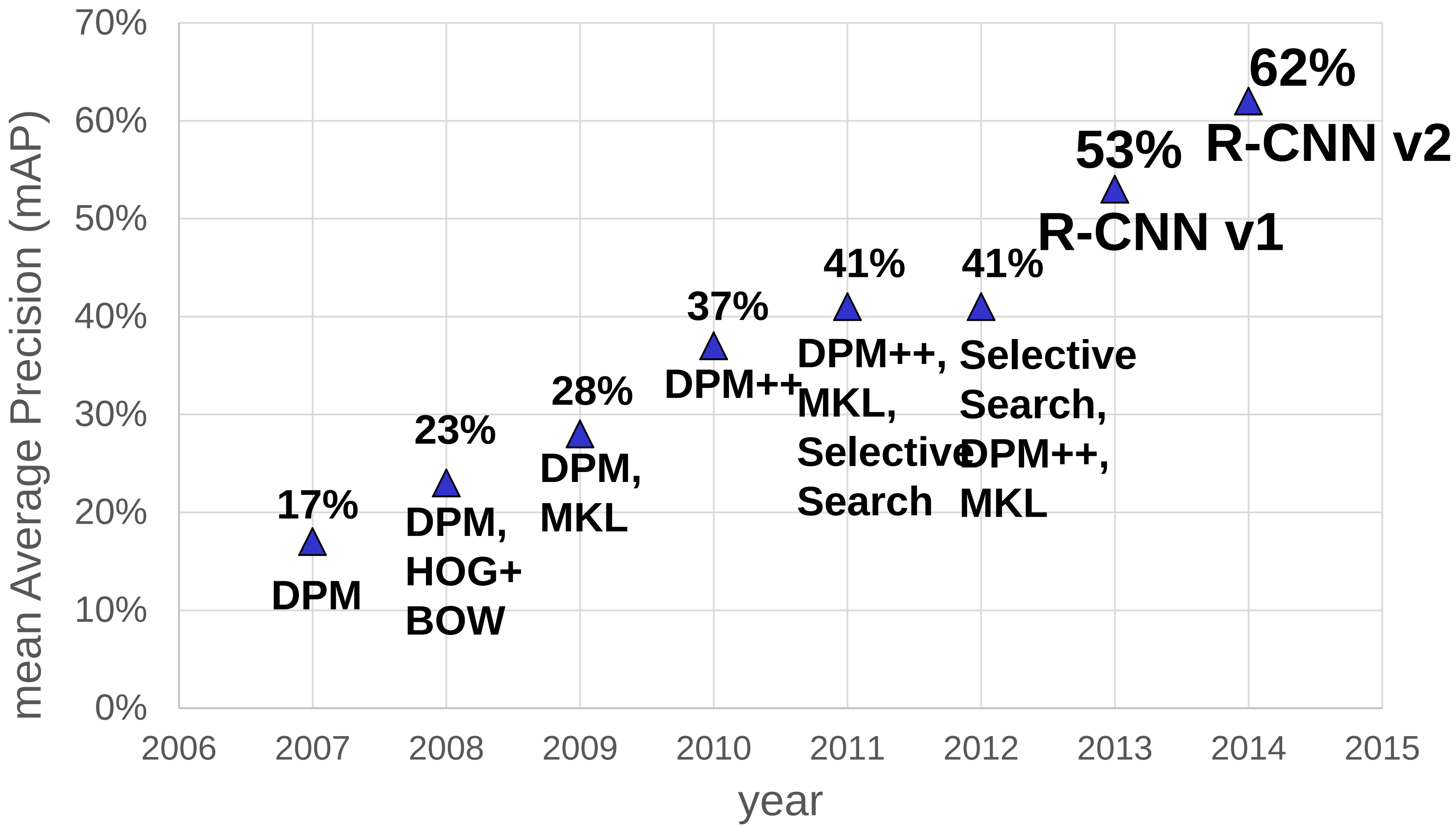
Part-based models & multiple features (MKL)



