

A Sampling of Advanced Approaches to Vision

VISION 3

CMRoboBits 11/12/2007

Presenter: Stefan Zickler (szickler [at] cs.cmu.edu)

Announcements

◎ Projects!

- You all should have received an email from Manuela describing projects.
- If you haven't picked a project yet and sent us an email about it, please do so **TODAY, (right after class), so we can assign final groups tonight!**

◎ By this Wednesday (at **LAB-time**), please figure out how to split your project into 3 milestones, and think about details!

- November 28th, December 5th, December 12th
- We will discuss your ideas in detail on Wednesday, so please **be prepared!**

Project Ideas

- ⦿ Robot Soccer
- ⦿ Team Cleaning
- ⦿ Micromice
- ⦿ Play Recognition
- ⦿ Bar Pushing
- ⦿ Rescue Robots

Today's Lecture: Outline

- ⊙ Advanced Vision Bits & Pieces
 - Neural Nets
 - Contour Matching
 - Texture-Based (Bag of Words)
 - SIFT (& PCA-SIFT)
 - Active Appearance Models
- ⊙ Conclusions
 - Pros and Cons
 - What should *you* use?

So far, we have mainly looked at Color Segmentation

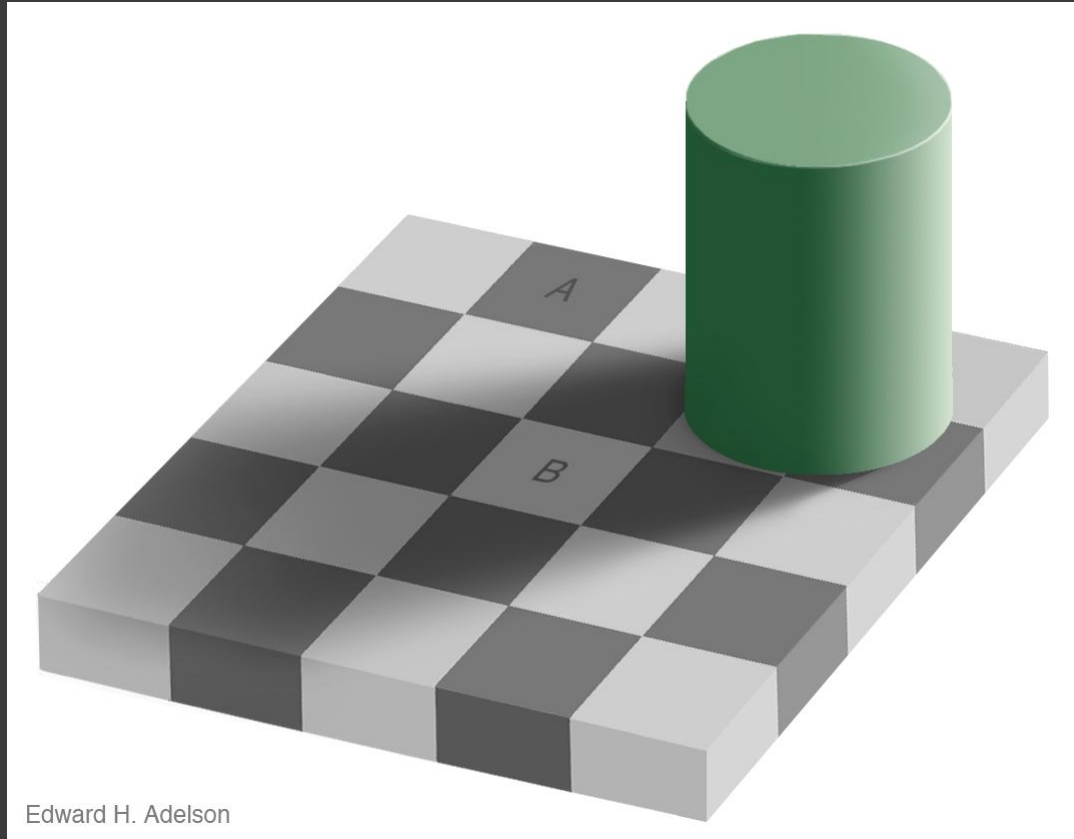
⦿ Why?

- Easy to implement
- Fast
- Works well in lab-environments

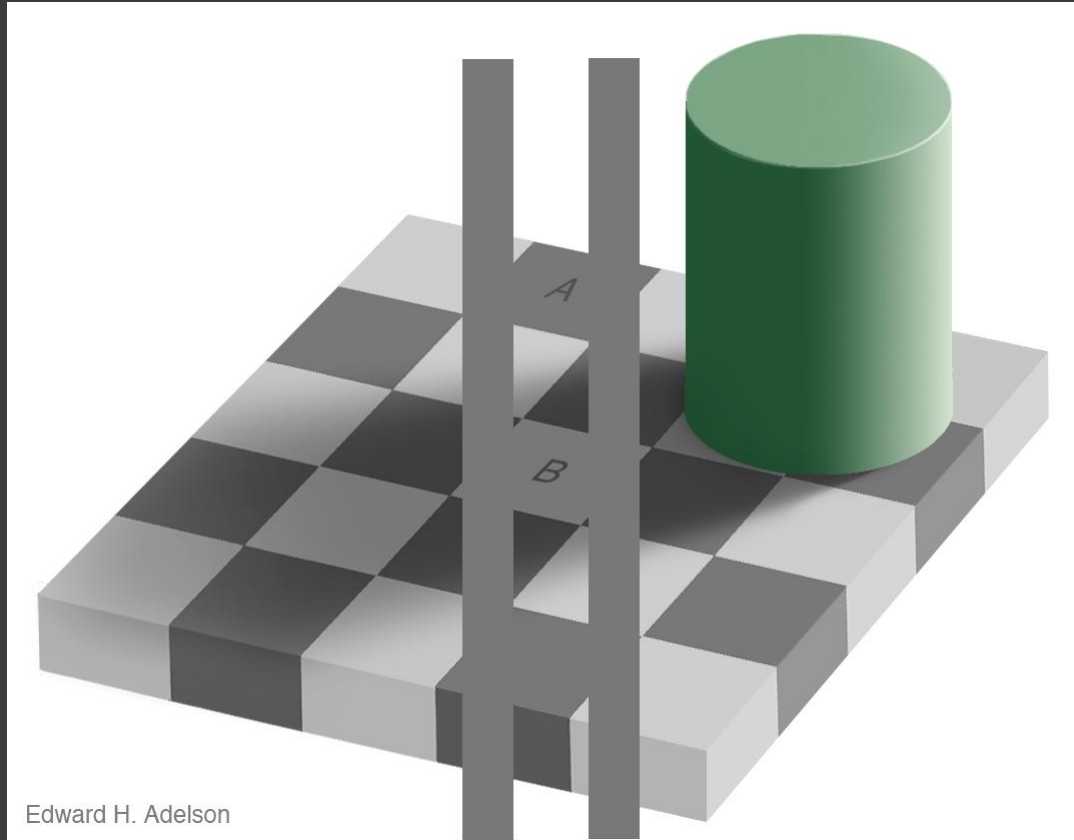
Color Segmentation contd.

- ⦿ What are the problems?
- ⦿ Not robust enough for general perception?

Measurement vs. Perception



Measurement vs. Perception



Proof!

Robust Perception

- Let's take a look at the most robust perceptual system known to man...

Summarizing Human Perception:

- All scientific evidence indicates that the human mind is nothing but a working brain.
- All scientific evidence indicates that the human brain is nothing but a bio-chemical, parallel computer.
- We can emulate computation, thus we should be able to emulate the brain.

So why haven't we...

- ⦿ ...emulated an entire brain yet?
- ⦿ 1. We don't fully understand it (yet).
 - We understand the low-level (neurons, chemicals, etc...)
 - We understand the high-level (regions of brain-activity related to certain tasks)
 - But we are still trying to understand what's in between...poking Monkey-neurons and looking at MRIs.
 - Where/how do patterns of neural activation form “thinking”, “consciousness”, “perception”?
 - It's a bit like reverse-engineering a processor from scratch
- ⦿ 2. We know enough to know that it's near impossible to *precisely* emulate on any of today's computational hardware.
 - Brain uses complex chemical reactions on atomic level. Some researchers even suggest that quantum-effects might play a significant role.
 - This is insanely difficult to emulate with even just a couple neurons. Try billions...

So what do we know?

Fun Facts about the brain.

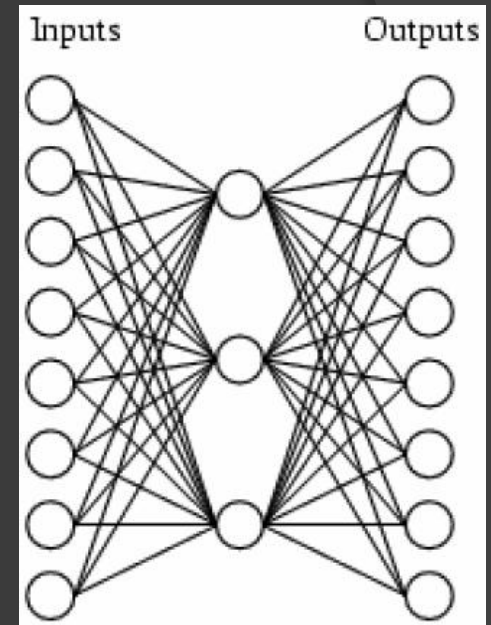
- Neuron Switching Time: .001 seconds
- Number of Neurons: 10^{10}
- Connections per Neuron: 10^{4-5}
- Scene recognition time: .1 seconds
- 100 inference steps doesn't seem like enough
- -> much parallel processing going on.

Imitation is the sincerest form of flattery: ANNs (Artificial Neural Networks)

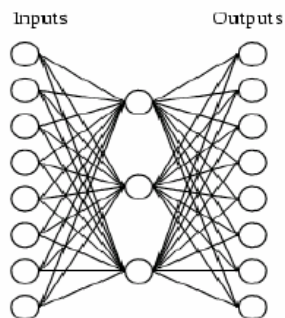
- ⦿ An approximation to what human neurons do.
- ⦿ Similarities:
 - Many switching units
 - Many weighted interconnections
 - Highly parallel, distributed (but normally computed by using a serial algorithm).

ANNs

- Normally, ANNs have an input layer, one or more hidden layers, and an output layer.
- Connections have weights.
- Neurons normally sum the input and have an activation threshold before they “fire”.
- ANNs can be trained e.g. using backpropagation.
- Lots of complex theory does exist.



The mystery of the hidden layers



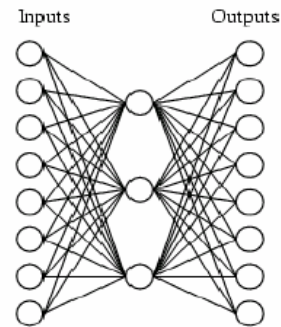
A target function:

Input	Output
10000000	→ 10000000
01000000	→ 01000000
00100000	→ 00100000
00010000	→ 00010000
00001000	→ 00001000
00000100	→ 00000100
00000010	→ 00000010
00000001	→ 00000001

Can this be learned??

The mystery of the hidden layers (contd.)

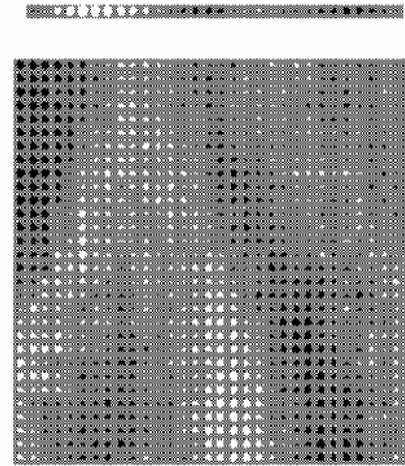
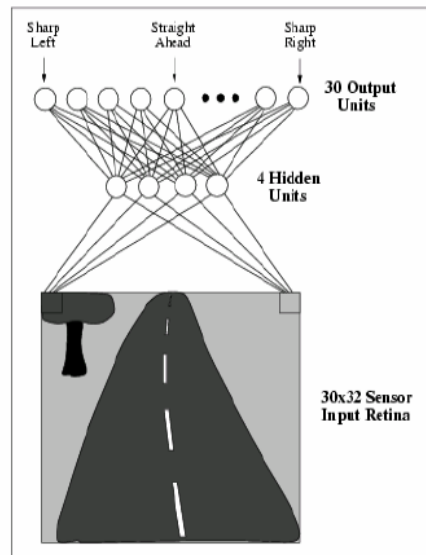
A network:



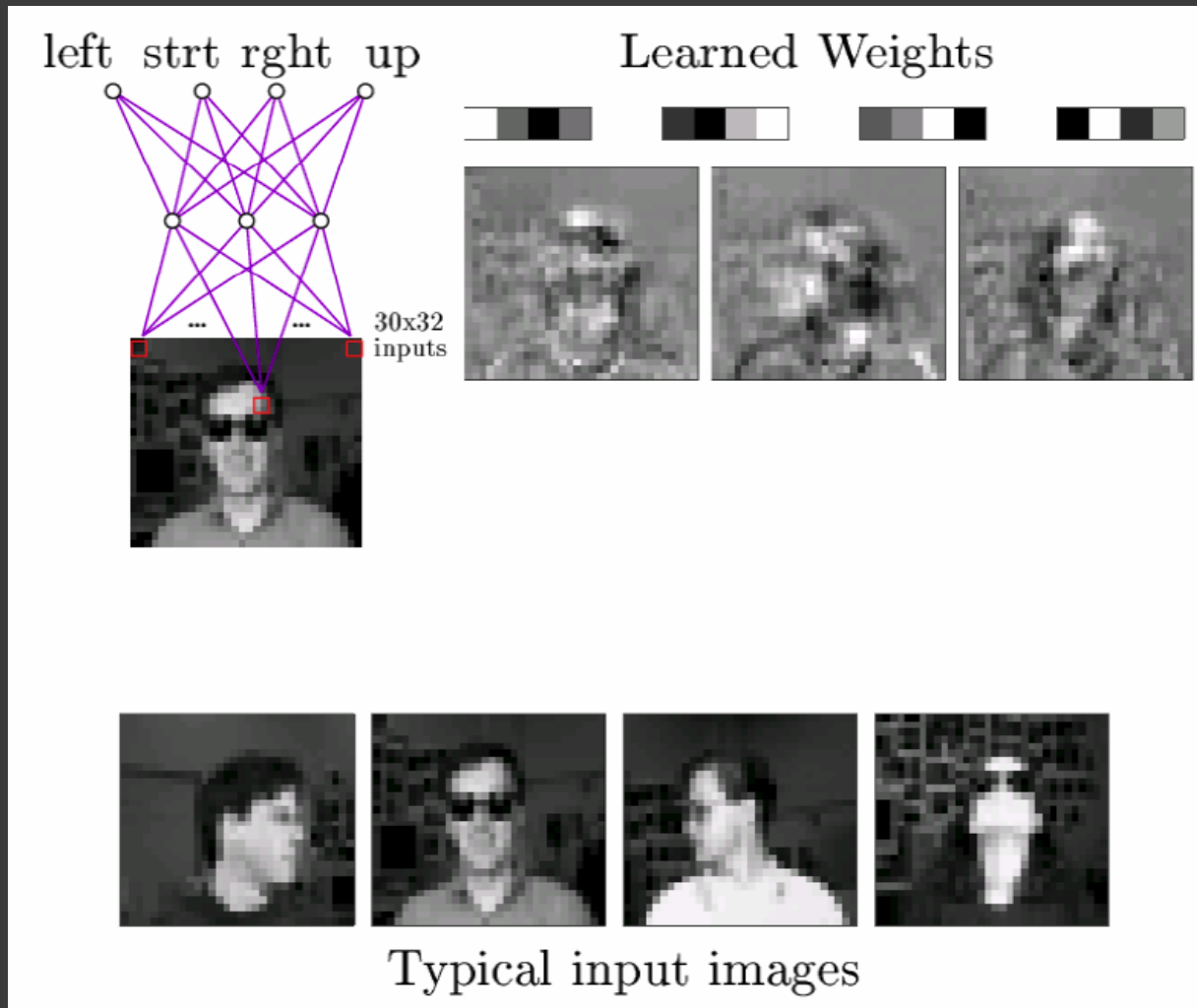
Learned hidden layer representation:

Input		Hidden Values		Output
10000000	→	.89 .04 .08	→	10000000
01000000	→	.01 .11 .88	→	01000000
00100000	→	.01 .97 .27	→	00100000
00010000	→	.99 .97 .71	→	00010000
00001000	→	.03 .05 .02	→	00001000
00000100	→	.22 .99 .99	→	00000100
00000010	→	.80 .01 .98	→	00000010
00000001	→	.60 .94 .01	→	00000001

Neural Nets for computer vision



Neural Nets for face recognition



ANNs: summary

⦿ Strengths:

- Can deal with input noise
- We don't need to know how to solve the problem

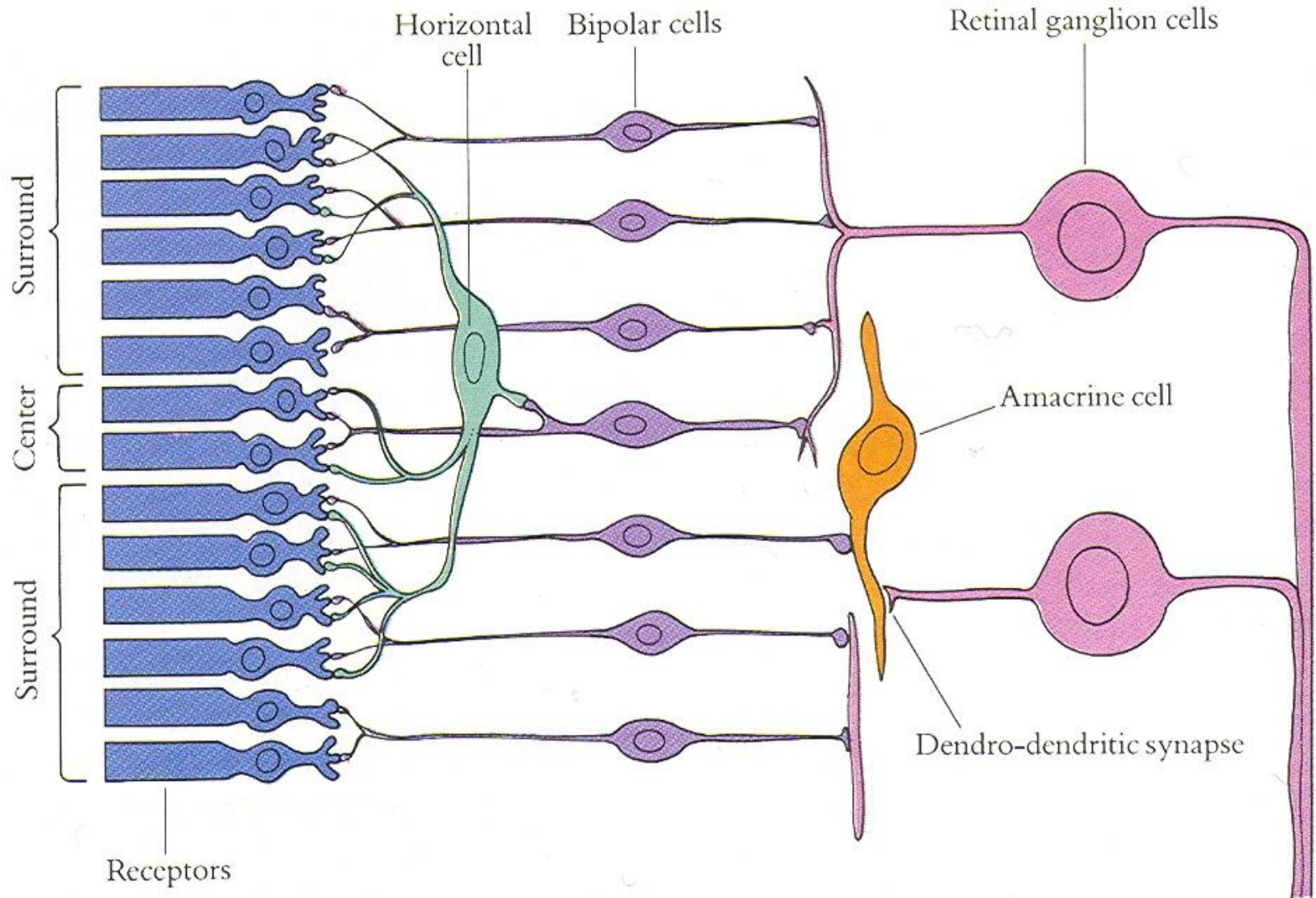
⦿ Weaknesses:

- We need to find a good layout for encoding the problem / tweak network structure
- Overfitting
- Slow to learn
- Solution is unintelligible
- Not very fast to execute (remember, we are *simulating* parallel computation)
 - This might improve as massive computational parallelism becomes a commodity.

