Object Recognition

15-494 Cognitive Robotics
David S. Touretzky &
Ethan Tira-Thompson

Carnegie Mellon
Spring 2012

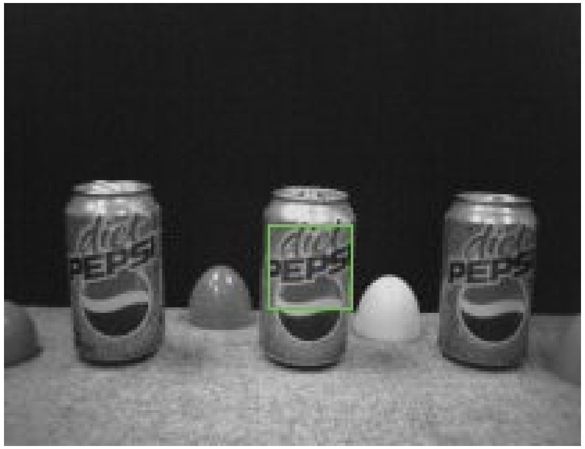
What Makes Object Recognition Hard?

- Translation invariance
- Scale invariance
- Rotation invariance (2D)
- Rotation invariance (3D)
- Occlusion
- Figure/ground segmentation (where is the object?)
- Articulated objects (limbs, scissors)

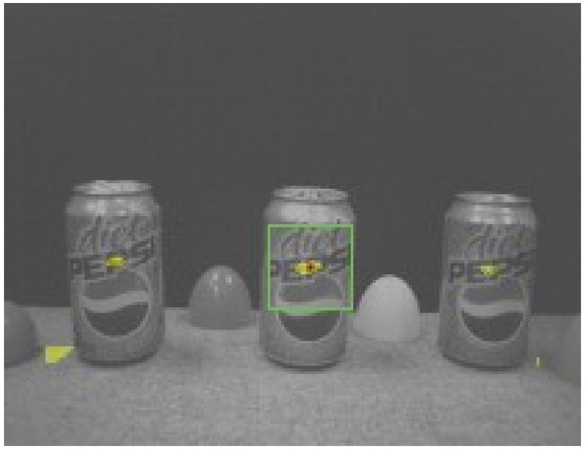
Template Matching

- Simplest possible object recognition scheme.
- Compare template pixels against image pixels at each image position.