Kitchen-sink approaches

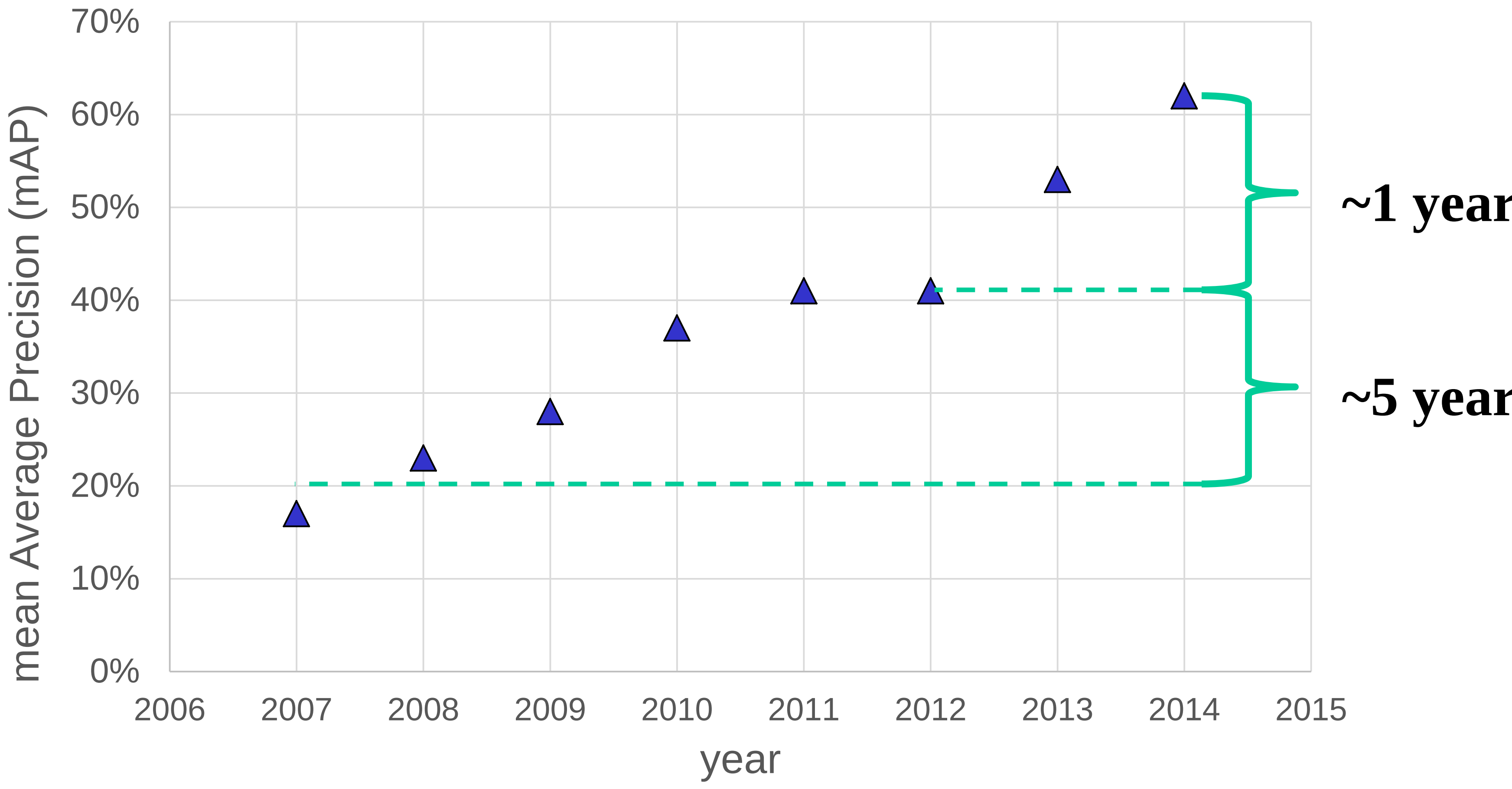


Region-based Convolutional Networks (R-CNNs)



[R-CNN. Girshick et al. CVPR 2014]

Region-based Convolutional Networks (R-CNNs)



[R-CNN. Girshick et al. CVPR 2014]