A second look at human perception

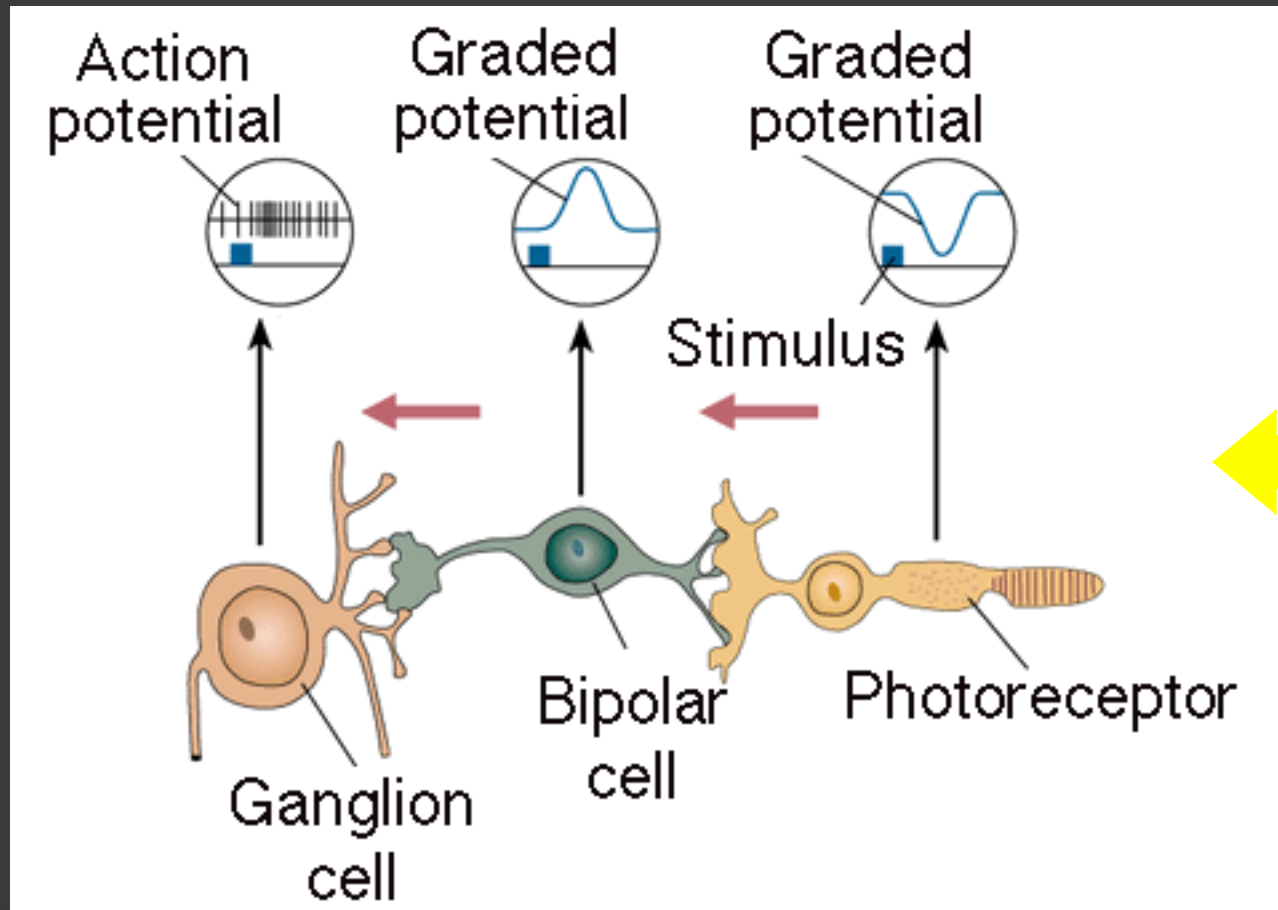
- ⦿ Maybe it is easier not to emulate the brain...
- ⦿ ...but rather emulate its functionality
- ⦿ Let's focus on what our human *perceptual* system does!

Retinal Receptive Fields



Retinal Receptive Fields

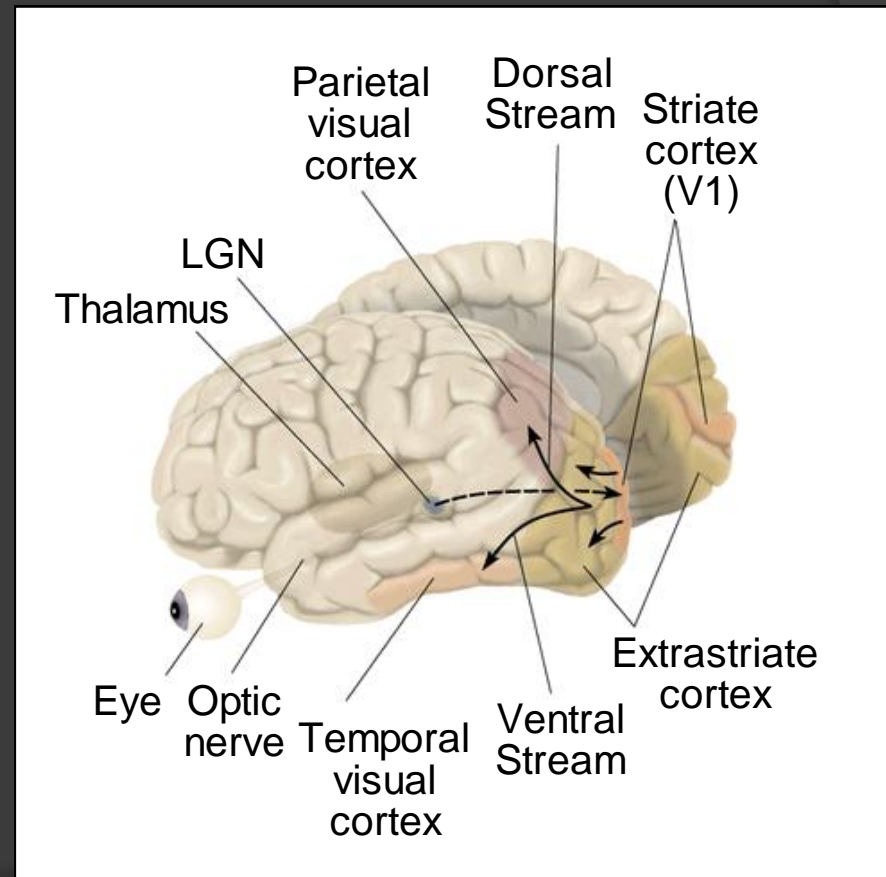
Receptive field structure in bipolar cells



Visual Cortex

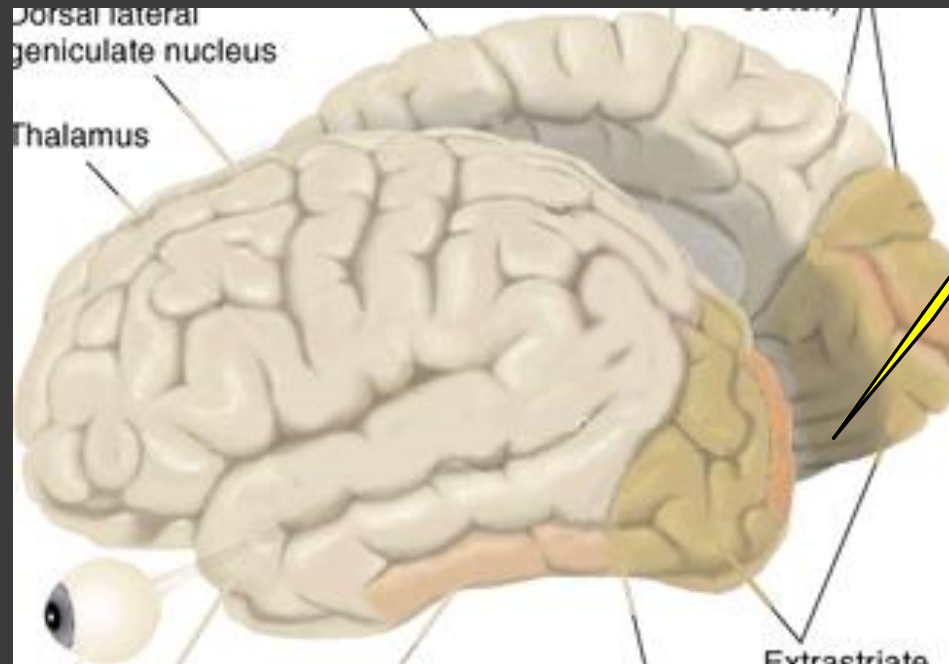
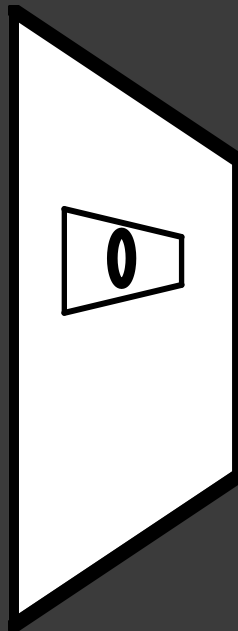
Cortical Area V1

aka:
Primary visual cortex
Striate cortex
Brodmann's area 17



Cortical Receptive Fields

Single-cell recording from visual cortex



Cortical Receptive Fields

Three classes of cells in V1

Simple cells

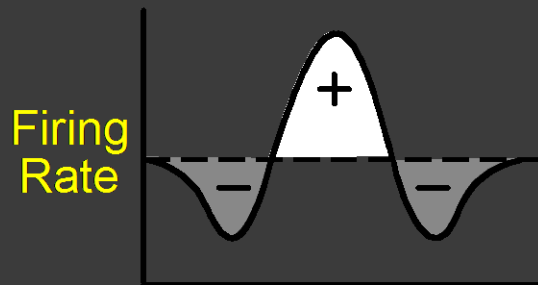
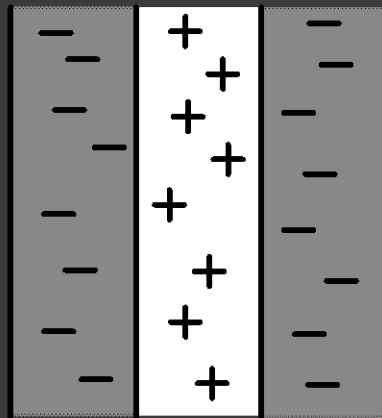
Complex cells

Hypercomplex cells

Cortical Receptive Fields

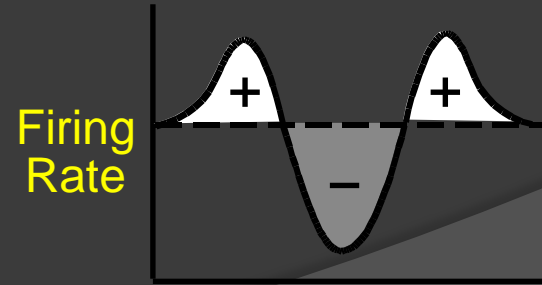
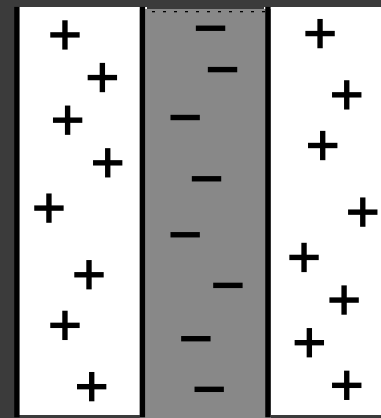
Simple Cells: “Line Detectors”

A. Light Line Detector



Horizontal Position

B. Dark Line Detector

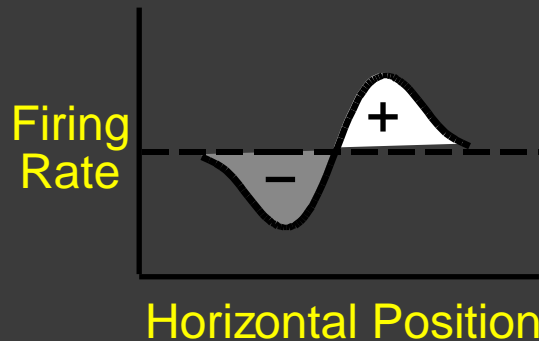
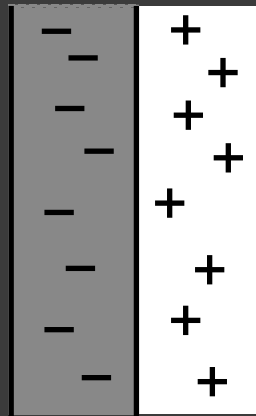


Horizontal Position

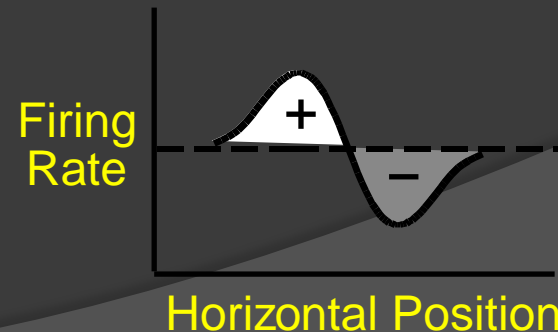
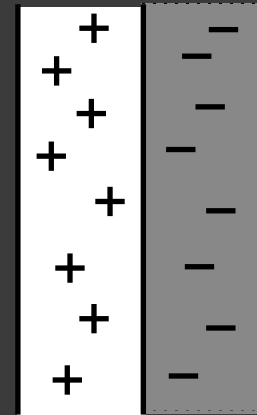
Cortical Receptive Fields

Simple Cells: "Edge Detectors"