Source image



Template

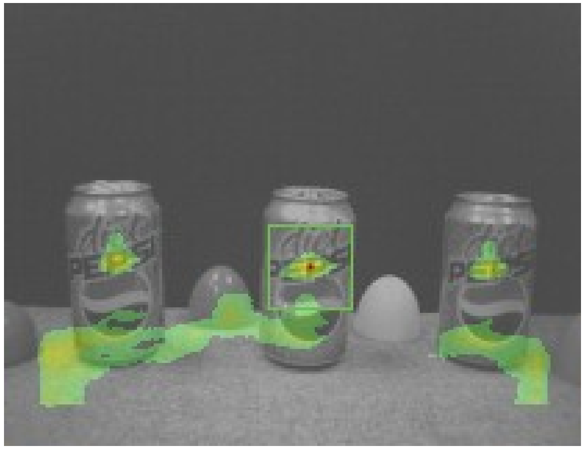


Match Score

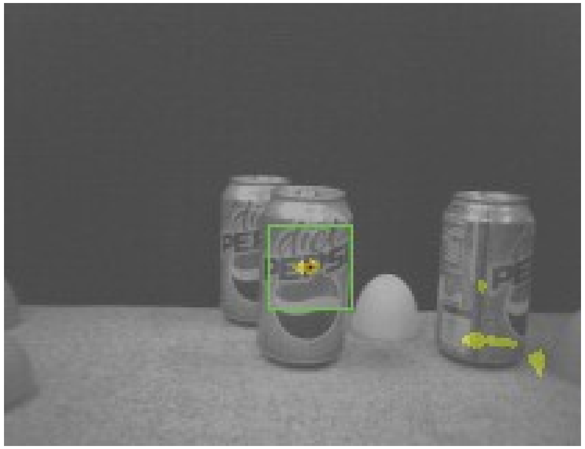
Template Matcher

```
Sketch<uint> templateMatch(const Sketch<uchar> &sketch,
     Sketch<uchar> &kernel, int istart, int jstart, int width, int height)
{
  Sketch<uint> result("templateMatch("+sketch->getName()+")",sketch);
  result->setColorMap(jetMapScaled);
  int const npix = width * height;
  int const di = - (int)(width/2);
  int const dj = - (int)(height/2);
  for (int si=0; si<sketch.width; si++)
    for (int sj=0; sj<sketch.height; sj++) {
      int sum = 0;
      for (int ki=0; ki<width; ki++)
        for (int kj=0; kj<height; kj++) {
          int k_pix = kernel(istart+ki,jstart+kj);
            if ( si+di+ki >= 0 && si+di+ki < sketch.width &&
                 sj+dj+kj >= 0 && sj+dj+kj < sketch.height ) {
              int s_pix = sketch(si+di+ki,sj+dj+kj);
              sum +=  (s_pix - k_pix) * (s_pix - k_pix);
            }
            else
              sum += k_pix * k_pix;
        }
      result(si,sj) =  uint(65535 - sqrt(sum/float(npix)));
    }
  result -= result->min();
  return result;
}
```