Convolutional Neural Networks

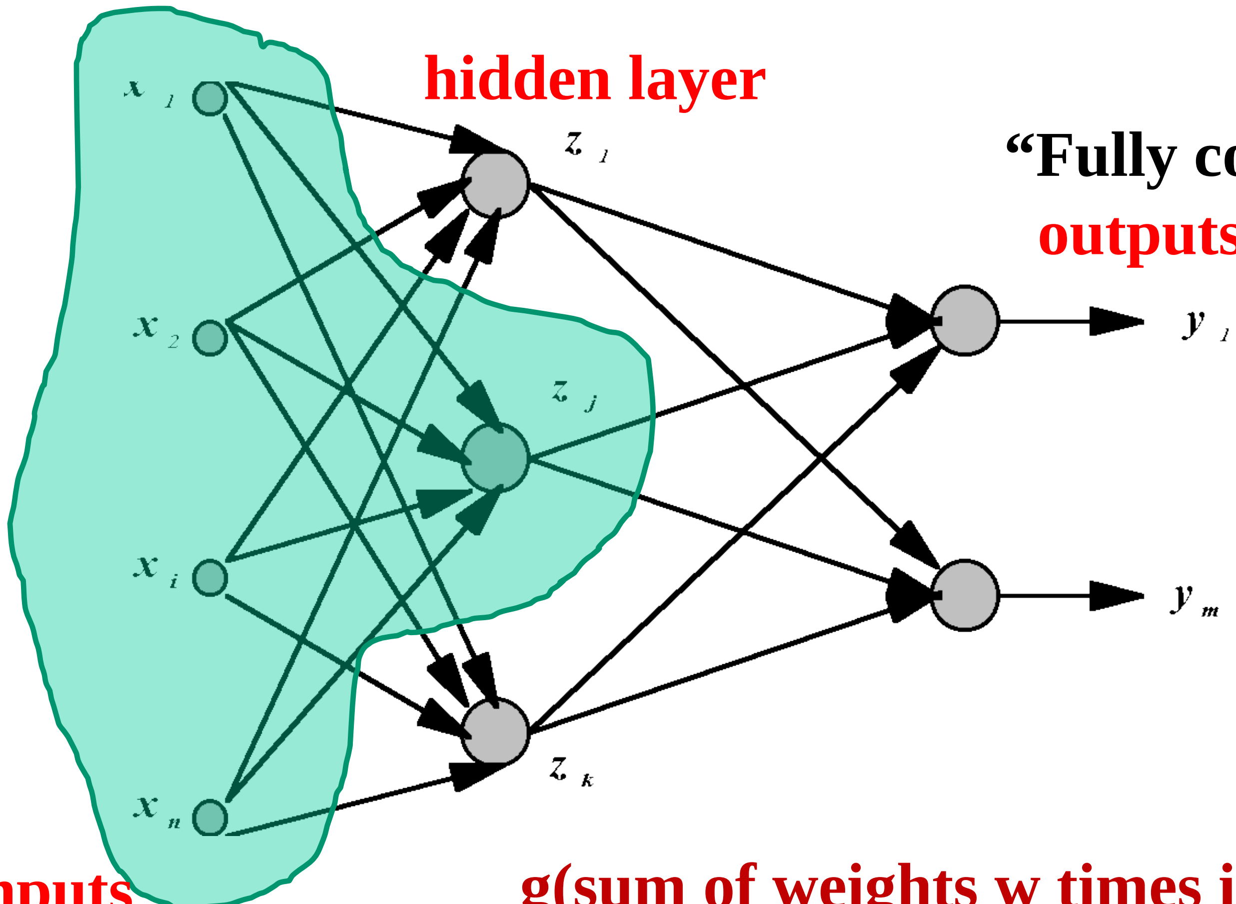
- Overview

Standard Neural Networks

hidden layer

"Fully connected"

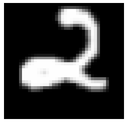
outputs

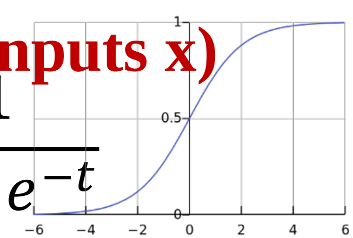


inputs

g(sum of weights w times inputs x)

$\boldsymbol{x} = (x_1, \ldots, x_{784})^T \quad z_j = g(\boldsymbol{w}_j^T \boldsymbol{x}) \quad g(t) = \frac{1}{1 + e^{-t}}$



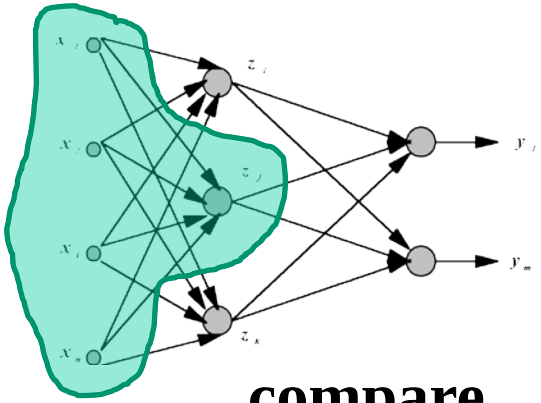


From NNs to Convolutional NNs

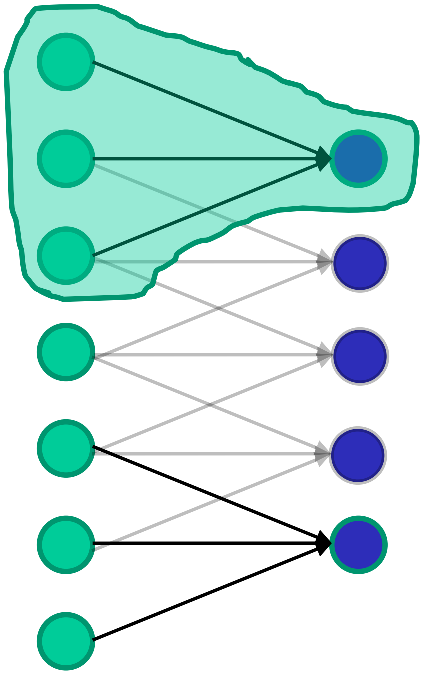
- Local connectivity
- Shared (“tied”) weights
- Multiple feature maps
- Pooling