C. Dark-to-light Edge Detector

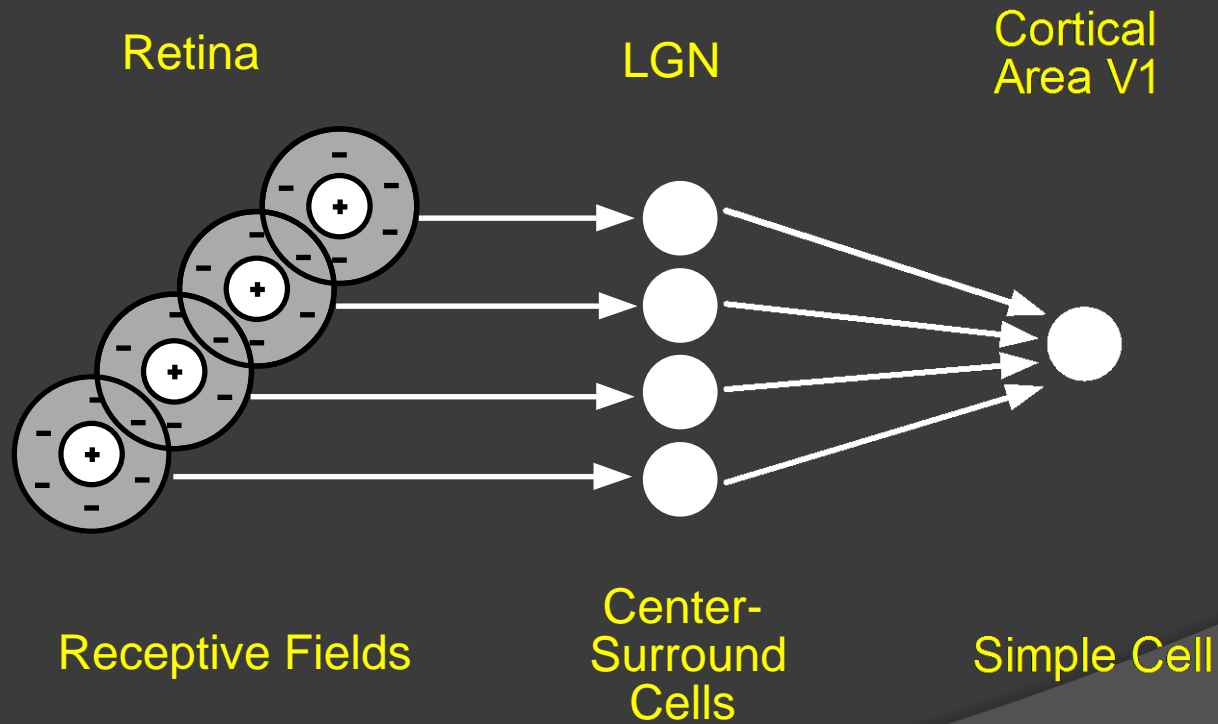


D. Light-to-dark Edge Detector



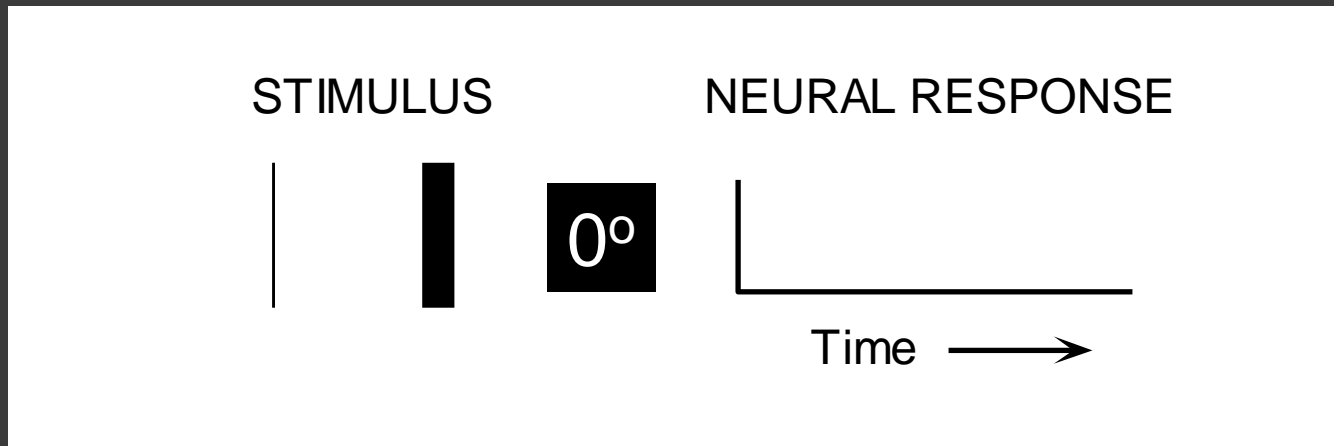
Cortical Receptive Fields

Constructing a line detector



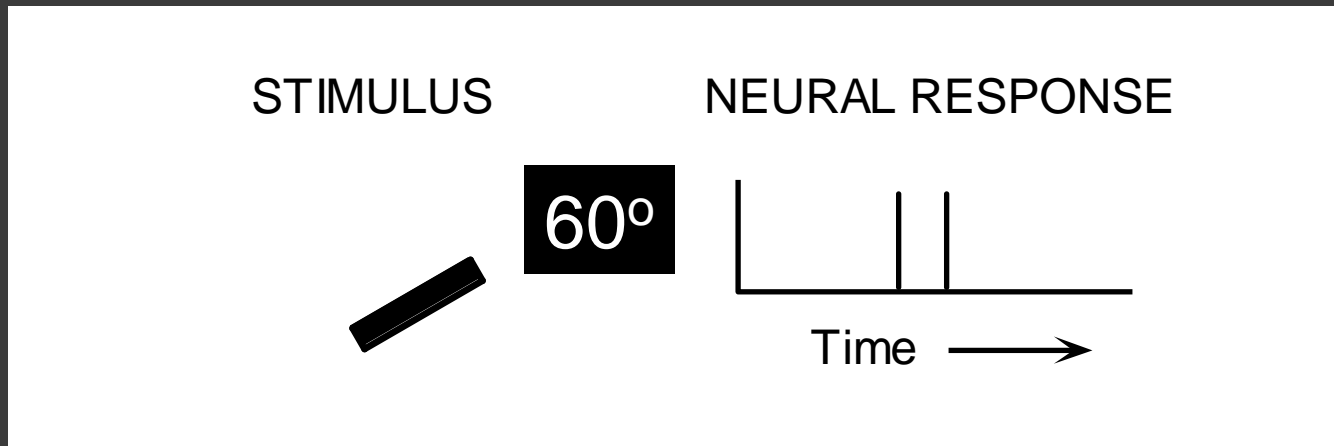
Cortical Receptive Fields

Complex Cells



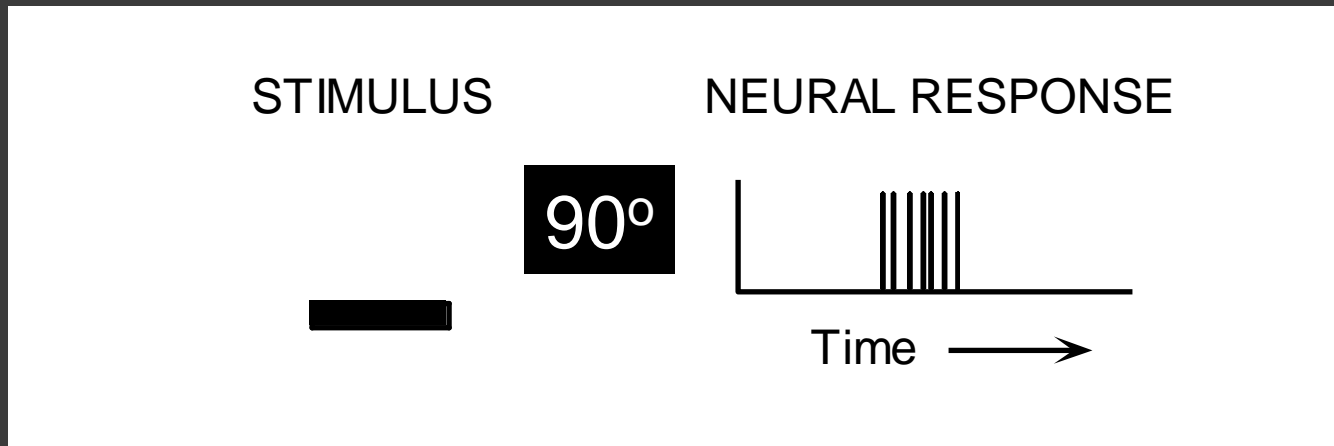
Cortical Receptive Fields

Complex Cells



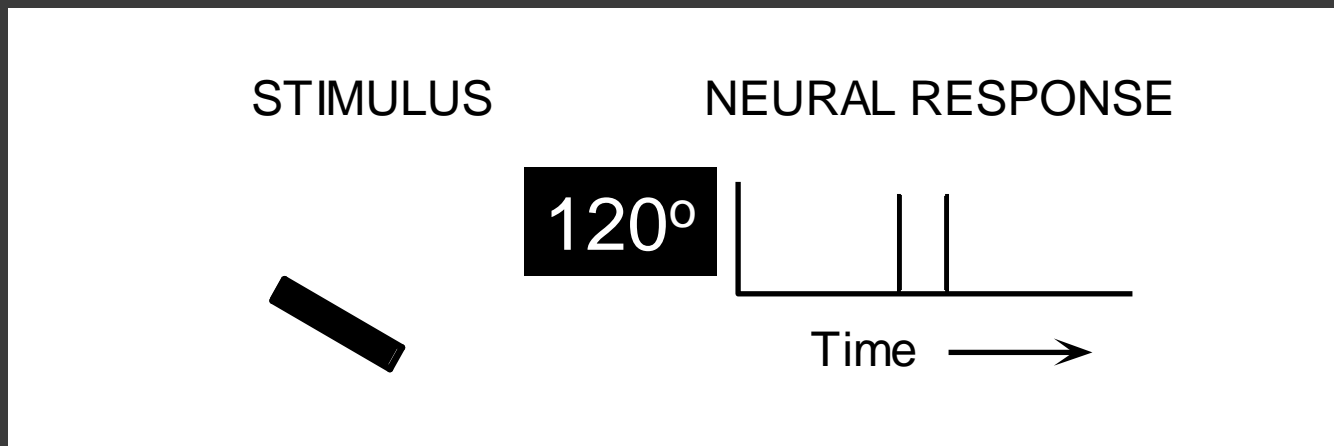
Cortical Receptive Fields

Complex Cells



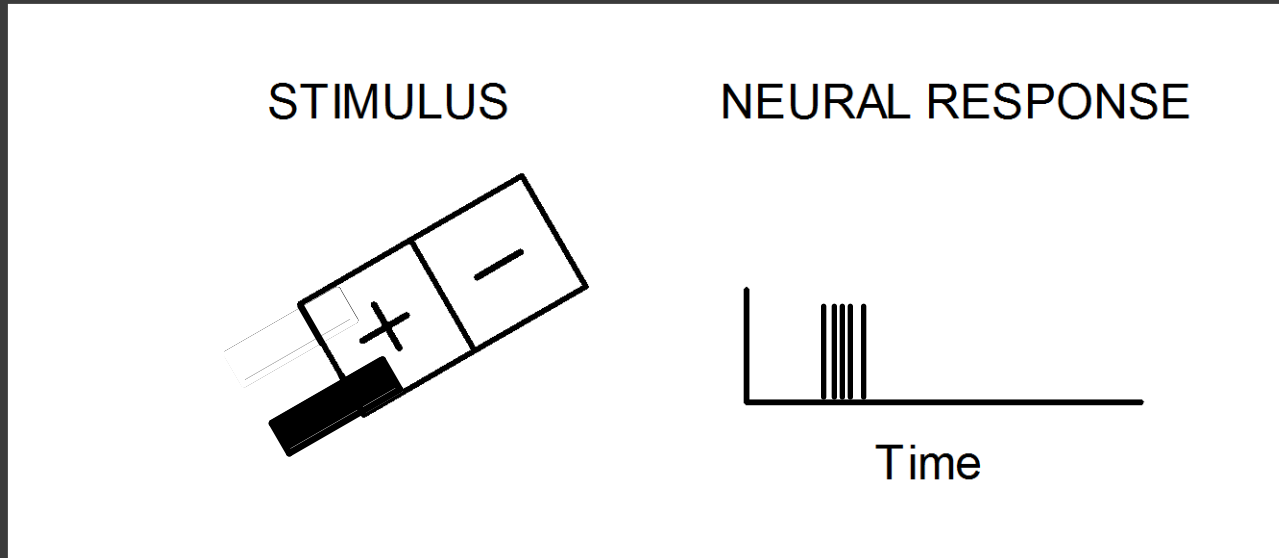
Cortical Receptive Fields

Complex Cells



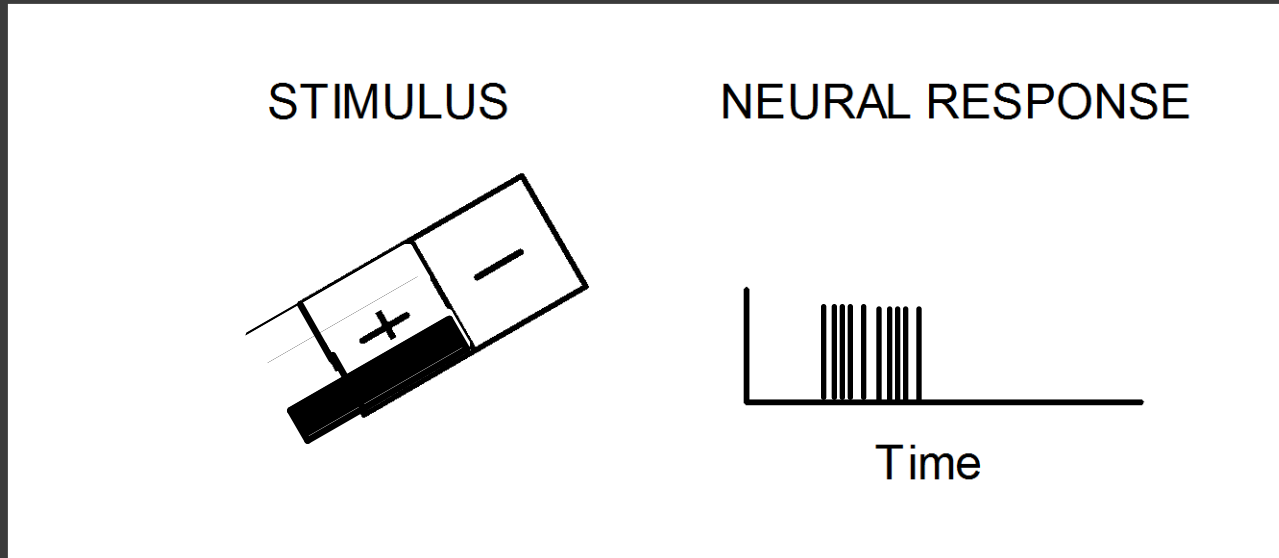
Cortical Receptive Fields

Hypercomplex Cells



Cortical Receptive Fields

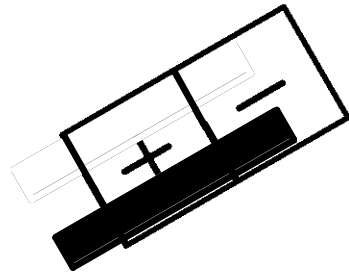
Hypercomplex Cells



Cortical Receptive Fields

Hypercomplex Cells

STIMULUS

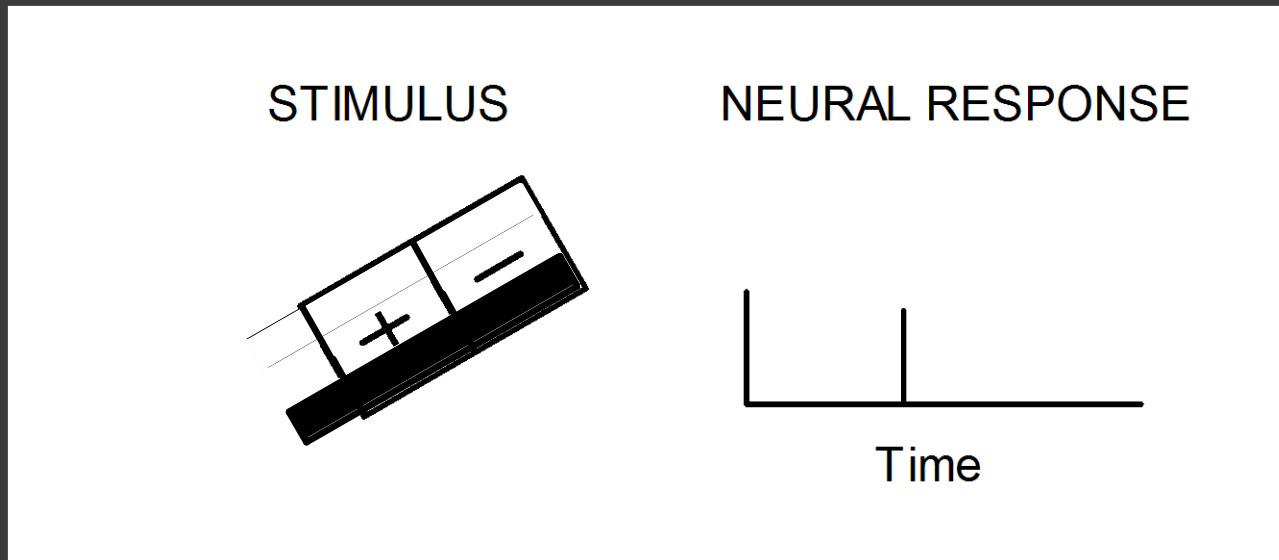


NEURAL RESPONSE



Cortical Receptive Fields

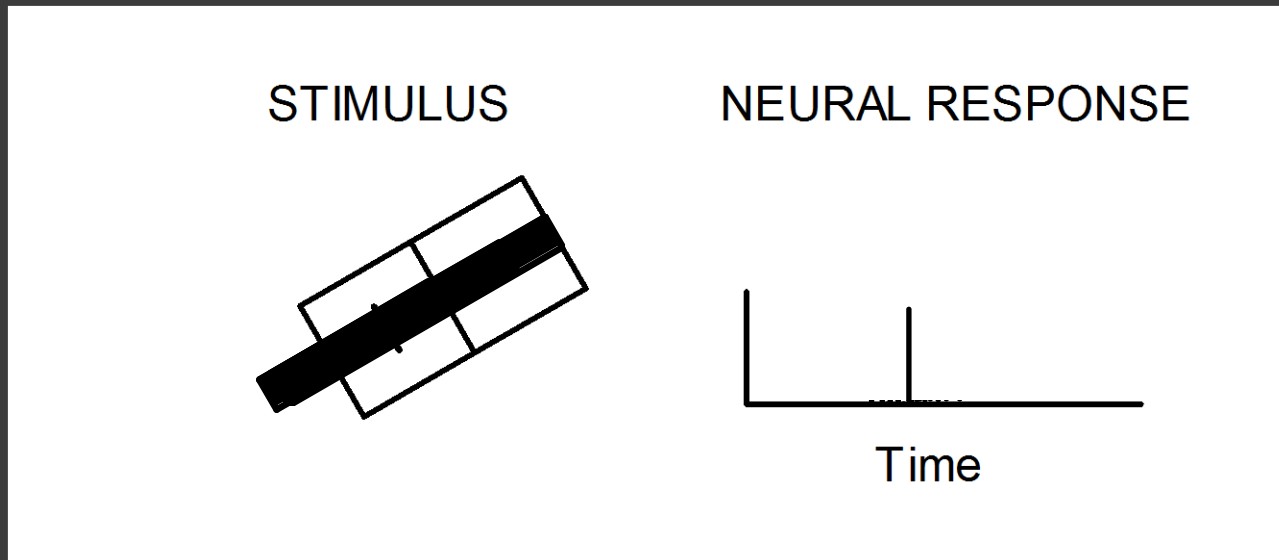
Hypercomplex Cells



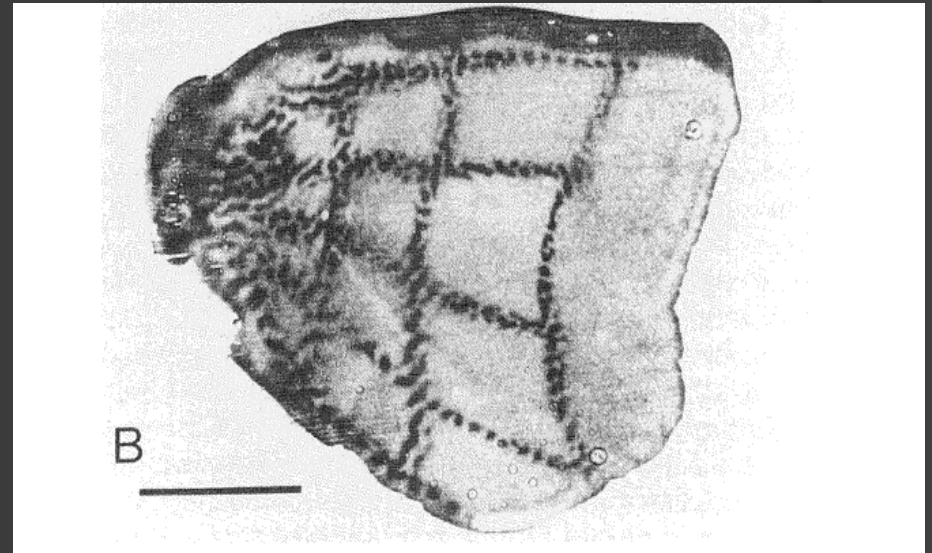
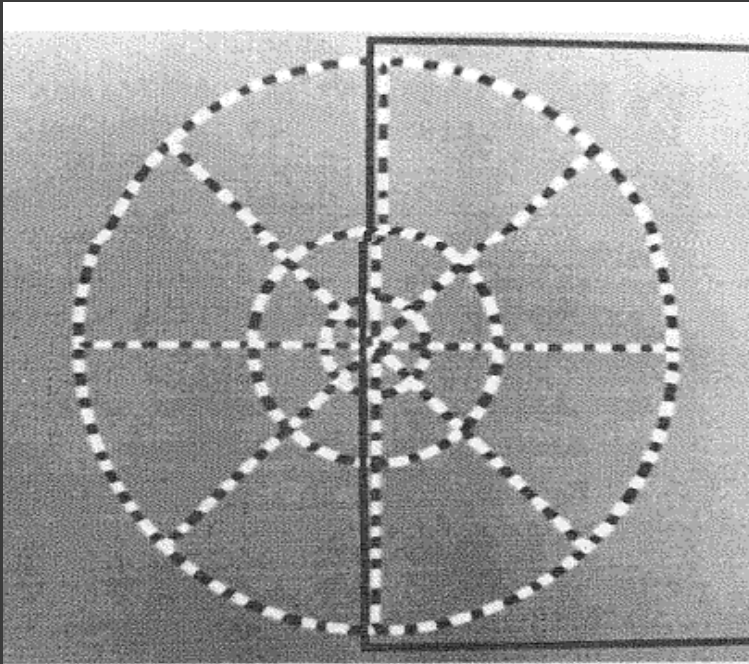
“End-stopped” Cells

Cortical Receptive Fields

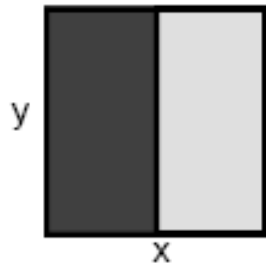
“End-stopped” Simple Cells



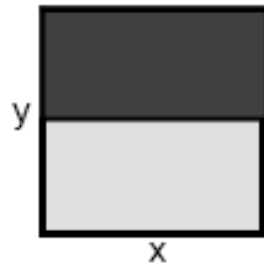
Mapping from Retina to V1



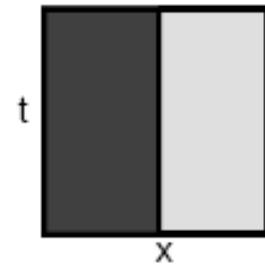
Why edges?