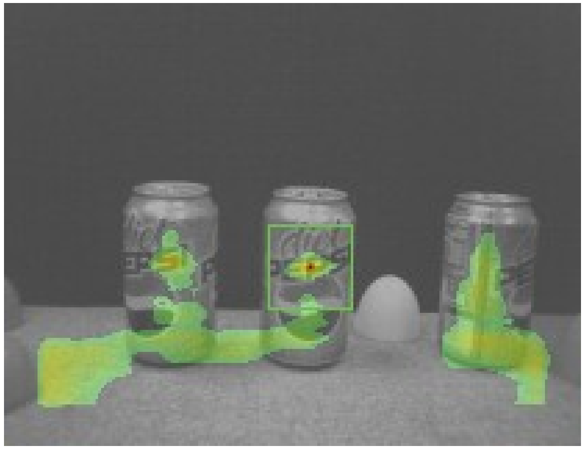
Limited Invariance Properties



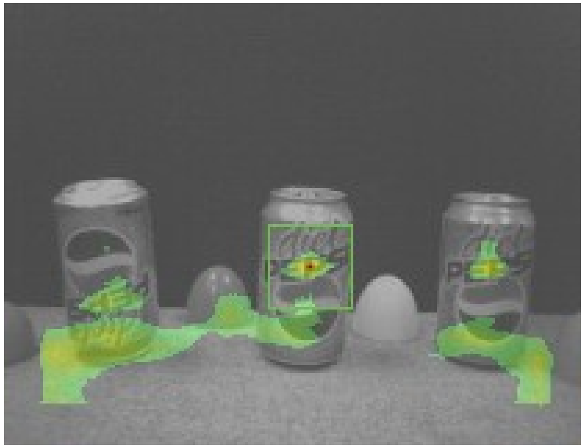
Original



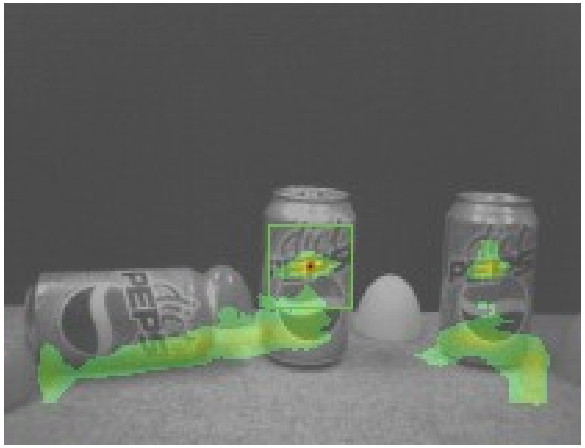
Occluded



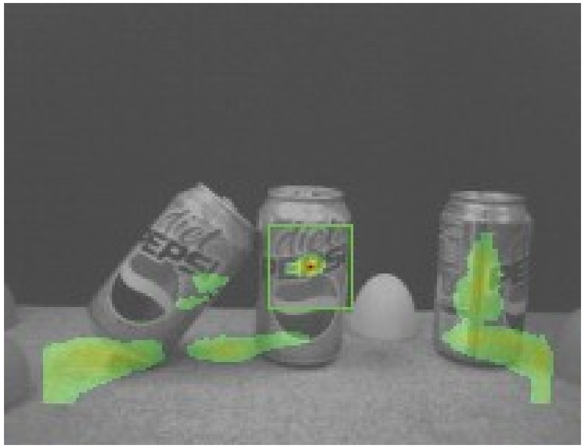
Rotated



Flipped



Sideways



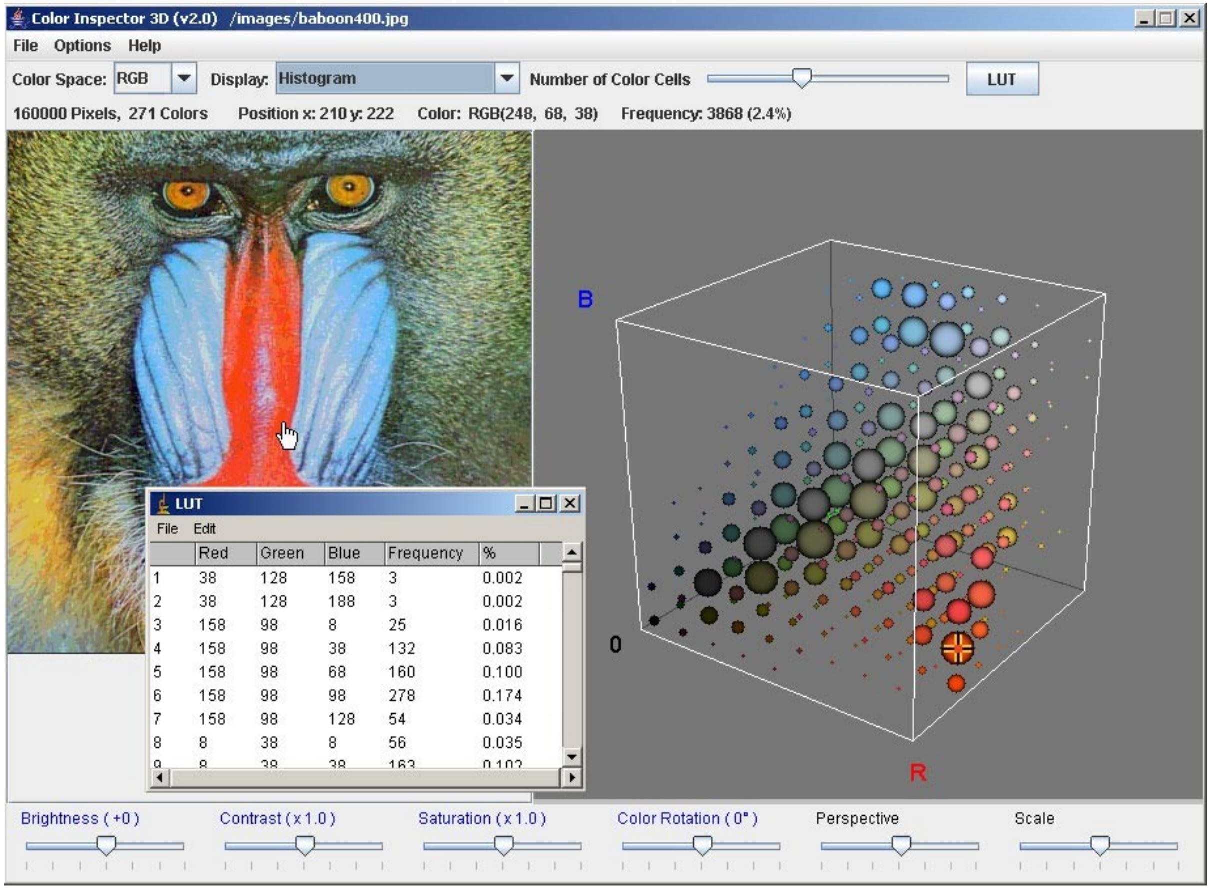
Diagonal

Color Histograms (Swain)

- Invariant to translation, 2D rotation, and scale.