Convolutional NNs

- Local connectivity



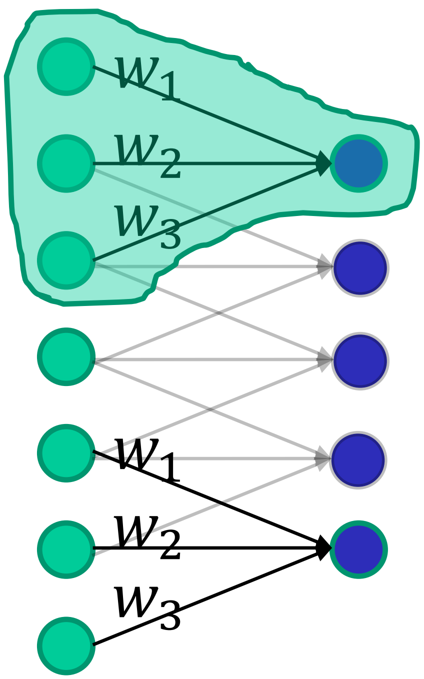
compare



- **Each green unit is only connected to (3) neighboring blue units**

Convolutional NNs

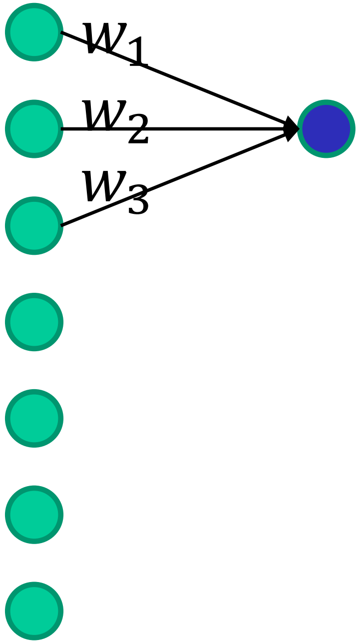
- Shared ("tied") weights



- **All green units share the same parameters *w***
- **Each green unit computes the same function, but with a different input window**

Convolutional NNs

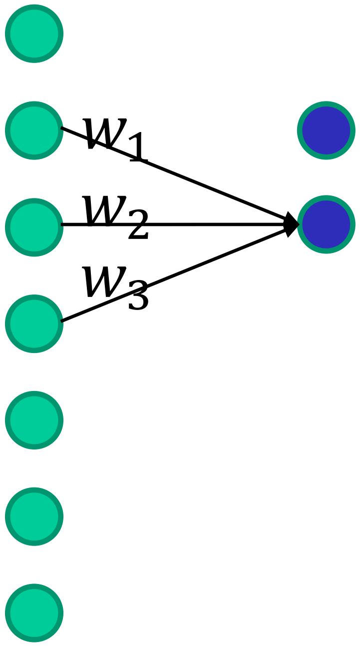
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- **All green units share the same parameters *w***
- **Each green unit computes the same function, but with a different input window**

Convolutional NNs

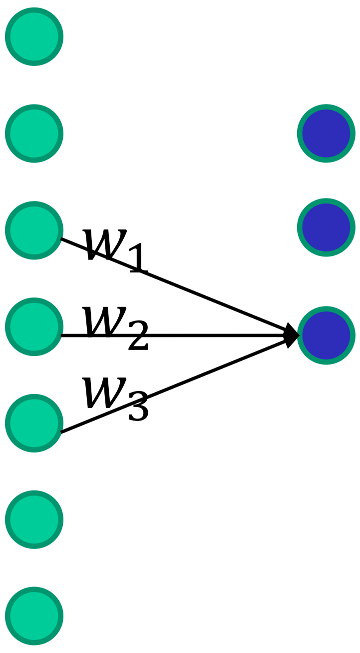
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- **All green units share the same parameters *w***
- **Each green unit computes the same function, but with a different input window**

Convolutional NNs

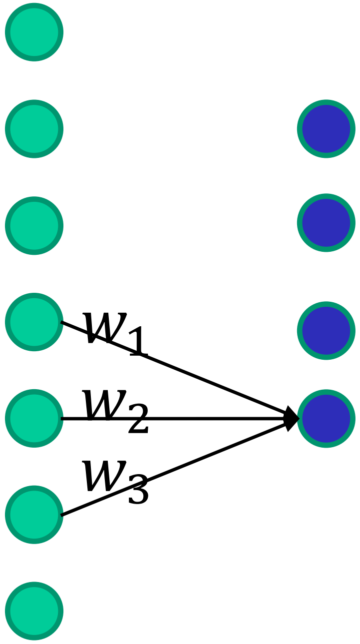
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- **All green units share the same parameters *w***
- **Each green unit computes the same function, but with a different input window**