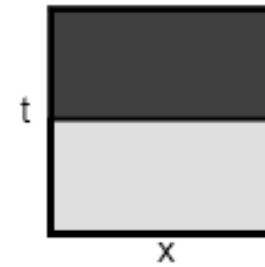
(a)



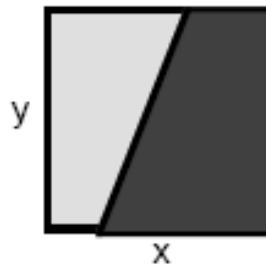
(b)



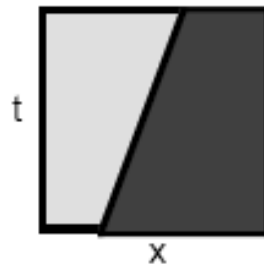
(c)



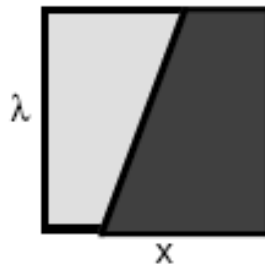
(d)



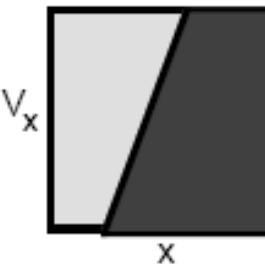
(e)



(f)

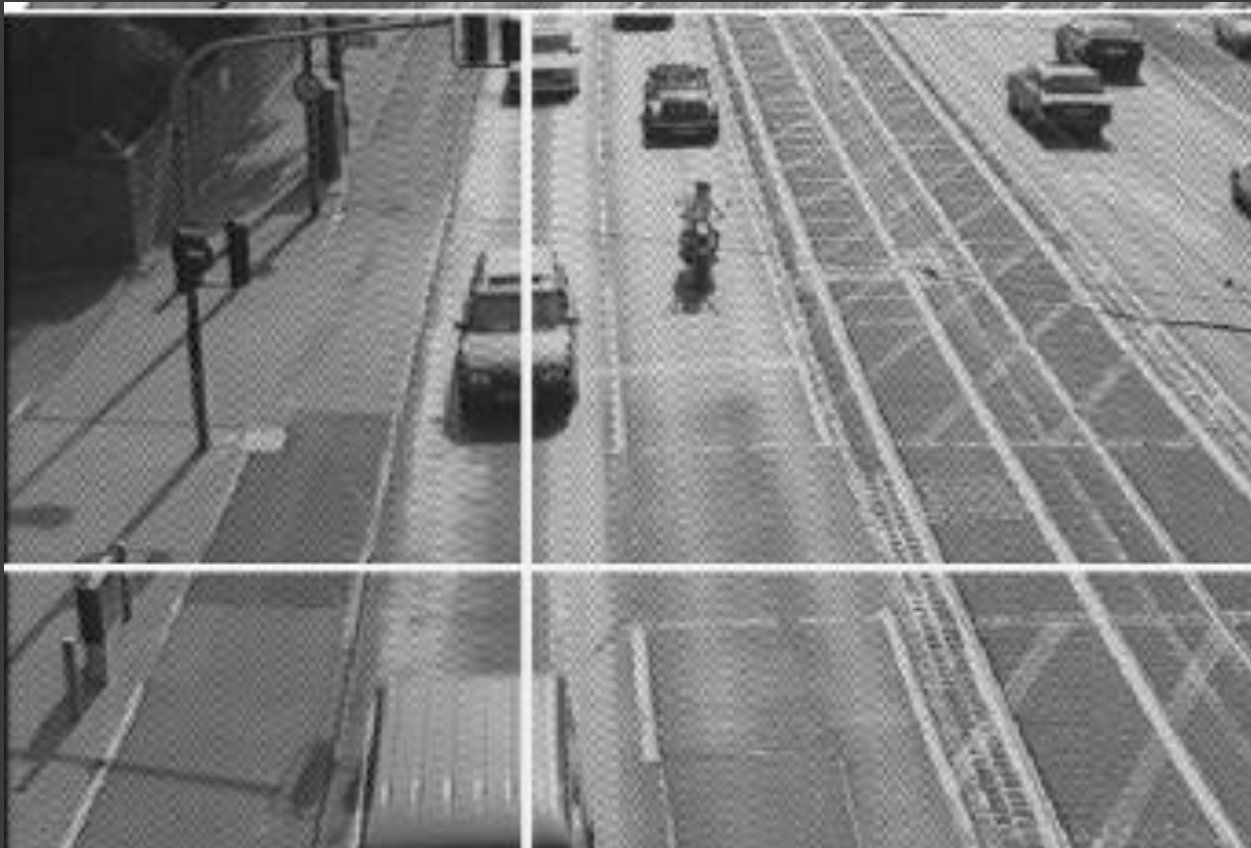


(g)



(h)

Because our world is structured!



Edge Detection

- Remember earlier vision lecture...
- We can use convolution to apply a matrix kernel to an image
- It's fairly simple to perform edge detection this way

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

$$O(i, j) = \sum_{k=1}^m \sum_{l=1}^n I(i+k-1, j+l-1)K(k, l)$$

Examples:

- Sobel:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

- Prewitt:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * \mathbf{A}$$

- If required, you could add diagonal filters for better results

Example: Sobel Detector



Canny Edge Detector

- ◎ Trying to avoid false positives
 - Adds a Gaussian smoothing step before using Sobel-Style Edge detection
 - Adds hysteresis step instead of static threshold to decide what's an edge

Edge Detection is Not Vision

- ⦿ We now have an edge-representation of the image
- ⦿ What's Next?
 - Example: Pedestrian Recognition

The following slides discuss approaches from:

Real-Time Object Detection for “Smart” Vehicles (Darius Gavrilla & Vasanth Philomin)

&

Automatic Target Recognition by Matching Oriented Edge Pixels (Clark Olson & Dan Huttenlocher)

Edge-Based Target Recognition

- ⦿ What are we trying to achieve?
 - We want to determine the presence and location of a template T in an image I .



Edge-Template

(hand-drawn from footage, or automatically generated from CAD models)

?



Image Scene

Real world, real time video footage.

Basic Idea

- Our template T is an edge-map.
- Create edge map of image. This is our feature-image I .
- Slide T over I , until it somehow delivers the best match.



Feature
Template T



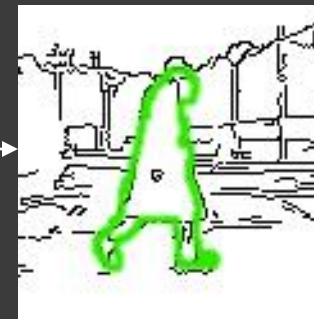
Raw Image



Feature Image I



Search for best
match of T in I



Found match of
 T in I

Naïve Approach: Binary Matching

- ⦿ We determine a match by counting the pixels that match between the template and the edge-image. If this count is high enough (if it is close to the count of pixels in the template) then we have a match.
- ⦿ This approach only works well if the template really has the exact size, shape and orientation as the image.
- ⦿ It does not give us any information about how far the non-matching pixels are off.

Chamfer Distance

- ⦿ Let T be our template.
- ⦿ Let I be the image's edge-map.
- ⦿ The Chamfer distance is the average distance to the nearest feature.

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

Hausdorff Measure

$$h(M, I) = \max_{m \in M} \min_{i \in I} \|m - i\|$$

- Let M be the set of object model pixels.
Let I be the set of image edge pixels.

$h(M, I)$ is the distance of the worst matching object pixel to its closest image pixel.

Problem: The Hausdorff measure makes the assumption that each object pixel occurs in the image. This is obviously not true when an object is occluded.

Partial Hausdorff Measure

$$h_K(M, I) = K^{th}_{m \in M} \min_{i \in I} \|m - i\|$$

- ⦿ K object pixels that are closest to the image.
- ⦿ K can be tweaked to the minimum number of pixels that we expect to find in an image.
- ⦿ K can also be set higher to reduce the rate of false positives, but we might miss some matches that way.

A shortcut to the partial Hausdorff Measure

- ⦿ Normally, we want to know whether the partial hausdorff measure of the K closest object pixels is below a certain threshold δ .
 - $h_K(M,I) < \delta$
- ⦿ An alternative method to achieve this, is by dilating our image edge-map by a disk of radius δ . We then simply count the amount of template pixels that match our dilated map. If this count surpasses K then we have a match (this is equivalent to $h_K(M,I) < \delta$)

Distance Metrics Compared

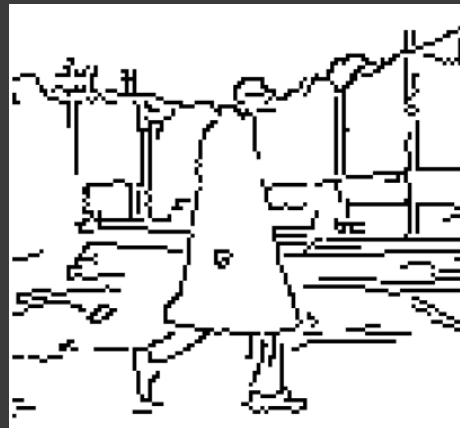
- ◉ Chamfer:
 - Average Distance between template and image
 - Doesn't handle occlusion too well.
- ◉ Hausdorff:
 - Maximum Distance between template and image
 - Doesn't handle occlusion at all.
- ◉ Partial Hausdorff
 - Distance of Kth closest match
 - We can treat occlusion by tweaking K.

The search for matches

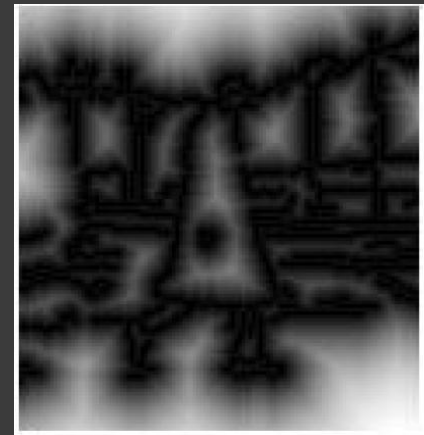
- ⦿ Computing Chamfer or Hausdorff measures is computationally expensive.
- ⦿ Mainly because we need to perform a search for the closest image pixel of each template pixel.
- ⦿ Sliding a template over an image in a pixel-by-pixel fashion is inefficient.
- ⦿ We need better search methods!

Distance Transform

- ⦿ For each image:
 - We first compute the image's edge map
 - We then compute the Distance Transform (DT) which is an intensity map that marks the distance to the closest pixel on the edge map.



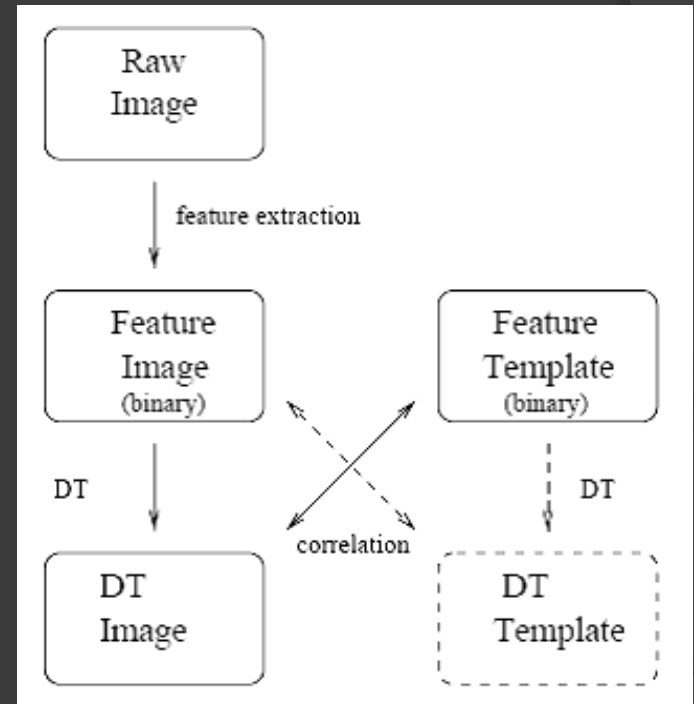
Edge Map



DT

Why Distance Transform?

- Provides us with inherent distance information that can be used by our template matching algorithm.
- It acts as our lookup-table for finding the distance of the closest matching object pixel that we previously needed to search for manually.



Chamfer in DT-space

- Remember Chamfer:

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

$d_I(t)$ can now be computed by a single lookup in the DT image!

This DT image stays the same while we slide our template over it => We only need to compute it once!

Using DT and the partial Hausdorff shortcut to achieve efficient matching.

- ⦿ We create the Distance Transform DT of our image I .
- ⦿ We now dilate the distance transform of I by δ .
- ⦿ We match our template against this. We find the K best matching pixels from our dilated edge-map.
- ⦿ If the K^{th} match has a distance value $d = 0$ then we have a match (equivalent to $h_K(M, I) < \delta$)
- ⦿ If the K^{th} match has a distance value $d > 0$ then d is the distance to the closest possible position where our template could actually match.
- ⦿ Thus, we can rule out any template positions that are less than d . This can reduce our search space dramatically!

Oriented Edges

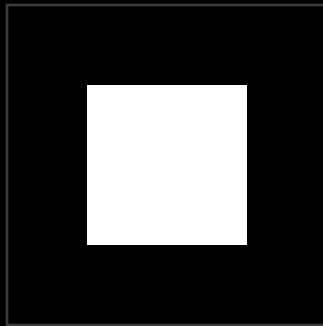
⦿ Problem:

- Plain edges only indicate the presence of a contrast boundary at a given location, but carry no additional information about the gradient itself.
- This is often not informative enough and might lead to false positives in target recognition.

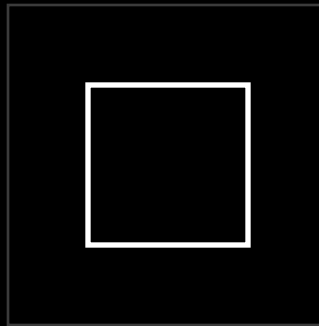
⦿ Improvement:

- Orientation of gradients add a more distinctive dimension

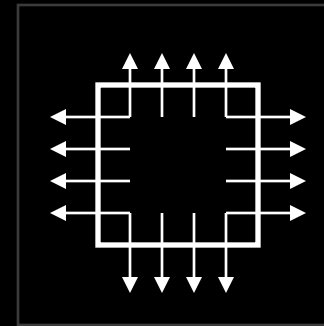
Oriented Edges



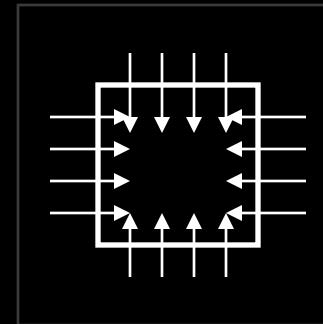
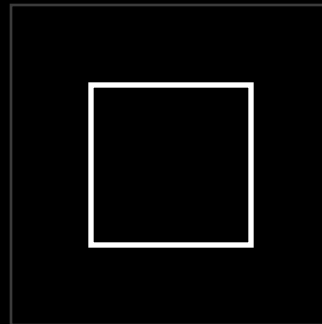
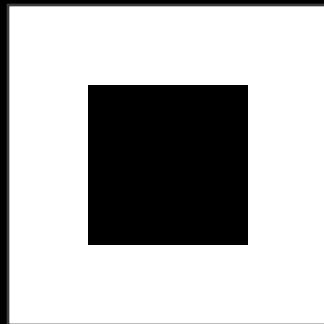
Original Image



Edge
Detection
Result



Oriented
Edges based
on gradients



Oriented Edges

- Remember Sobel:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

- Computing the gradient direction is easy:

$$\Theta = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Hausdorff for Oriented Edges

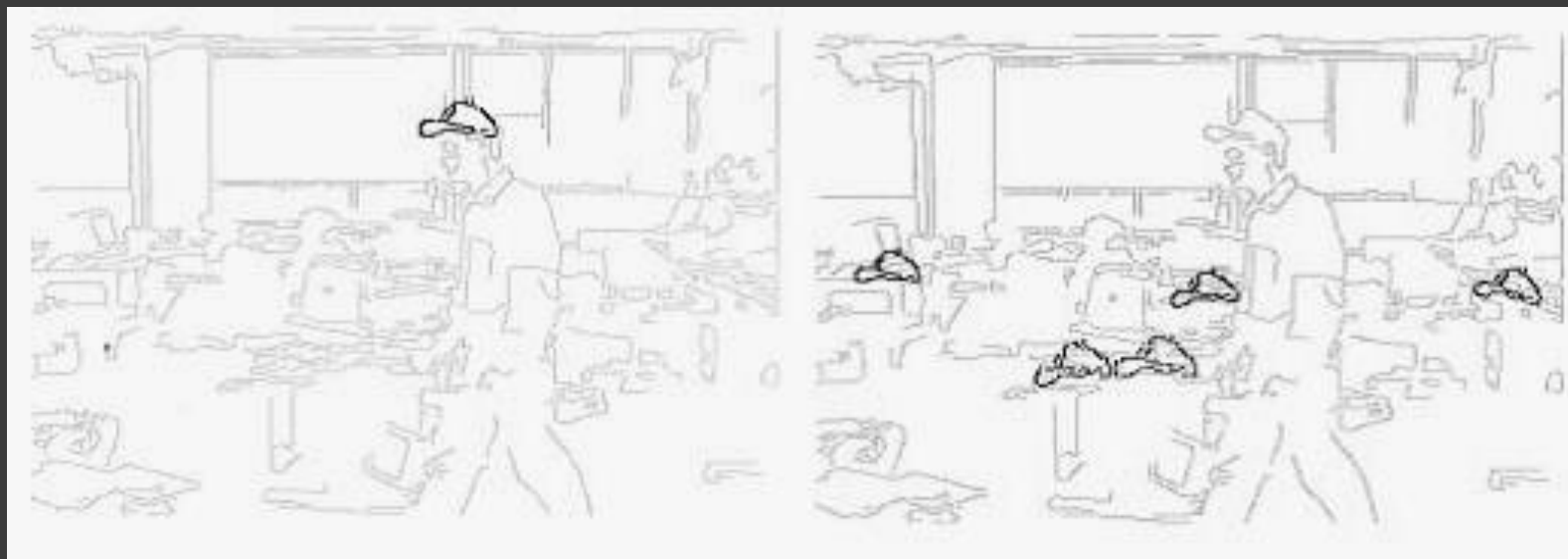
$$h_{\alpha}(M, I) = \max_{m \in M} \min_{i \in I} \max \left\{ \left\| \begin{bmatrix} m_x - i_x \\ m_y - i_y \end{bmatrix} \right\|, \frac{|m_o - i_o|}{\alpha} \right\}.$$

Distance between
edge pixels

Orientation difference

Our optimizations using a distance transform representation still works just the same, except that our DT image is now a 3-dimensional image.

Oriented Edges



Matching Multiple Templates

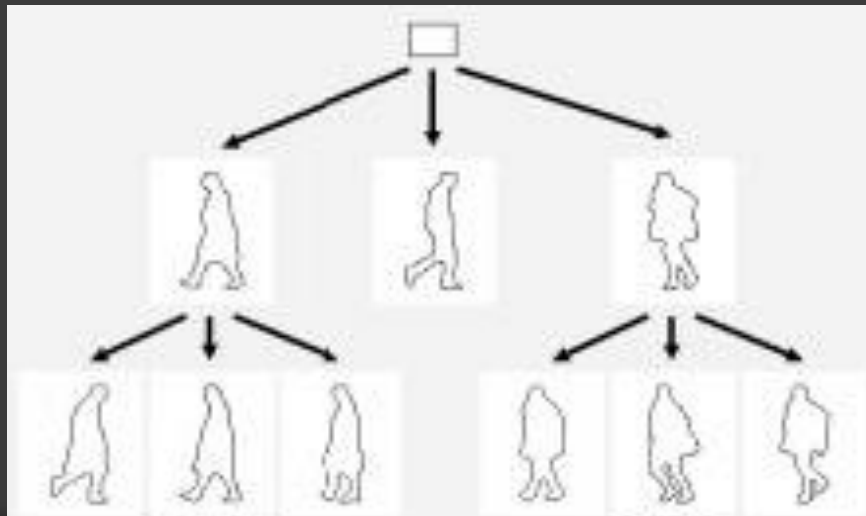
- ⦿ So far we have looked at matching single edge templates to an image.
- ⦿ In the real world however, objects tend to appear in many different shapes
 - Our viewpoint can change
 - The object might actively change its shape (such as walking pedestrians).

Matching Multiple Templates (contd.)

- We need to create many templates (possibly one for each expected combination of viewpoint and shape).
- That's a lot of templates to match, especially for real-time purposes!
- There must be a better way than comparing each template separately (especially because they all look kind of similar).

Coarse to Fine Hierarchical Organization

- Our tree is ordered by generality, the most general template is the root of our tree.
 - The most general template is the one which has the lowest maximum distance measure to all other templates.
- The leafs of our tree are all possible templates.

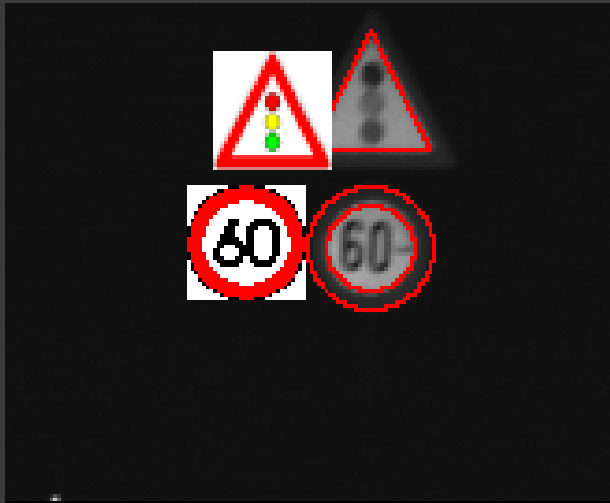


Coarse To Fine Hierarchical Search

- ⦿ We start at the root template and try to find a match in our image. We choose the distance threshold to be large enough so that our match could potentially contain any of our child-nodes.
- ⦿ If a match is found, we descend down the tree, and try to match the next level of templates (by focusing only on the area in the image that has been matched by our parent). We now use a smaller distance threshold that is still large enough to possibly contain each of our child-templates.
- ⦿ We repeat this process (usually using depth-first search) until one of our leafs matches.
- ⦿ How much speed do we gain? Gavrilin and Philomin say, “Up to three orders of magnitude”, but depends on various factors.