- Handles some occlusion.

- But assumes object has already been segmented.



Color Inspector 3D (v2.0) /images/baboon400.jpg

Color Space: RGB Display: Histogram

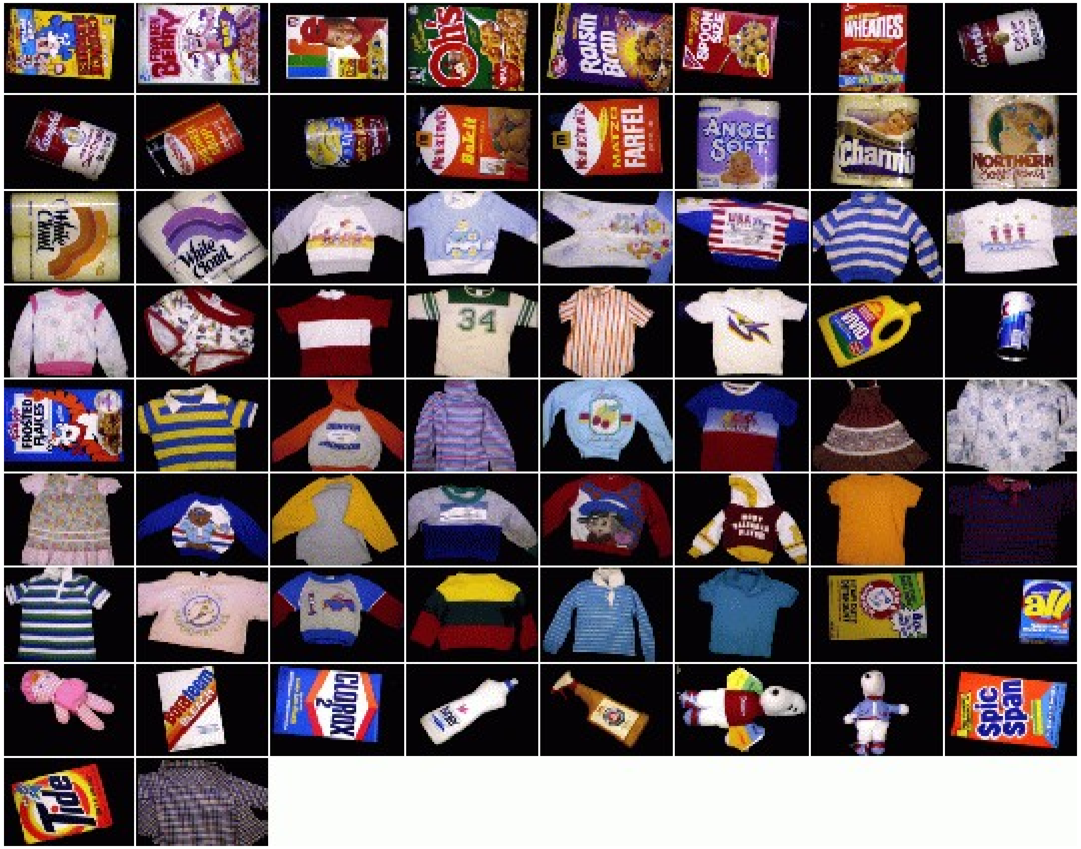
160000 Pixels, 271 Colors Position x: 210 y: 222 Color: RGB(248, 68, 38) Frequency: 3868 (2.4%)

LUT

	Red	Green	Blue	Frequency	%
1	38	128	158	3	0.002
2	38	128	188	3	0.002
3	158	98	8	25	0.016
4	158	98	38	132	0.083
5	158	98	68	160	0.100
6	158	98	98	278	0.174
7	158	98	128	54	0.034
8	8	38	8	56	0.035
9	8	38	38	163	0.102