Convolutional NNs

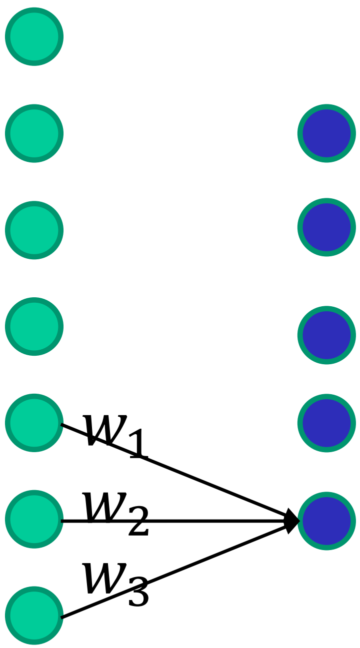
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- **All green units share the same parameters $\boldsymbol{w}$**
- **Each green unit computes the same function, but with a different input window**

Convolutional NNs

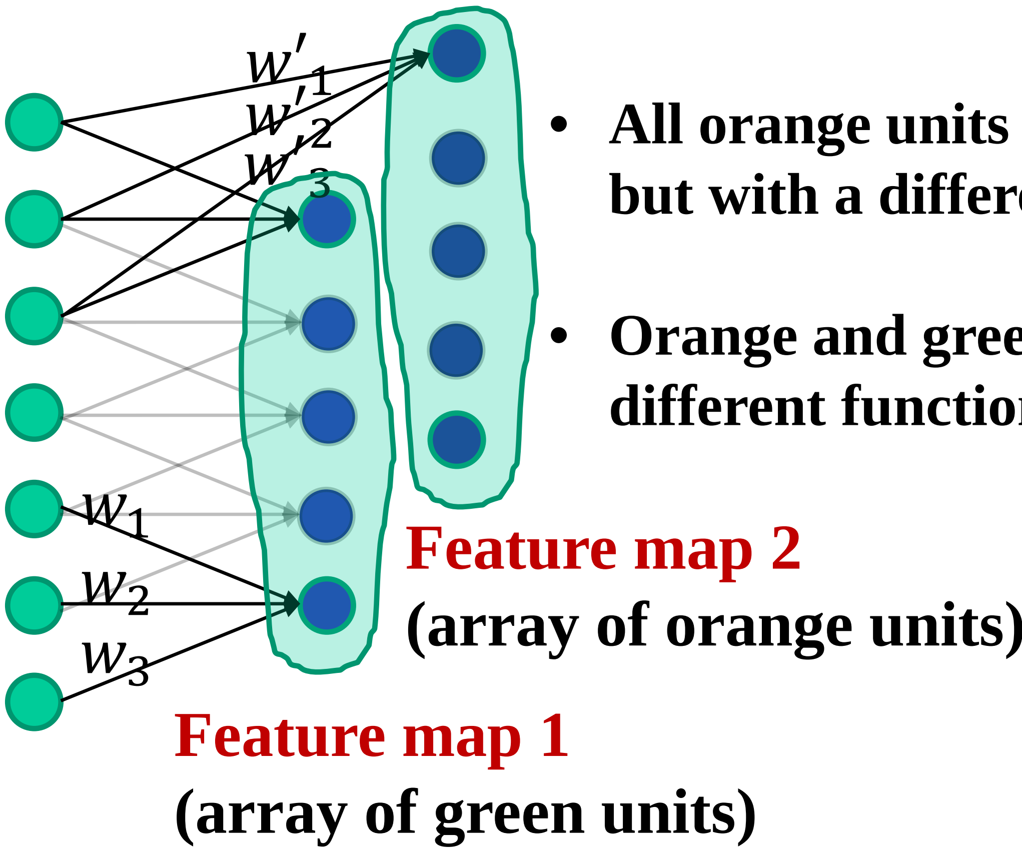
- Convolution with 1-D filter: $[w_3, w_2, w_1]$



- **All green units share the same parameters $\boldsymbol{w}$**
- **Each green unit computes the same function, but with a different input window**