Let's look at the application

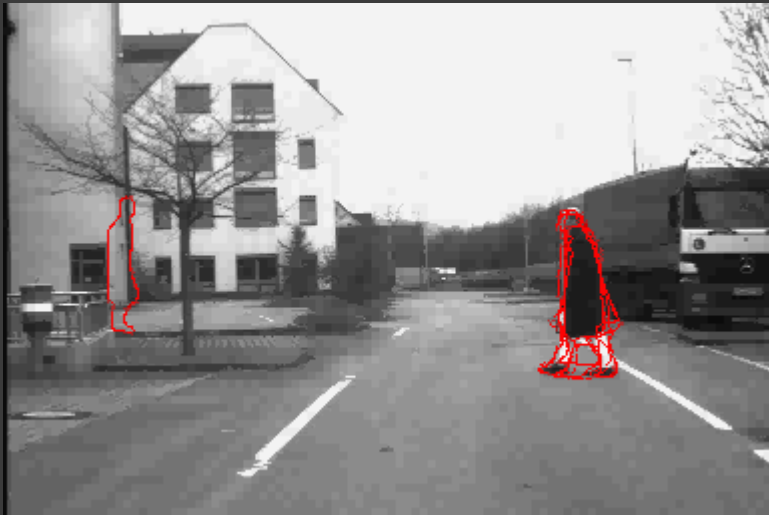
- Our Goal:
 - We want to assist the driver:
 - Avoid Pedestrians
 - Recognize Road Signs



Their solution

- ⦿ Using Chamfer Distance (w/ DT) and Hierarchical Organization
- ⦿ Optimized code for utilizing Pentium 2's MMX instructions.
- ⦿ Road Signs:
 - 36 templates (circles and triangles)
 - 80% to 95% recognition rate, based on visibility
- ⦿ Pedestrians:
 - 1100 shapes, 5 scales = 5500 templates
 - Preliminary results showed 75%-80% when requiring false positives to be 2 or less.

Videos



Contour Matching Summary

- ⦿ Speed is impressive, DT and Hierarchical Organization can improve the speed dramatically.
- ⦿ Detection results are only as good as the underlying edge-data.
- ⦿ Ambiguity and False Positives are a serious issue. Oriented edges can potentially help.
- ⦿ Permutations of Scale, and 3D transformations are still an issue, both in template generation and matching
- ⦿ It is questionable whether this system is ready for production.
 - Most Modern Smart Cars seem to rely mainly on active sensors (LIDAR & RADAR)

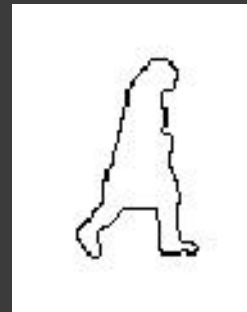
Outline

- So far, we have looked at:

- Color



- Contours (outer shapes)



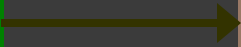
- What's missing?

“Bag of Words”: recognition using texture

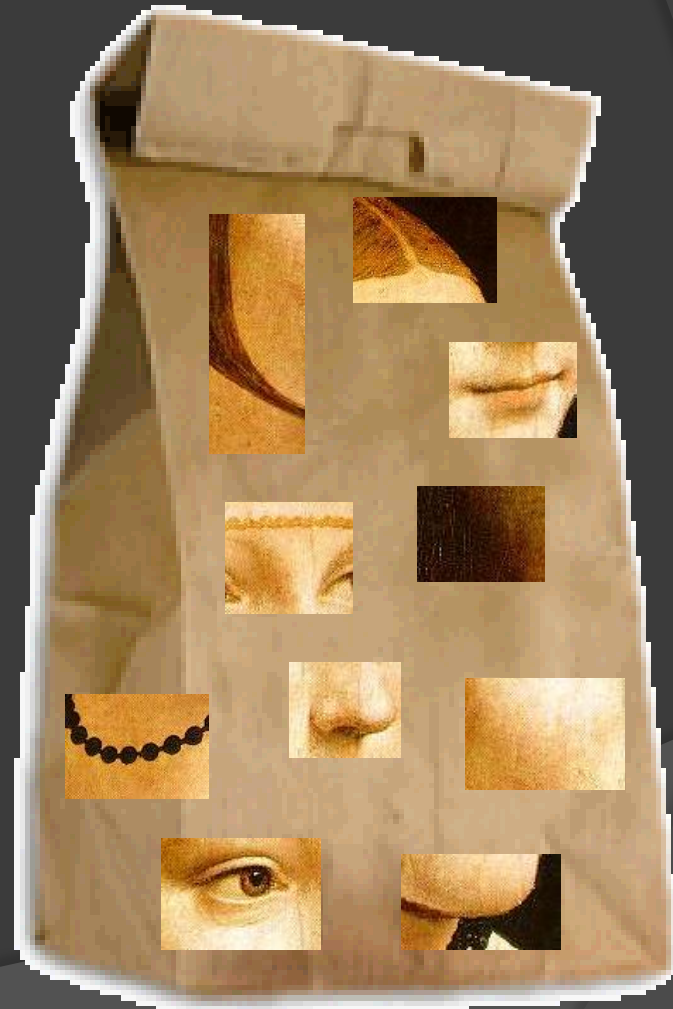


Adopted from Alexei Efros and Fei-Fei Li, with
some slides from L.W. Renninger

Object




Bag of 'words'



Analogy to documents

Of all the sensory impressions proceeding to the brain, the visual experiences are the dominant ones. Our perception of the world around us is based essentially on the messages that reach our eyes. For a long time, the retinal image was considered as a movie screen. It was discovered that the visual centers in the brain are a more complex system following the path to the various cells of the cortex, Hubel and Wiesel have demonstrated that the *message about the image falling on the retina undergoes a point-by-point analysis in a system of nerve cells stored in columns. In this system each cell has its specific function and is responsible for a specific detail in the pattern of the retinal image.*

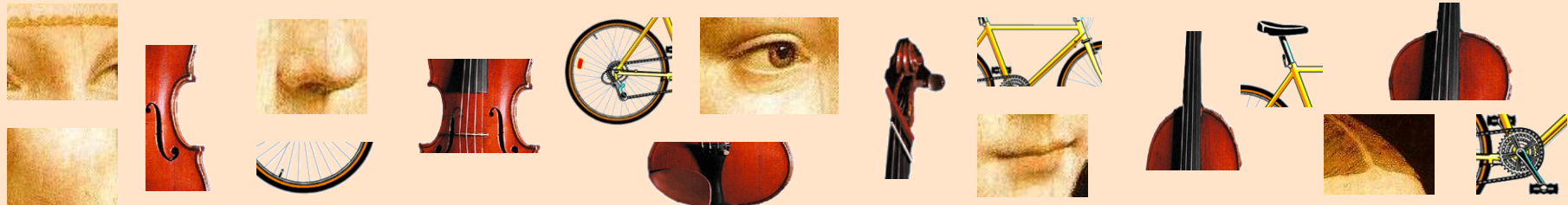
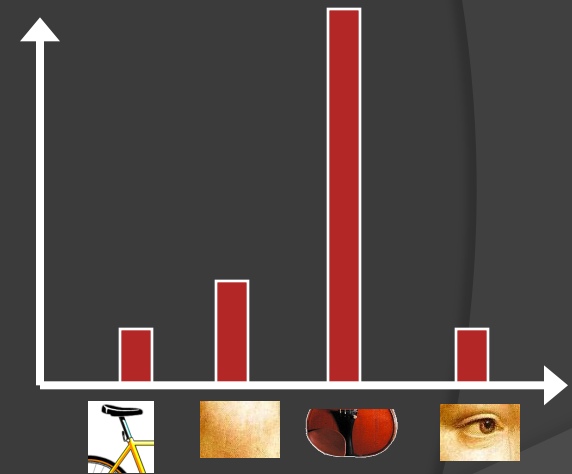
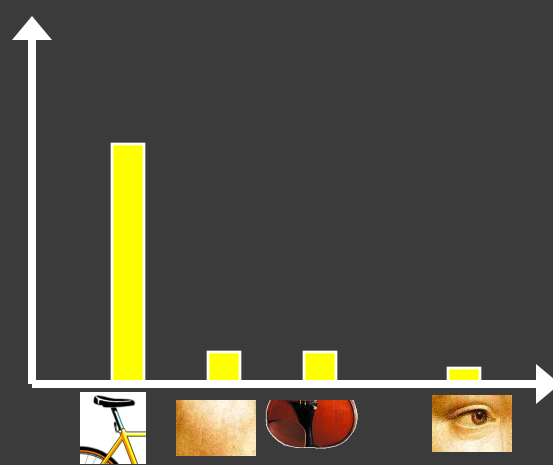
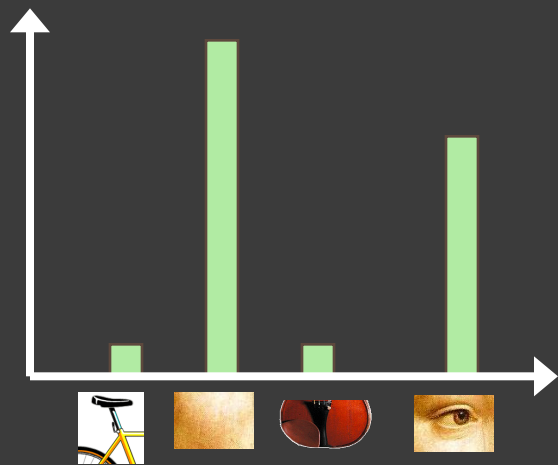


**sensory, brain,
visual, perception,
retinal, cerebral cortex,
eye, cell, optical
nerve, image
Hubel, Wiesel**

China is forecasting a trade surplus of \$90bn (£51bn) to \$100bn this year, a threefold increase on 2004's \$32bn. The Commerce Ministry said the surplus would be created by a predicted 30% increase in exports to \$750bn, compared with \$560bn in 2004. The surplus will annoy the US because it will reduce the value of China's exports. The US government has agreed to let the yuan rise against the dollar. The US government also needs to reduce the demand for exports from the country. China has agreed to let the yuan against the dollar rise and permitted it to trade within a narrow band but the US wants the yuan to be allowed to rise freely. However, Beijing has made it clear that it will take its time and tread carefully before allowing the yuan to rise further in value.



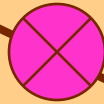
**China, trade,
surplus, commerce,
exports, imports, US,
yuan, bank, domestic,
foreign, increase,
trade, value**



learning



feature detection
& representation



codewords dictionary

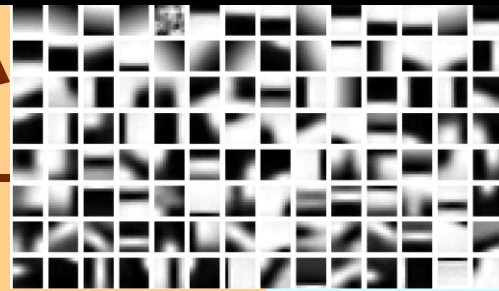
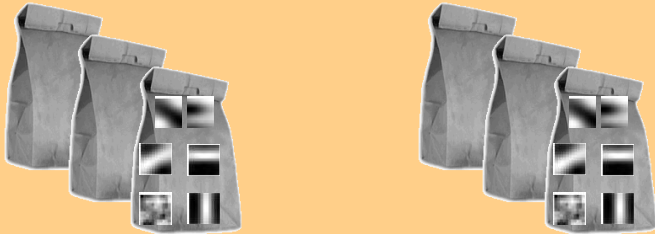
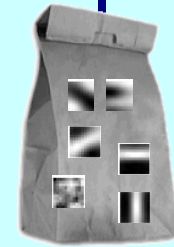
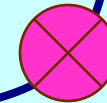


image representation



**category models
(and/or) classifiers**

recognition



**category
decision**

1. Feature detection and representation



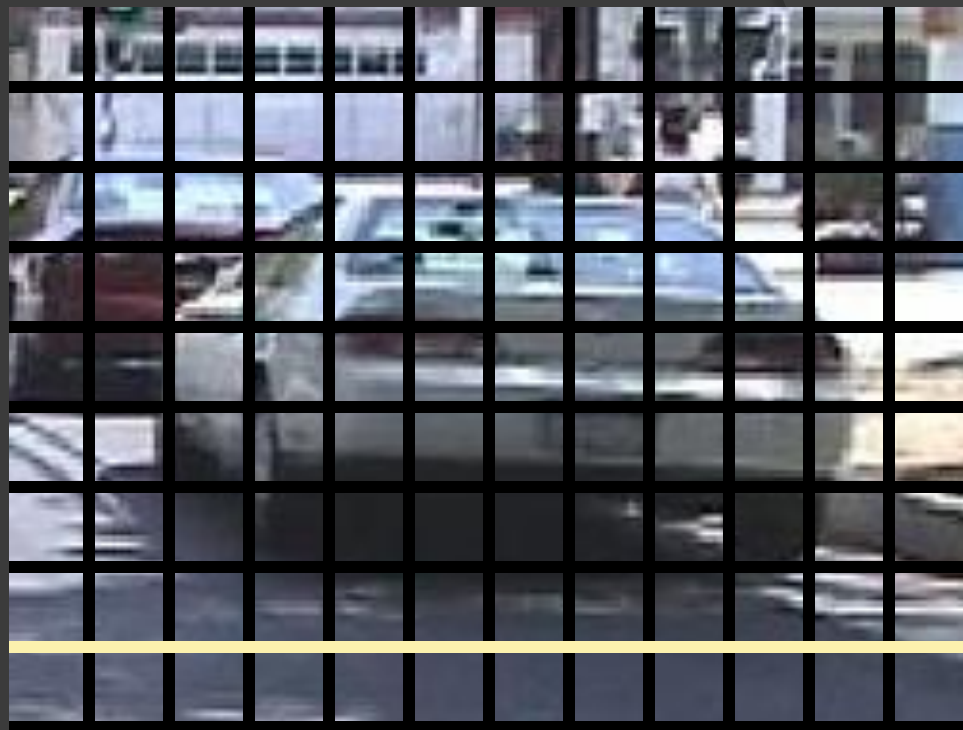
Feature detection

- ◎ Sliding Window
 - Leung et al, 1999
 - Viola et al, 1999
 - Renninger et al 2002



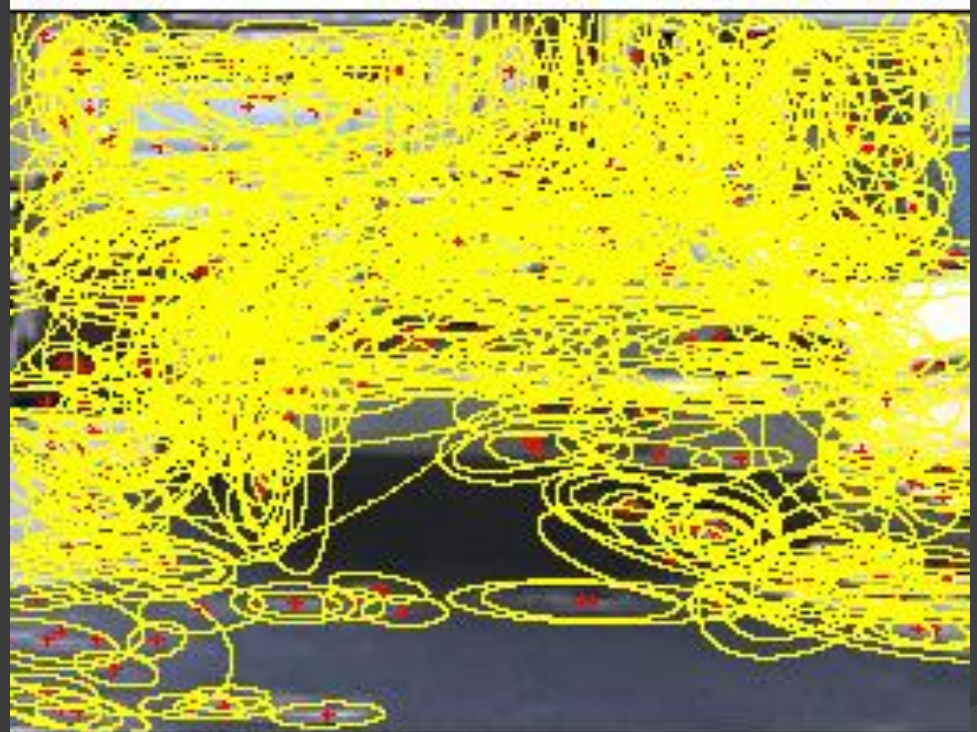
Feature detection

- ◎ Sliding Window
 - Leung et al, 1999
 - Viola et al, 1999
 - Renninger et al 2002
- ◎ Regular grid
 - Vogel et al. 2003
 - Fei-Fei et al. 2005



Feature detection

- ⊙ Sliding Window
 - Leung et al, 1999
 - Viola et al, 1999
 - Renninger et al 2002
- ⊙ Regular grid
 - Vogel et al. 2003
 - Fei-Fei et al. 2005
- ⊙ Interest point detectors
 - Csurka et al. 2004
 - Fei-Fei et al. 2005
 - Sivic et al. 2005



Feature Representation

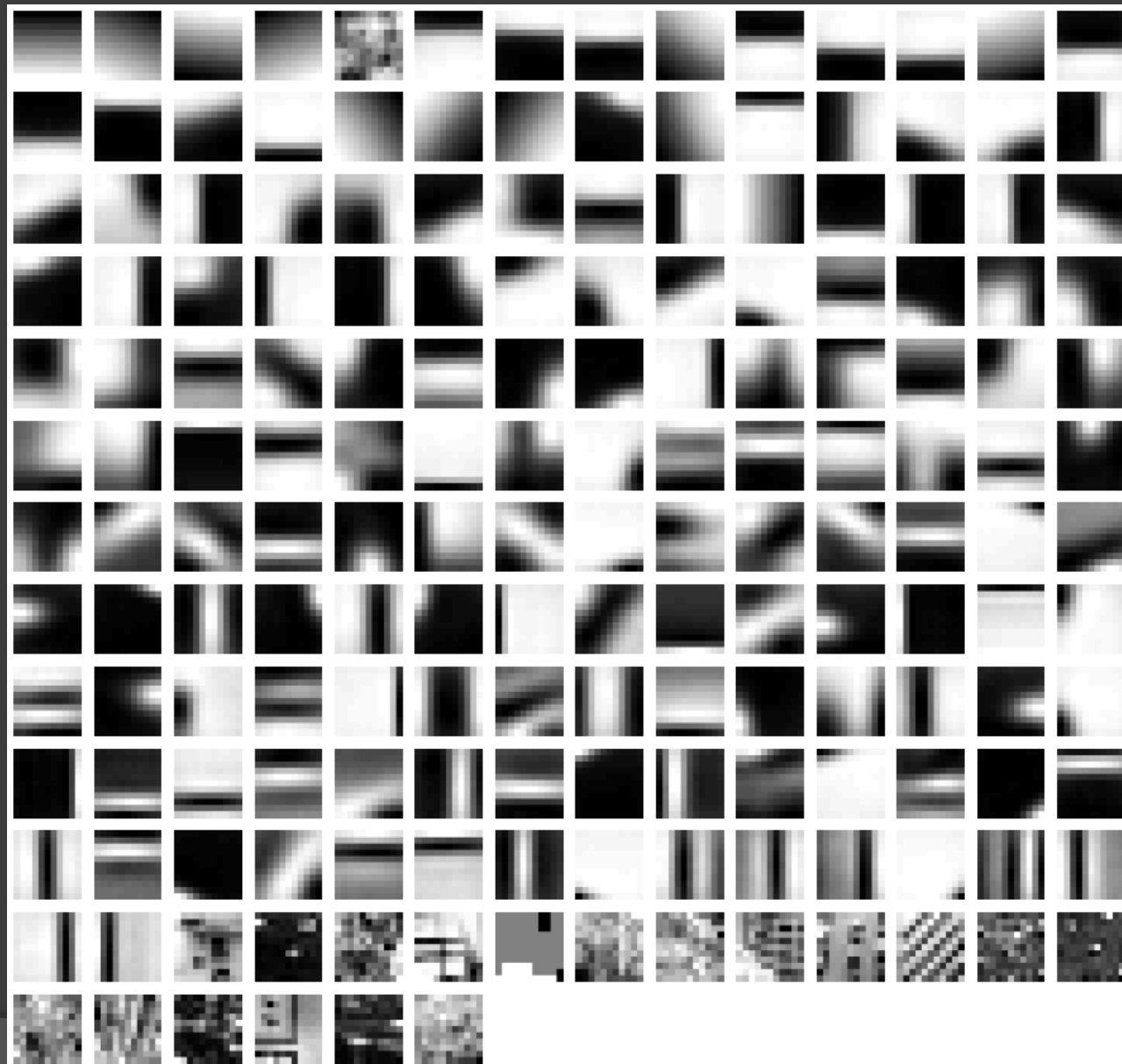
Visual words, aka textons, aka keypoints:
K-means clustered pieces of the image