Convolutional NNs

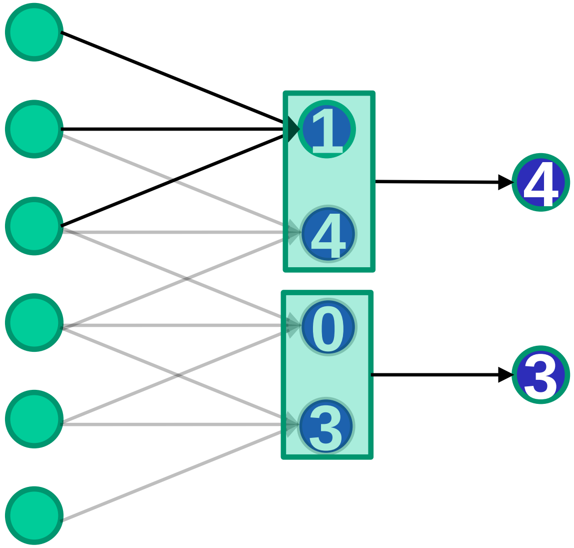
- Multiple feature maps



- **All orange units compute the same function but with a different input windows**
- **Orange and green units compute different functions**

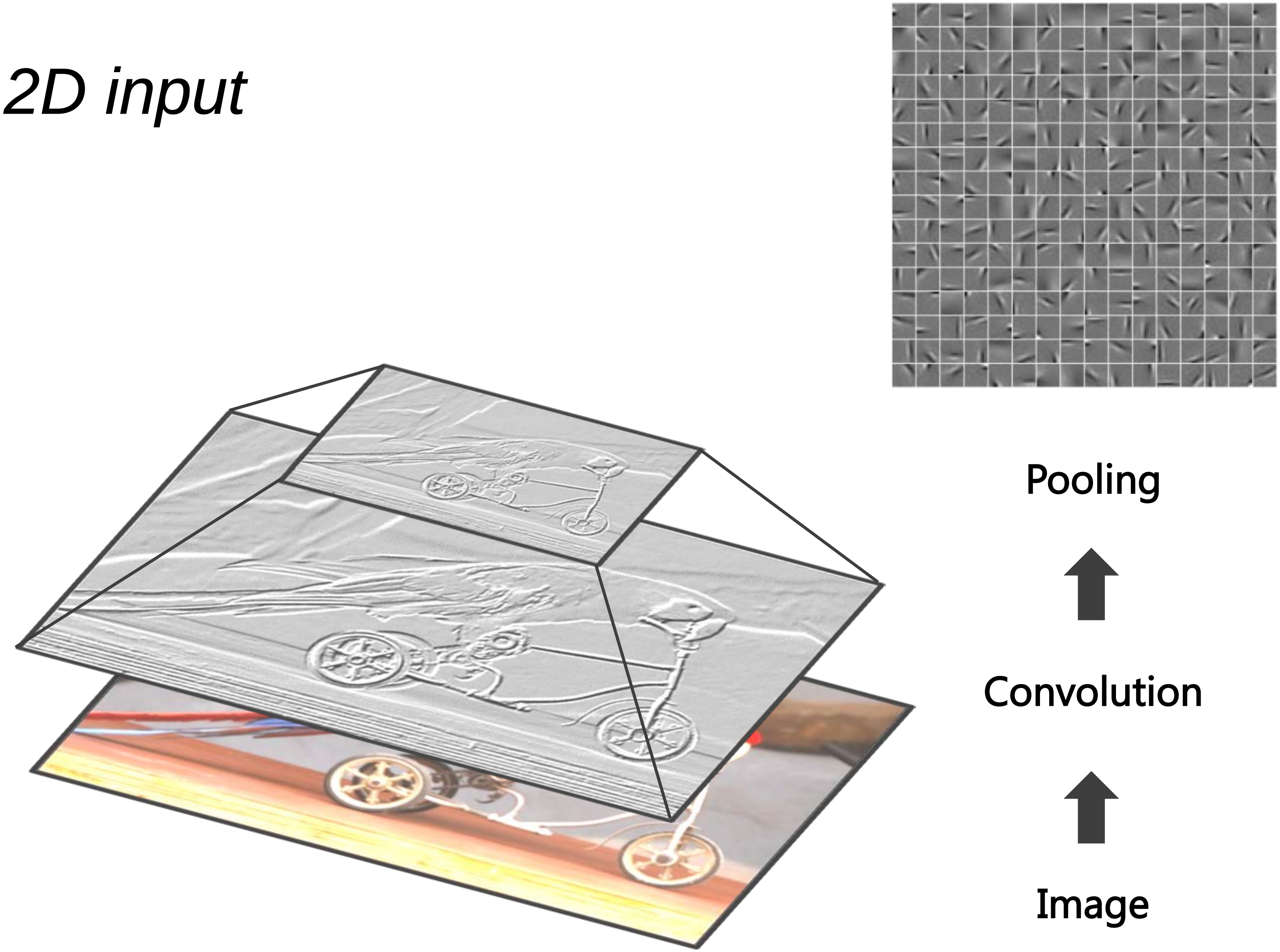
Convolutional NNs

- Pooling (max, average)



- **Pooling area: 2 units**
- **Pooling stride: 2 units**
- **Subsamples feature maps**