⦿ Various Representations:

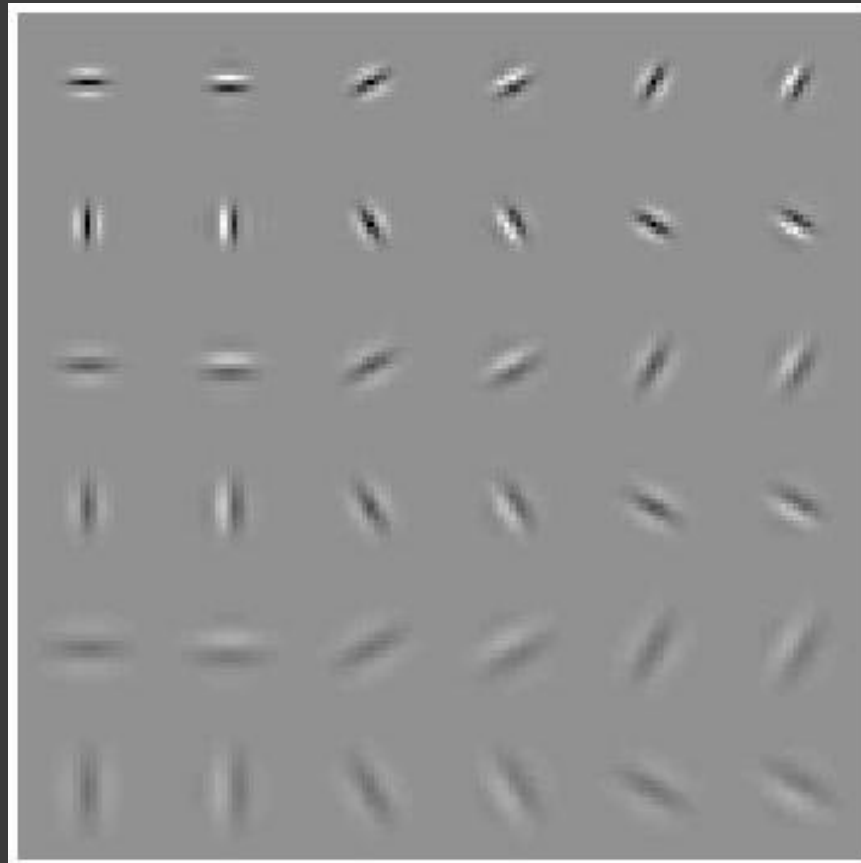
- Filter bank responses
- Image Patches
- SIFT descriptors

All encode more-or-less the same thing...

Clustered Image Patches

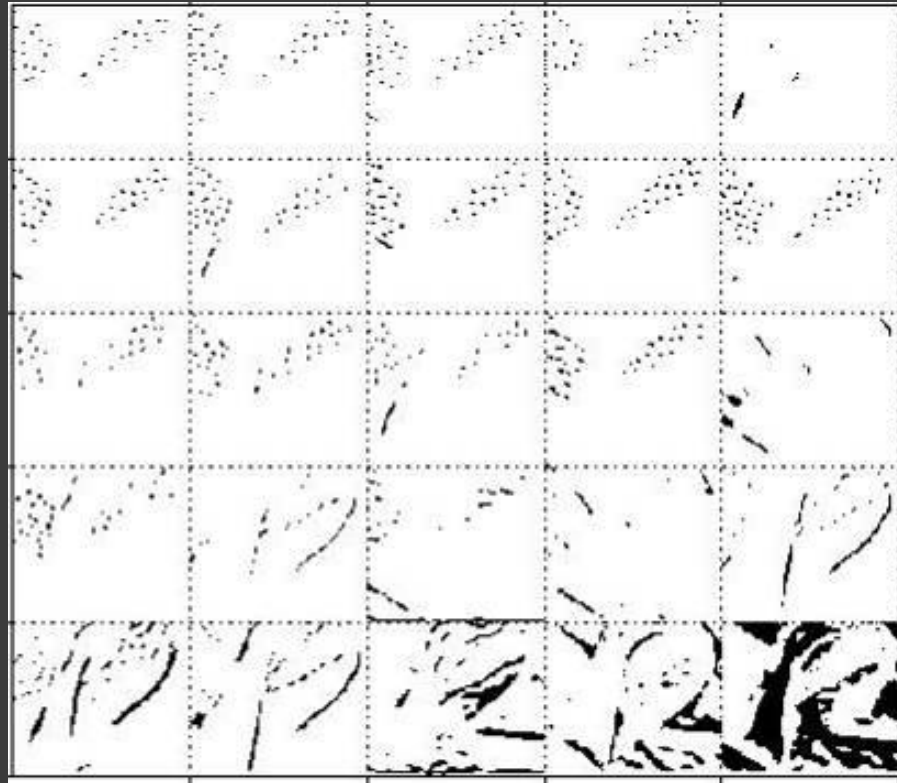
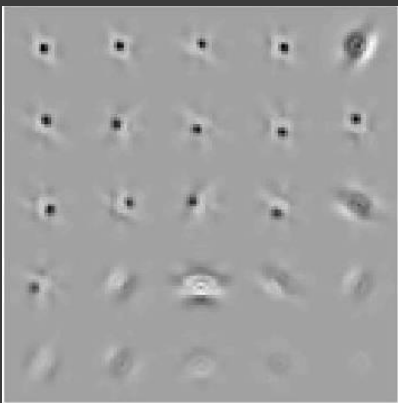
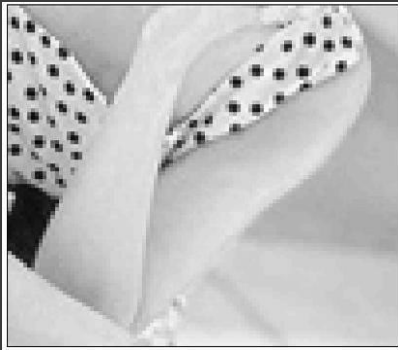


Filterbank

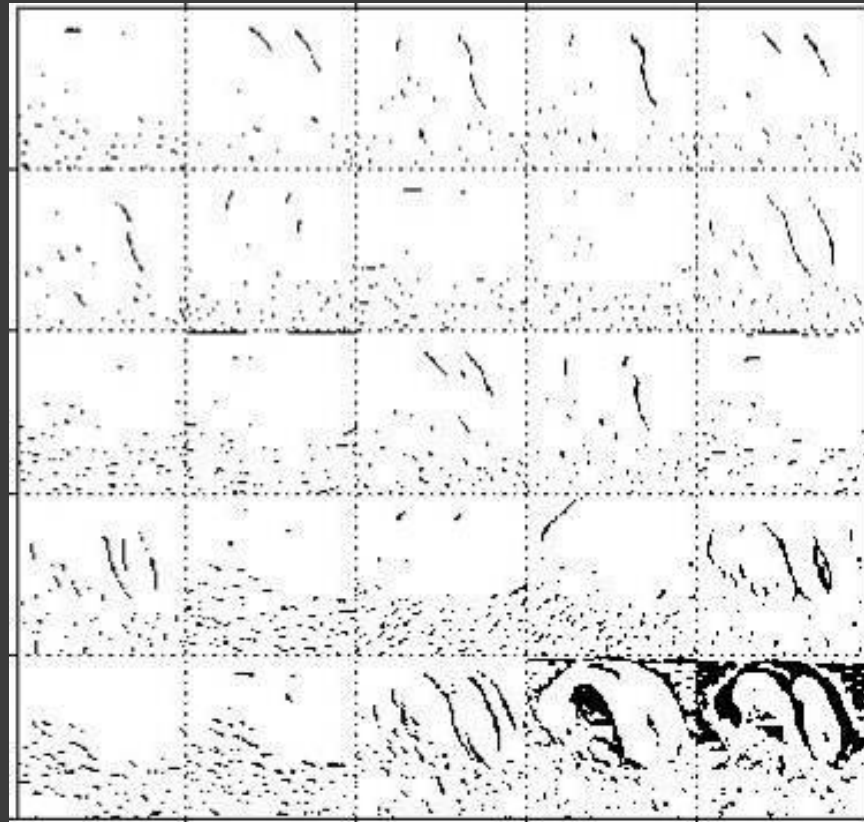
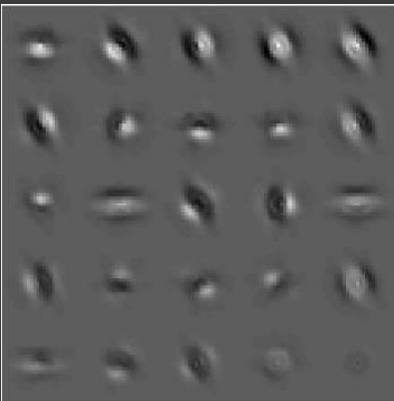


Textons (Malik et al, IJCV 2001)

- ⦿ K-means on vectors of filter responses



Textons (cont.)



Object Recognition using texture

Object Categorization by Learned Universal Visual Dictionary

J. Winn, A. Criminisi and T. Minka

Microsoft Research, Cambridge, UK – <http://research.microsoft.com/vision/cambridge/recognition/>

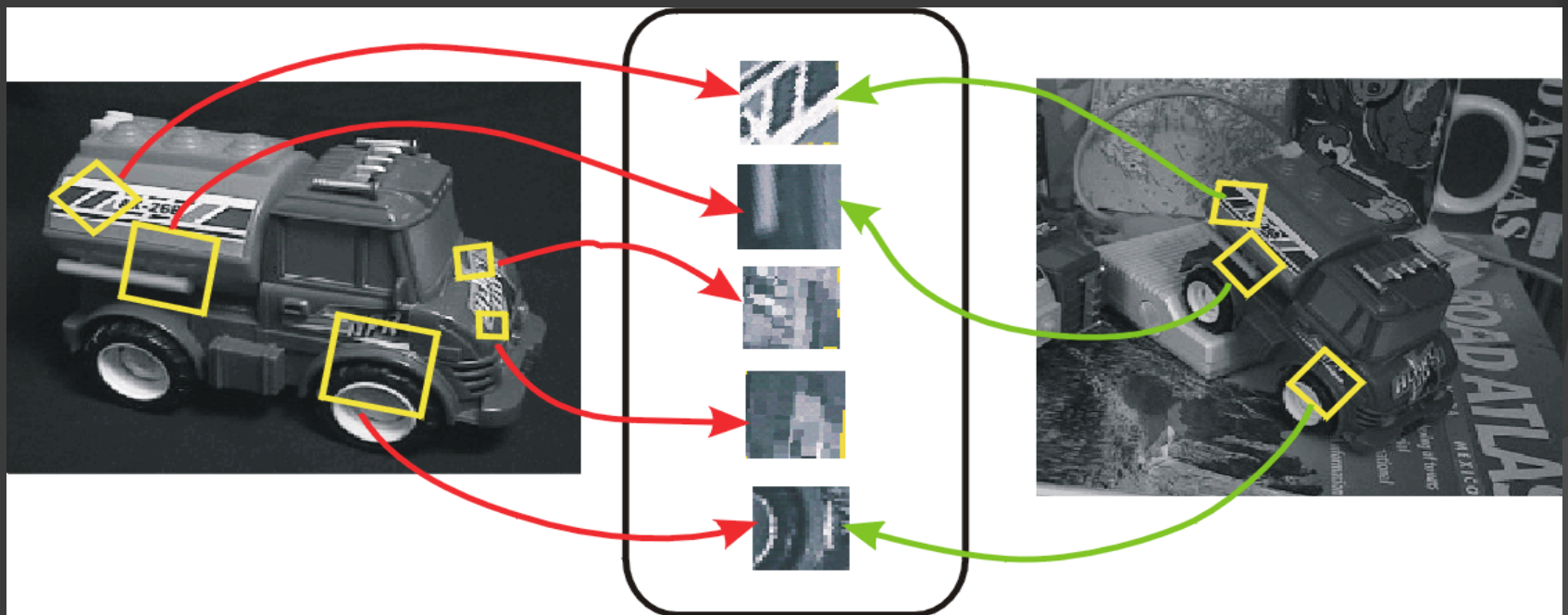


SIFT (Scale Invariant Feature Transform)

- ◎ Sift in one line:
 - A well-engineered feature descriptor

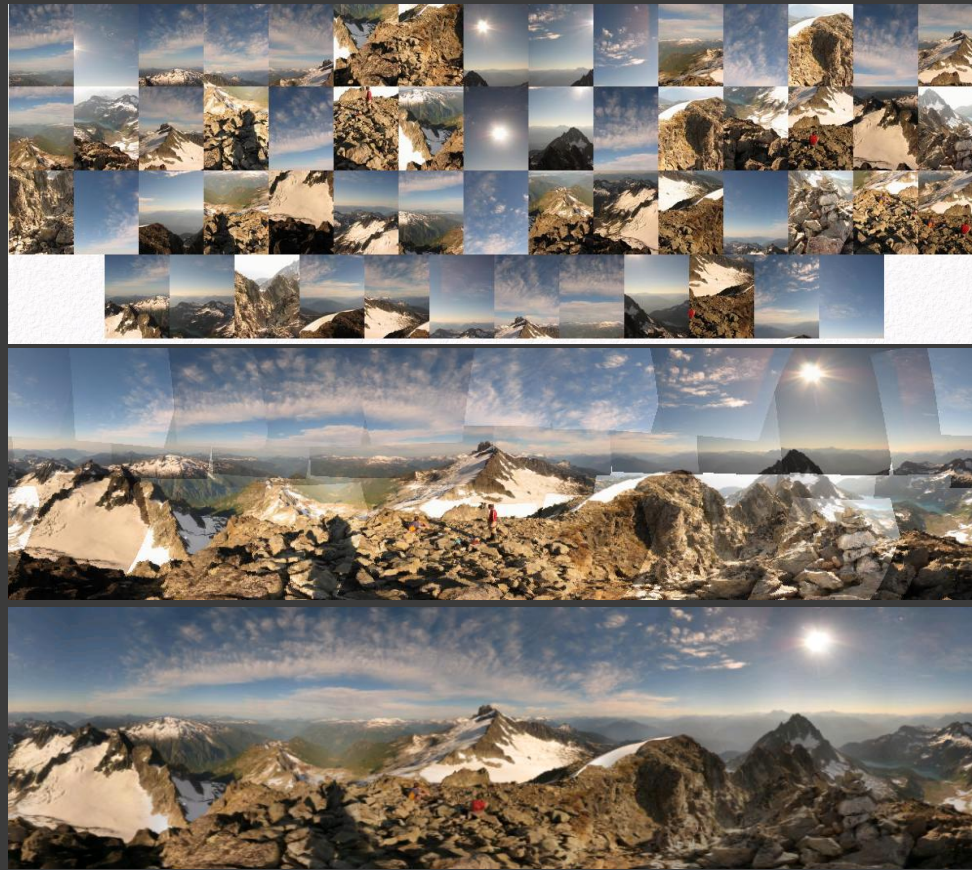
Introduction

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



SIFT Features

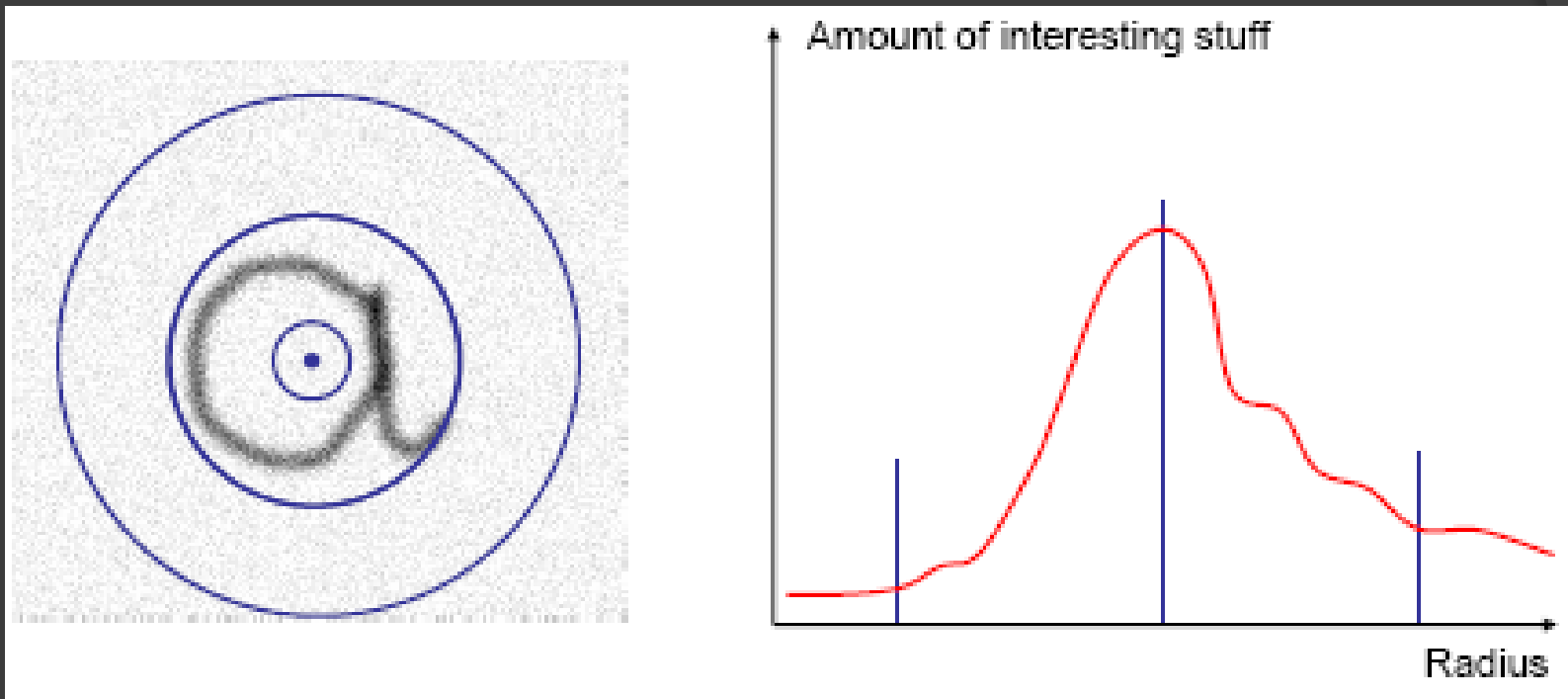
You might have already used SIFT (or a similar descriptor)



© <http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html>

Finding Keypoints – Scale, Location

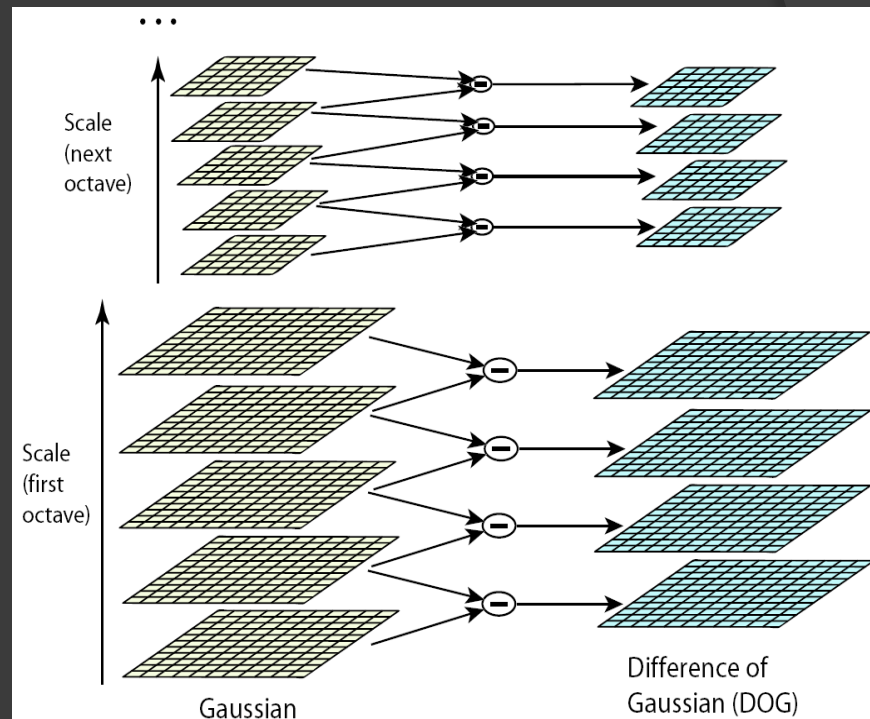
- How do we choose scale?



SIFT (Scale Invariant Feature Transform) review

Transform) review

- Compute the difference of Gaussians on several levels of the scale-space pyramid.
- Orientation vectors are computed for peaks in the Difference of Gaussians.
- SIFT feature is a 128 dimensional vector, combining the orientation histograms of locations closely surrounding the keypoint in scale-space.
- Invariant to scale & to rotation parallel to the image plane (roll).
- Robust against perspective rotation (yaw & pitch) to about 10-20 degrees.



Picture from [Lowe 2004]

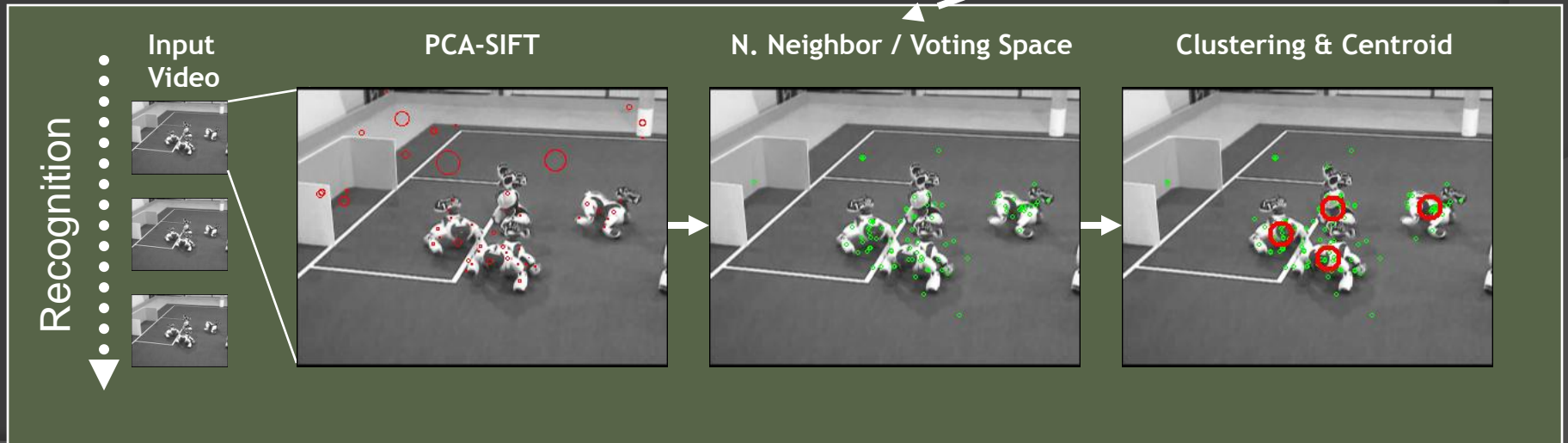
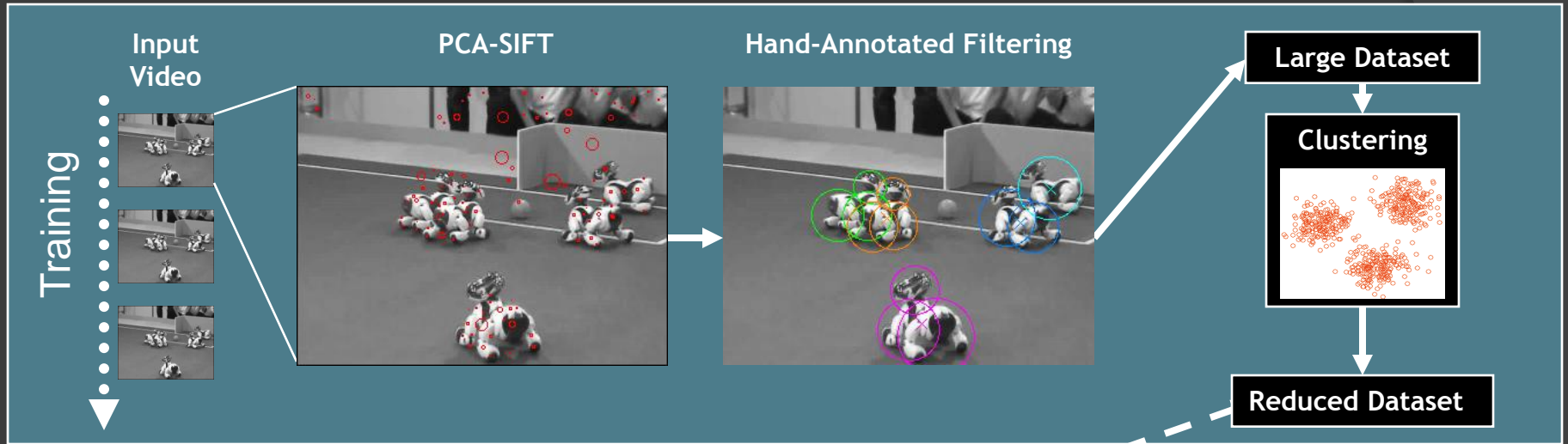
Detection of Multiple Deformable Objects Using PCA-SIFT

- ◉ Detection of Multiple Deformable Objects Using PCA-SIFT (S. Zickler & A. Efros, in Proceedings of AAAI 2007)
- ◉ Detection and Localization of Multiple Objects (S. Zickler & M. Veloso, in Proceedings of Humanoids 2006)

Why PCA-SIFT?

- ◎ SIFT is good
 - Has been shown to work well on still images
 - Provides Scale Invariance (which is a must for humanoid vision)
 - Provides rotational robustness of up to about +/- 20 degrees
- ◎ PCA-SIFT is better
 - Reduces dimensionality of SIFT using Principal Component Analysis (PCA).
 - This paper used reduction from 128-D SIFT to 20-D PCA SIFT.
 - Shown to perform as well as SIFT [Ke & Sukthankar 2004].
- ◎ We use a modified version of the PCA-SIFT implementation provided by [Ke & Sukthankar 2004].

Algorithm Overview



Training Step 1 – PCA SIFT Feature Generation on Training Data

- We run the PCA SIFT feature detector on our training footage.
- Training footage should contain various perspectives and poses of our target object.
 - SIFT is only robust to a maximum of ± 20 degrees of rotation.
- This will generate a lot of sample points.
- Example: the World through the eyes of a SIFT-detector:

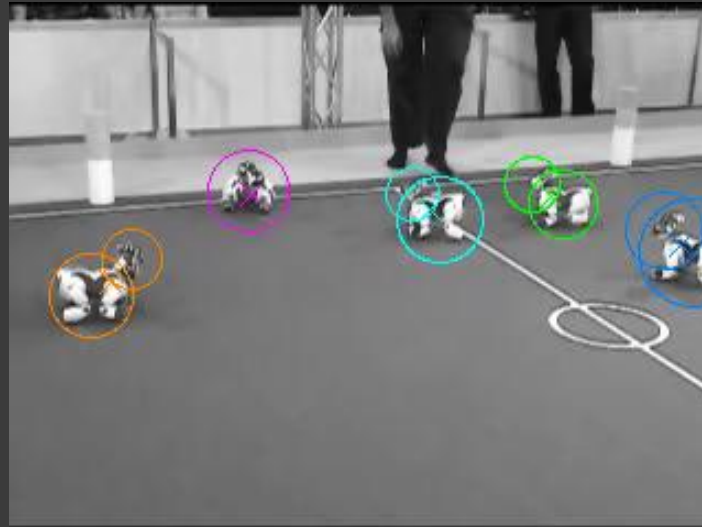


Training Step 2 - Annotation

- ⦿ We have generated SIFT-features of the entire video
- ⦿ We want to
 - keep the ones relating to our object of interest
 - reject everything else
- ⦿ Unsupervised training can be difficult
 - Why?
 - training footage might be highly interactive
 - training object might not be automatically distinguishable from its background
- ⦿ We use manual annotation

Annotation (contd.)

- Each object's shape is annotated using geometric primitives (e.g. circles)
- Centers of mass is computed for each object
- We store relative location of each feature vector in relation to the center of mass
- SIFT features outside of annotation mask are rejected
- 3 videos, each about 500 frames, ~4 dogs w/ ~2 circles each = 12000 Circles to draw (ok it was a little easier than that!)



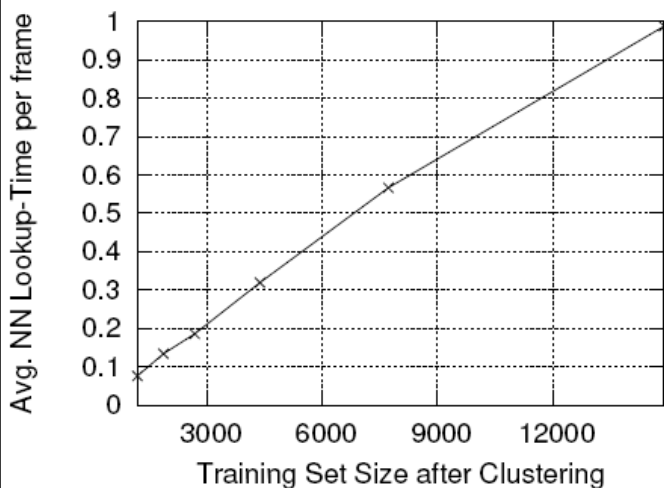
Training Step 3 – Post Processing

- ⦿ A typical 15 second annotation-filtered video still produces about 40,000 SIFT vectors.
 - Many of them are redundant (the object might not change much between two frames)
 - 40,000 is also a bit too large to perform real-time recognition.

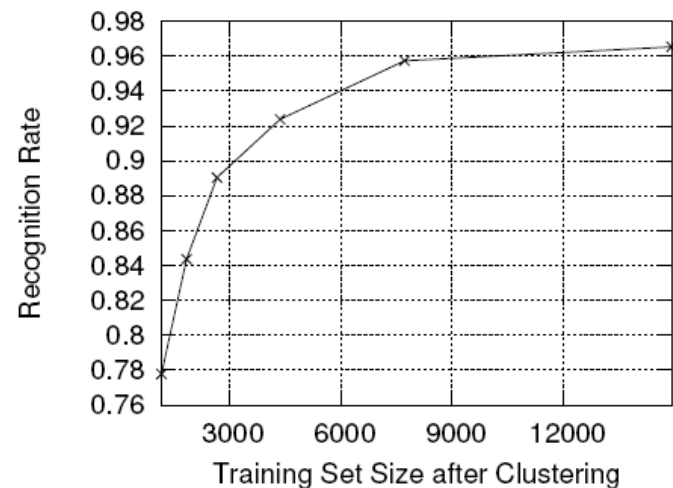
Clustering

- ⦿ We use Agglomerative Clustering to combine vectors that are closer than a particular threshold.
 - Combining two vectors is performed by computing the mean for each of their dimensions.
 - Unlike k-means, agglomerative clustering does not need to know k (the amount of clusters) a priori. Instead, our only stopping condition is the distance threshold itself.

The effects of post-processing the training data



(a) Runtime

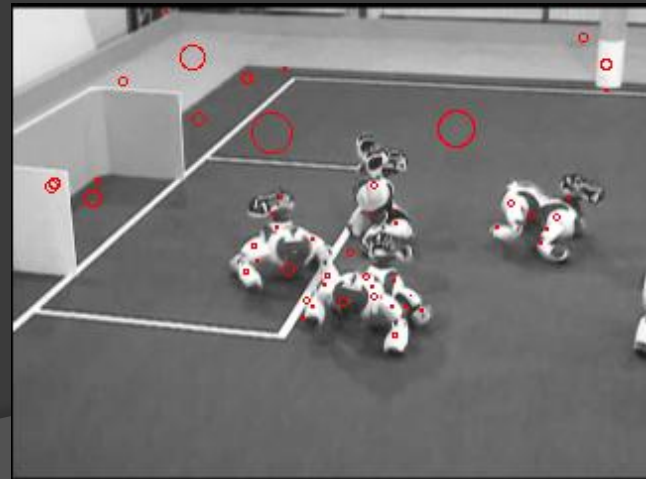


(b) Recognition Rate

- Cutting the training size in half (from 15000 to 7500) will reduce computational time by almost 50%, but it only hurts recognition rate by less than 2% !
- This is very good news!

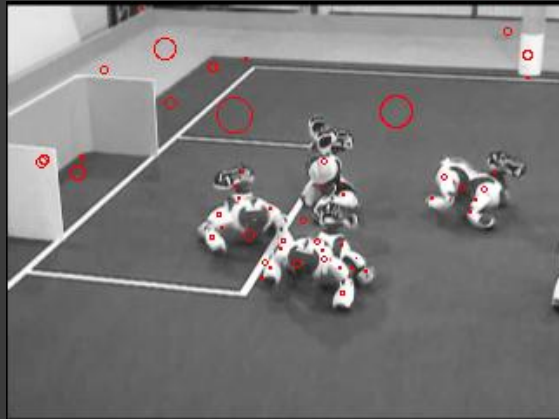
Recognition Step 1 – Find PCA-SIFT vectors and NN-lookup

- ⦿ We generate all PCA-SIFT features for the incoming frame
- ⦿ We perform a simple nearest neighbor lookup with the (reduced) training set, using a simple distance threshold (in Euclidean space)
- ⦿ Greater threshold can increase recognition rate, but also increases likelihood of false positives.
- ⦿ Smaller threshold can decrease false positives, but might hurt recognition rate.



Voting Space

- Each matching feature votes for where it predicts the relative center of the object (using the same offset that it was trained with, but scaled and rotated to match the detected feature)



PCA-Space



Voting-Space

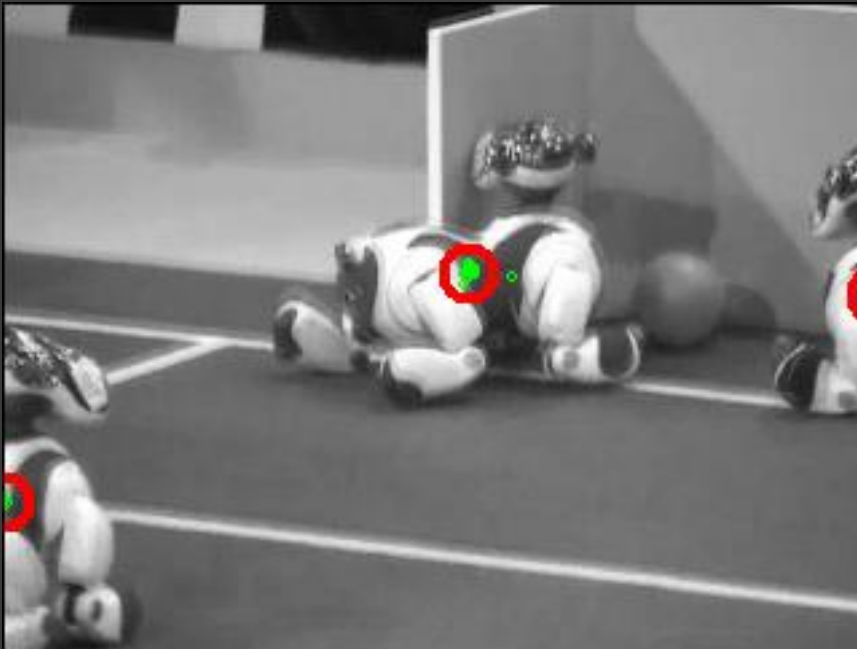
Voting Space (continued)

- We need to cluster our votes (find peaks in voting-space)
- Agglomerative Clustering vs. Mean-Shift
 - Mean-Shift is faster ($\sim \log(n)$), but uses random heuristics
 - Agglomerative clustering is precise but is n^2
 - Experiments show that both yield same quality of results
- We reject all clusters below a certain minimum size, this avoids outlier-votes.
- We compute the centroid of each cluster. This is our final localization result.

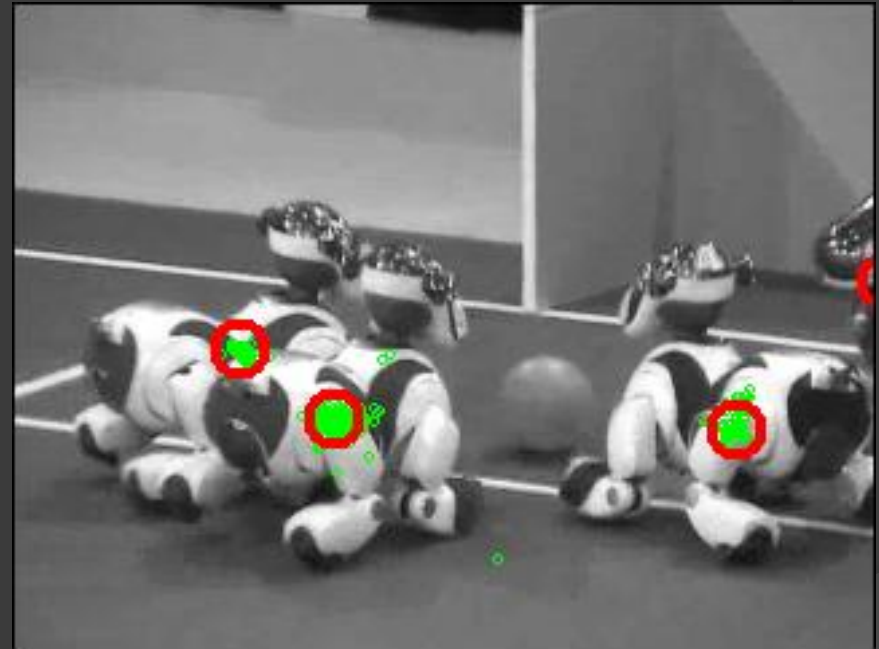


Voting Space contd.

◉ Solving Occlusion:



Objects are off the screen.



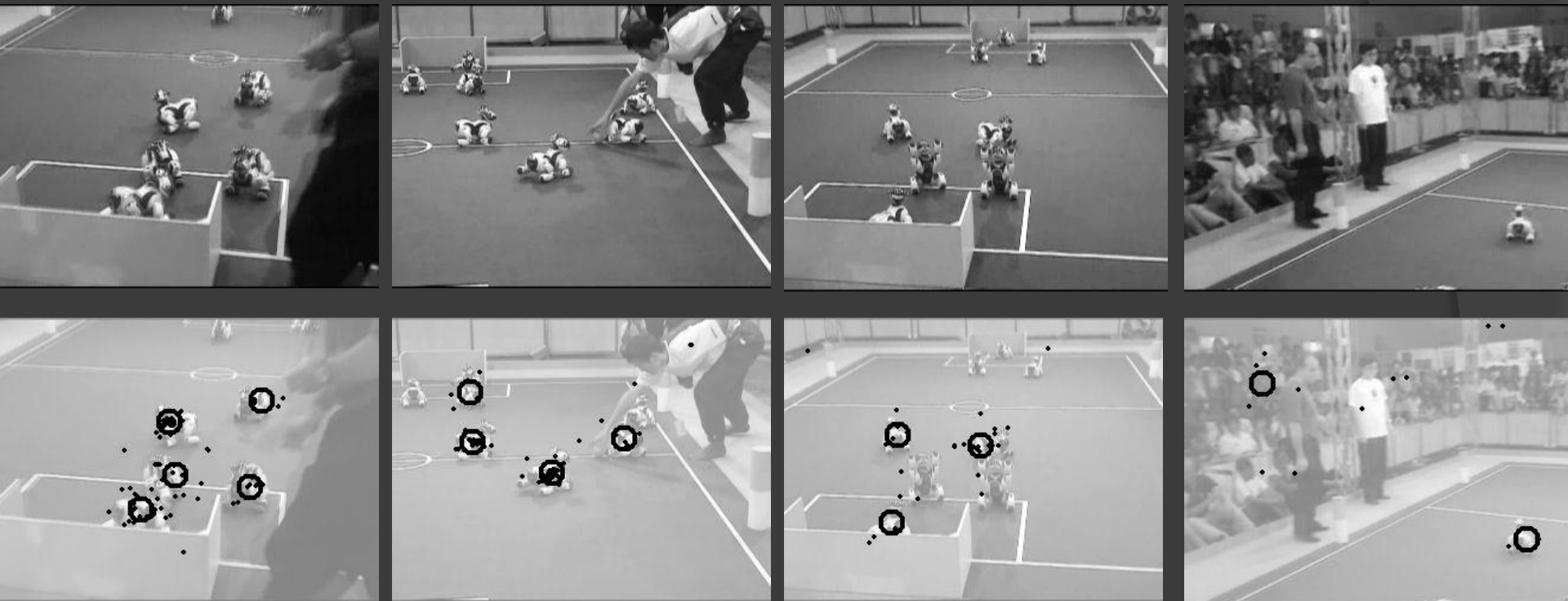
Objects are occluding each other.