2D input

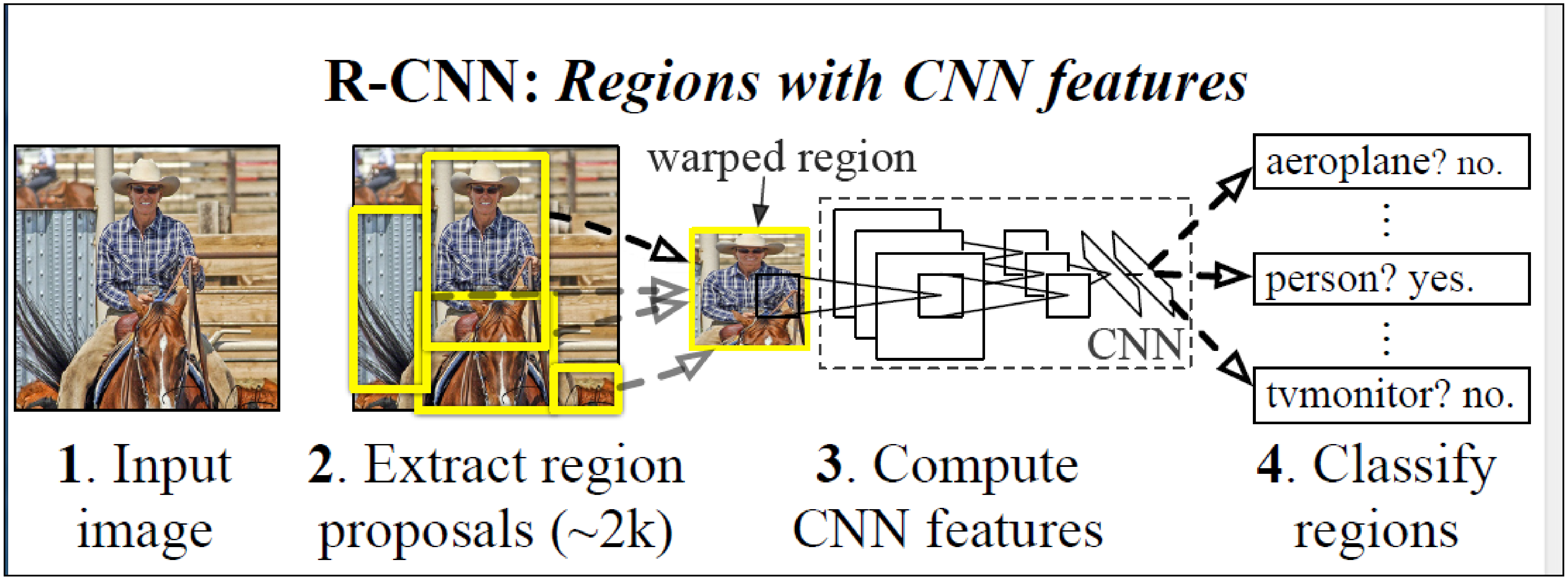


Core idea of “deep learning”

- Input: the “*raw*” signal (image, waveform, …)

- Features: hierarchy of features is *learned* from the raw input

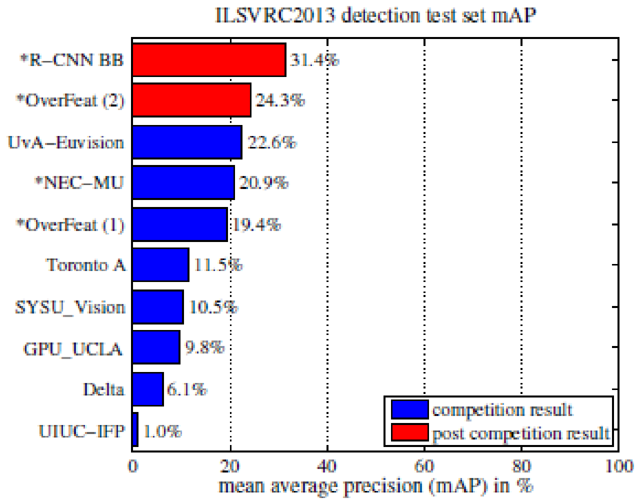
Ross's Own System: Region CNNs



Competitive Results

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [20]†	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [39]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [41]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [18]†	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	**71.8**	**65.8**	**53.0**	**36.8**	**35.9**	**59.7**	**60.0**	**69.9**	**27.9**	**50.6**	**41.4**	**70.0**	**62.0**	**69.0**	**58.1**	**29.5**	**59.4**	**39.3**	**61.2**	**52.4**	**53.7**

Table 1: Detection average precision (%) on VOC 2010 test. R-CNN is most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding-box regression (BB) is described in Section C. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. †DPM and SegDPM use context rescoring not used by the other methods.



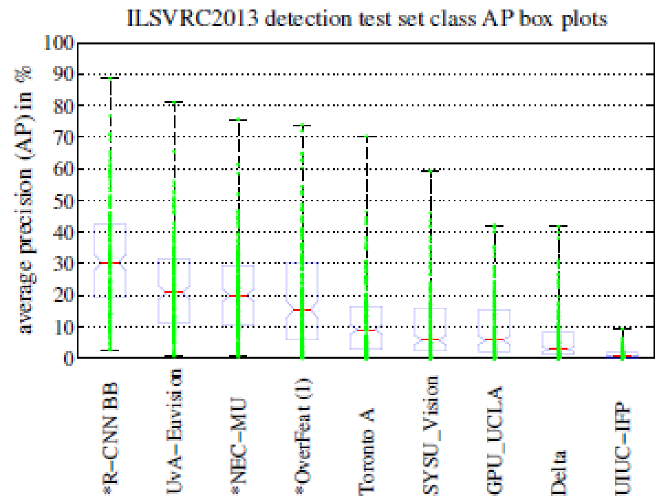


Figure 3: (Left) Mean average precision on the ILSVRC2013 detection test set. Methods preceeded by * use outside training data (images and labels from the ILSVRC classification dataset in all cases). **(Right) Box plots for the 200 average precision values per method.** A box plot for the post-competition OverFeat result is not shown because per-class APs are not yet available (per-class APs for R-CNN are in Table 8 and also included in the tech report source uploaded to arXiv.org; see `R-CNN-ILSVRC2013-APs.txt`). The red line marks the median AP, the box bottom and top are the 25th and 75th percentiles. The whiskers extend to the min and max AP of each method. Each AP is plotted as a green dot over the whiskers (best viewed digitally with zoom).

Top Regions for Six Object Classes

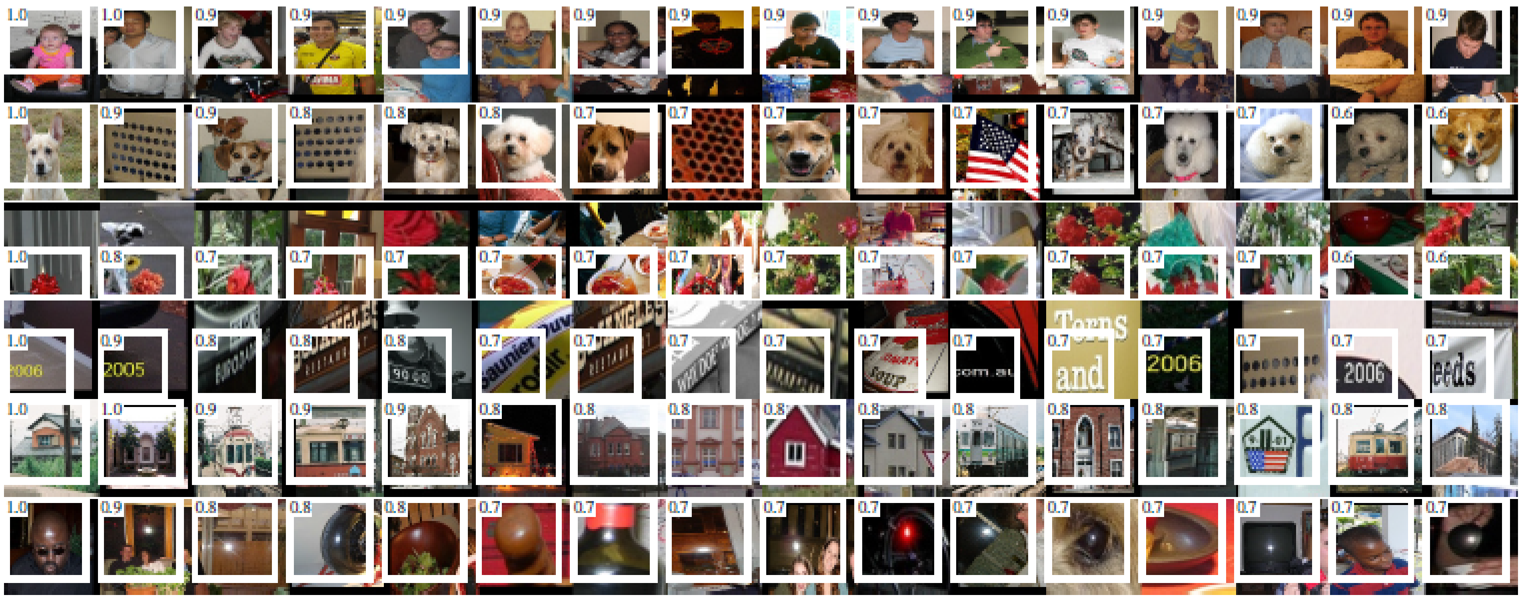


Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).