Results

- ⦿ AIBO RoboCup-domain
 - grayscale, 320x240 pixel
 - Reduced from 110,000 features (post-annotation) down to 15,000 features
 - Training & Testing came from different angles + different games
- ⦿ QRIO domain
 - grayscale, 320x240
 - from 132,000 down to 4,575 features
 - Training & Testing came from the robot performing a different “dances”.

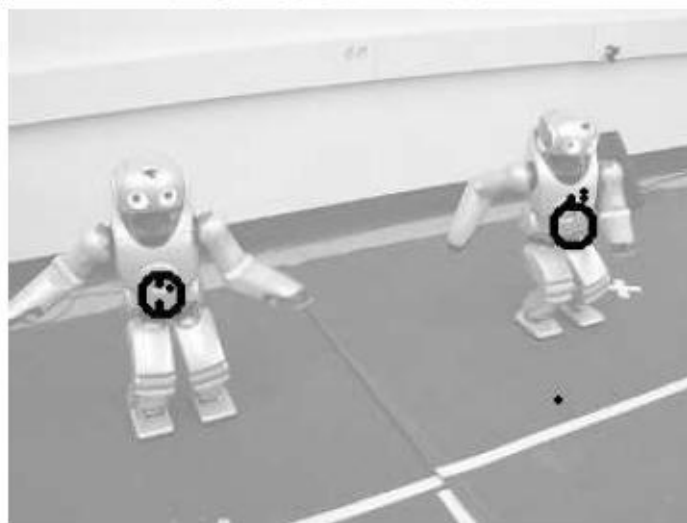
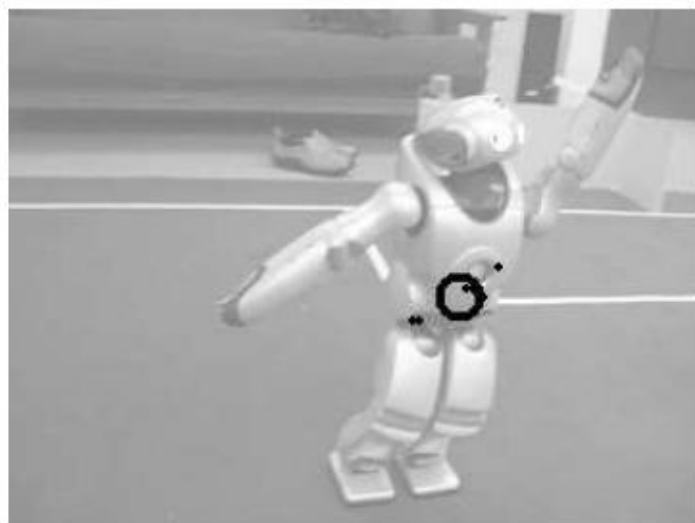
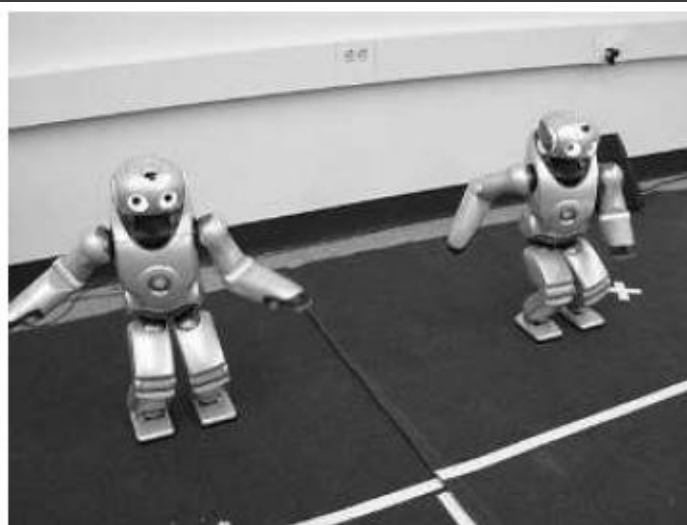
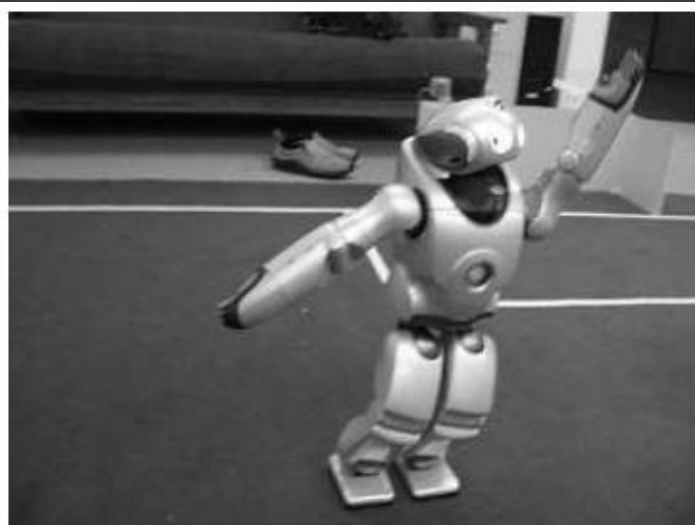


Visual Results (Aibo-Domain)



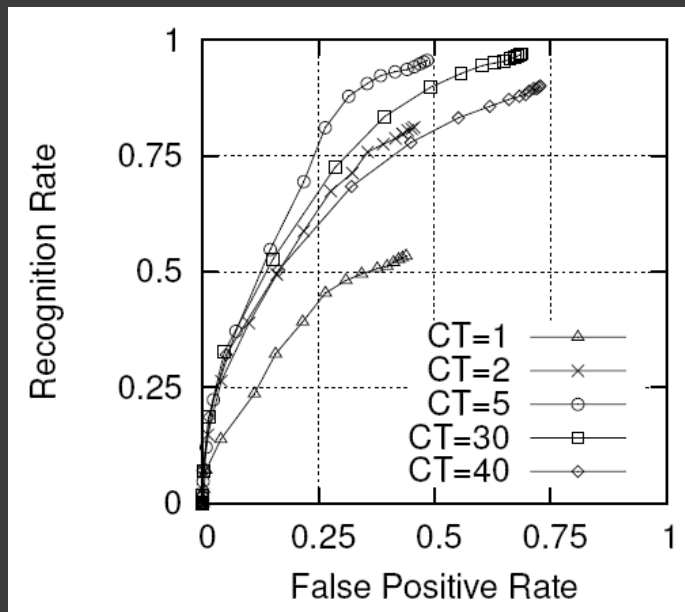
Small dots are votes; Larger Circles are final clustering results on the voting space (centroids)

Visual Results (QRIO Domain)



Results (continued)

- ROC curves show trade-off between detection & false positives
- QRIO domain performed better than AIBO-domain (not really a surprise)

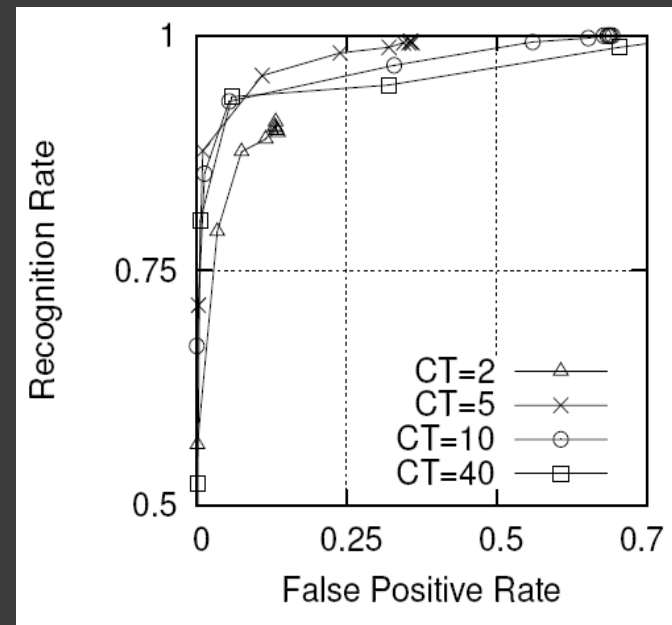


AIBO domain

“CT” = clustering threshold

~80% recognition at ~25% false positives

~90% recognition at ~35% false positives



QRIO domain (note different Y-scale)

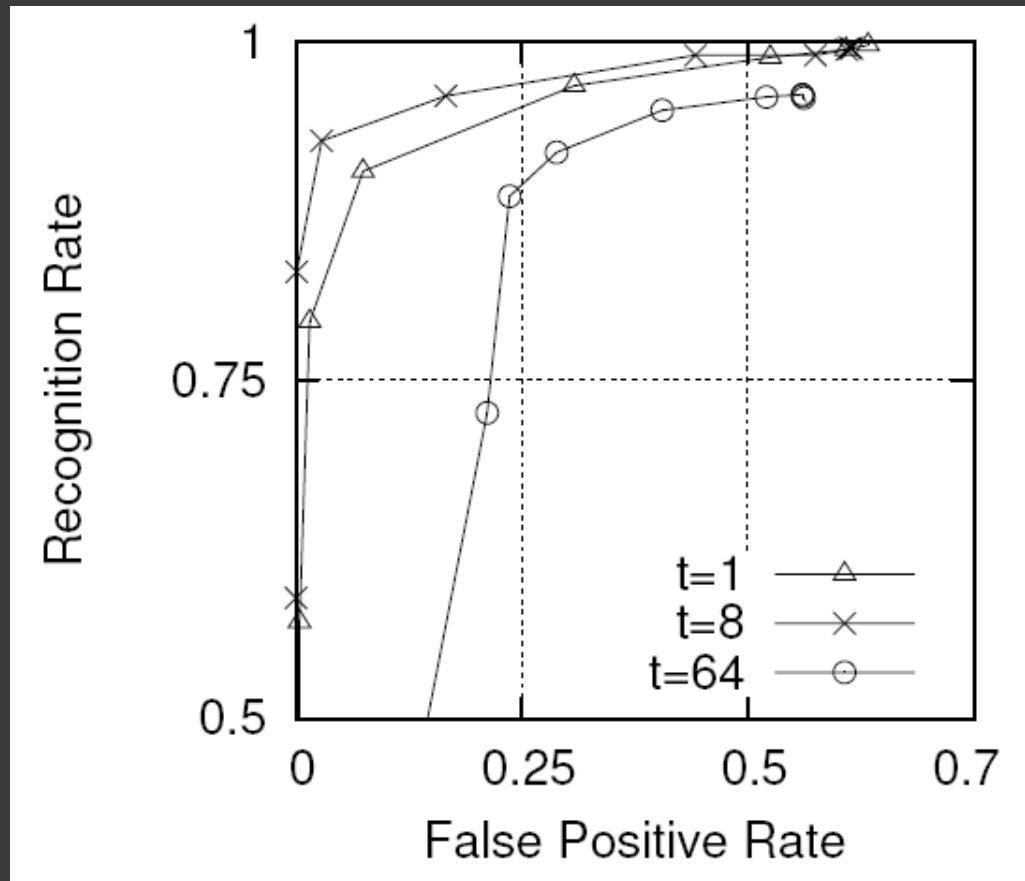
~90% recognition at < ~10% false positives

~98% recognition at 25% false positives

Temporal Clustering

- ⦿ Let's use the fact that we are working with continuous video data in order to reduce false positives caused by occasional outliers.
- ⦿ We do so by using the age of each frame as the third dimension of our voting space. We cluster over the N most recent frames.
 - We use a scalar multiplier to normalize the time dimension to match the magnitude of the spatial dimensions.
- ⦿ A greater value of N will increase the temporal filtering effect.
- ⦿ But, a greater value of N will also increase temporal lag. Making the current result less accurate.

Temporal Clustering can help



Sample Video

[open folder](#)

Conclusion

- ◎ PCA-SIFT is a feasible approach for detection of multiple non-rigid objects in various robotics applications.
- ◎ A grain of salt:
 - Why not detect humans?

The goal of vision

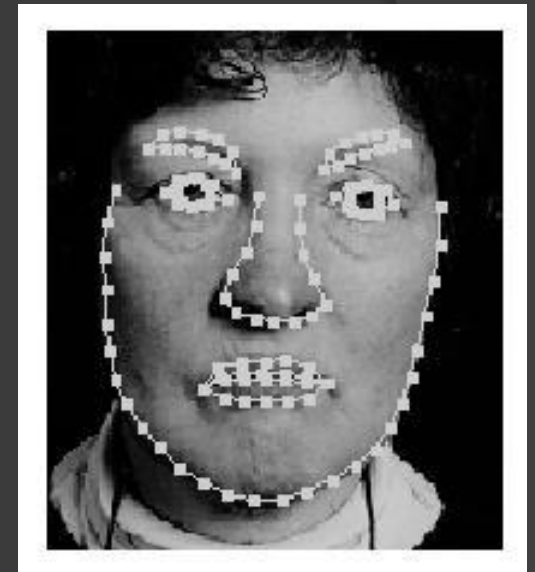
- ◎ Remember what vision is trying to achieve:
 - Understand the underlying 3D representation that creates the 2D image that we see.
 - Interestingly, that's the inverse problem of computer graphics.

Active Appearance Models

- ⦿ Let's start out with the following assumptions:
 - There exists an underlying 3D model
 - We already know about its basic structure
- ⦿ http://www.ri.cmu.edu/projects/project_448.html

ASMs

- ⦿ Idea: Active Shape Model (ASM)
 - Training:
 - Label facial images with keypoints
 - Find the principal components (using PCA)
 - Application:
 - Start with mean-image from training set
 - Modify component-weights until image matches the target.



ASM properties

- ⦿ Fast
- ⦿ Does not make use of textures

CAMs

- ◎ Slightly better: Combined Appearance Model (CAM)
 - Adds grey-level information on top of keypoints
 - Same procedure: generate principal components
 - During matching:
 - We again fit keypoints to match new image
 - We warp all learned grey-level information to match new geometry of keypoints
 - We fit grey-level components to image

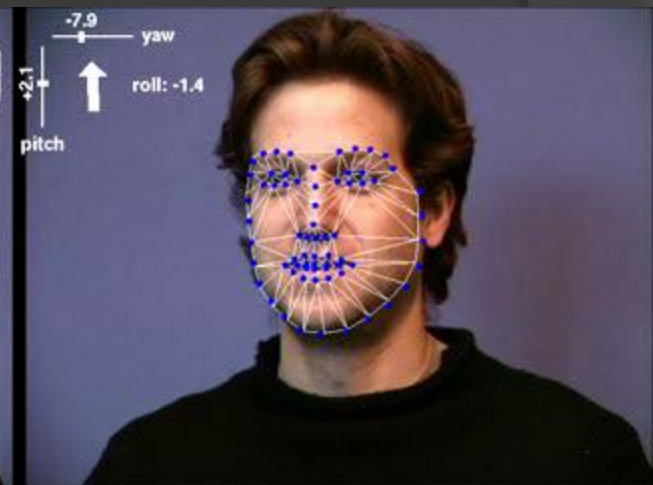
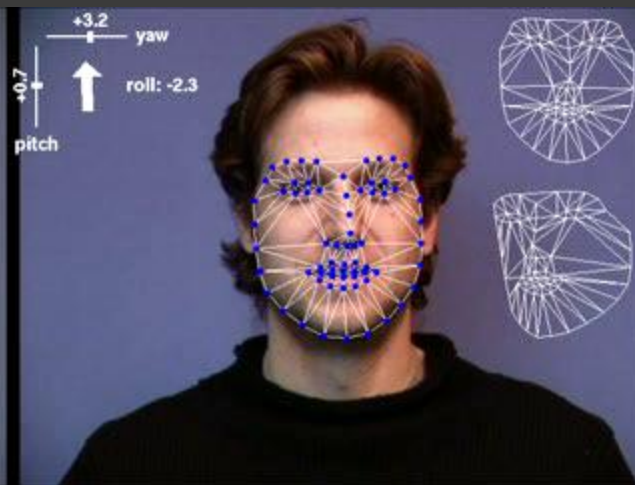
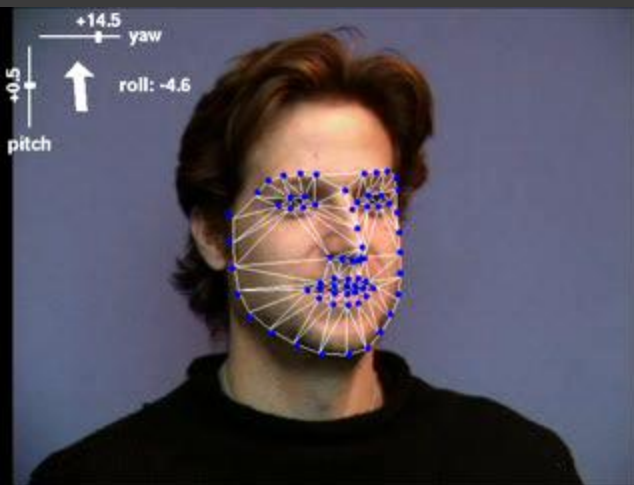
ASM and CAM problems

- ⦿ Not very robust...why?
 - New instances
 - Occlusion

AAMs

- ◎ Much Better: Active Appearance Models
 - Works on Unseen Images, by adapting parameters on the fly
 - Allows us to seamlessly fit our model to new imagery.
 - Recent work also solved the occlusion problem

Results



Handling Occlusions



Summary + What should *you* use?

Algorithm	Pros	Cons	Other Remarks
SIFT [Lowe 1999]	<ul style="list-style-type: none"> •Scale-Invariant •Rotational Robustness 	<ul style="list-style-type: none"> •High Dimensionality •Overfitting •Requires feature-richness 	Uses “inner features” only
PCA-SIFT [Ke & Sukthankar 2004]	<ul style="list-style-type: none"> •Same as SIFT & Reduced Dimensionality 	<ul style="list-style-type: none"> •Overfitting •Requires feature-richness 	Uses “inner features” only
Color-Segmentation (e.g. RoboCup) [Bruce et al. 2000]	<ul style="list-style-type: none"> •Simple To Implement •Very Fast 	<ul style="list-style-type: none"> •Requires “Color-Coded” Domain •Lighting Can be problematic 	Uses color only
Edge-Matching using Distance Transform Techniques [Gavrila & Philomin 1999]	<ul style="list-style-type: none"> •Relatively Simple 	<ul style="list-style-type: none"> •Combinatorial Explosion of Possible Shape Variations (can be overcome with good data structures) 	Uses “outer shape” only
Active Appearance Models [Schneiderman & Kanade 1999]	<ul style="list-style-type: none"> •Fast + Reliable 	<ul style="list-style-type: none"> •Requires Complex Statistical Model of the object to be detected. 	Works well for face-detection + modelling

Maybe...Use Multiple!

- A lot of recent work uses a combination of multiple classifier types.
- Achieving human-like perception:
 - We use color, edges, and texture!
 - We also use temporal reasoning
 - We use a lot of contextual high-level knowledge! We rely on our world being structured!