Brightness (+0) Contrast (x 1.0) Saturation (x 1.0) Color Rotation (0°)

Object Classes

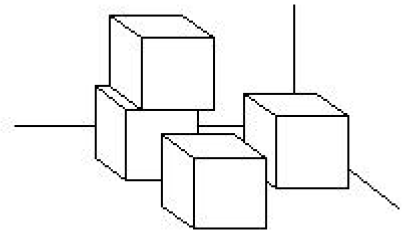


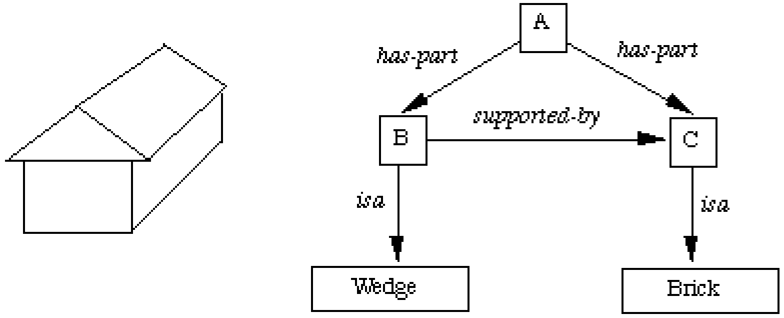
Test Images

Figure from M. A. Stricker,
http://www.cs.uchicago.edu/files/tr_authentic/TR-92-22.ps

Blocks World Vision

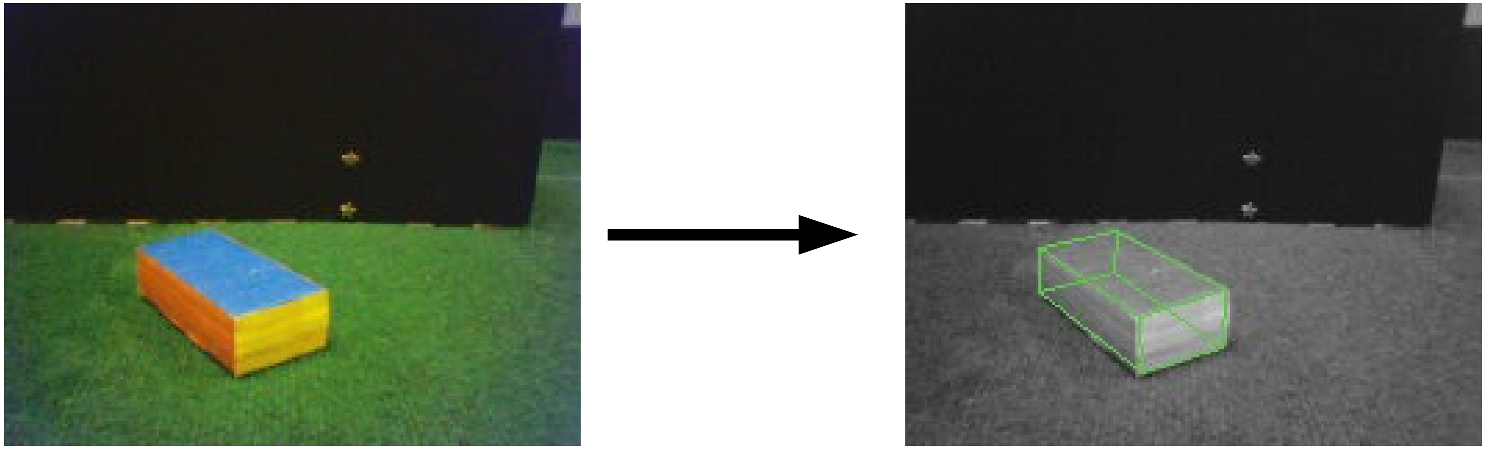
- One of the earliest computer vision domains.
  - Roberts (1965) used line drawings of block scenes: the first "computer vision" program.
- Simplified problem because shapes were regular.
  - Occlusions could be handled.
- Still a hard problem. No standard blocks world vision package exists.



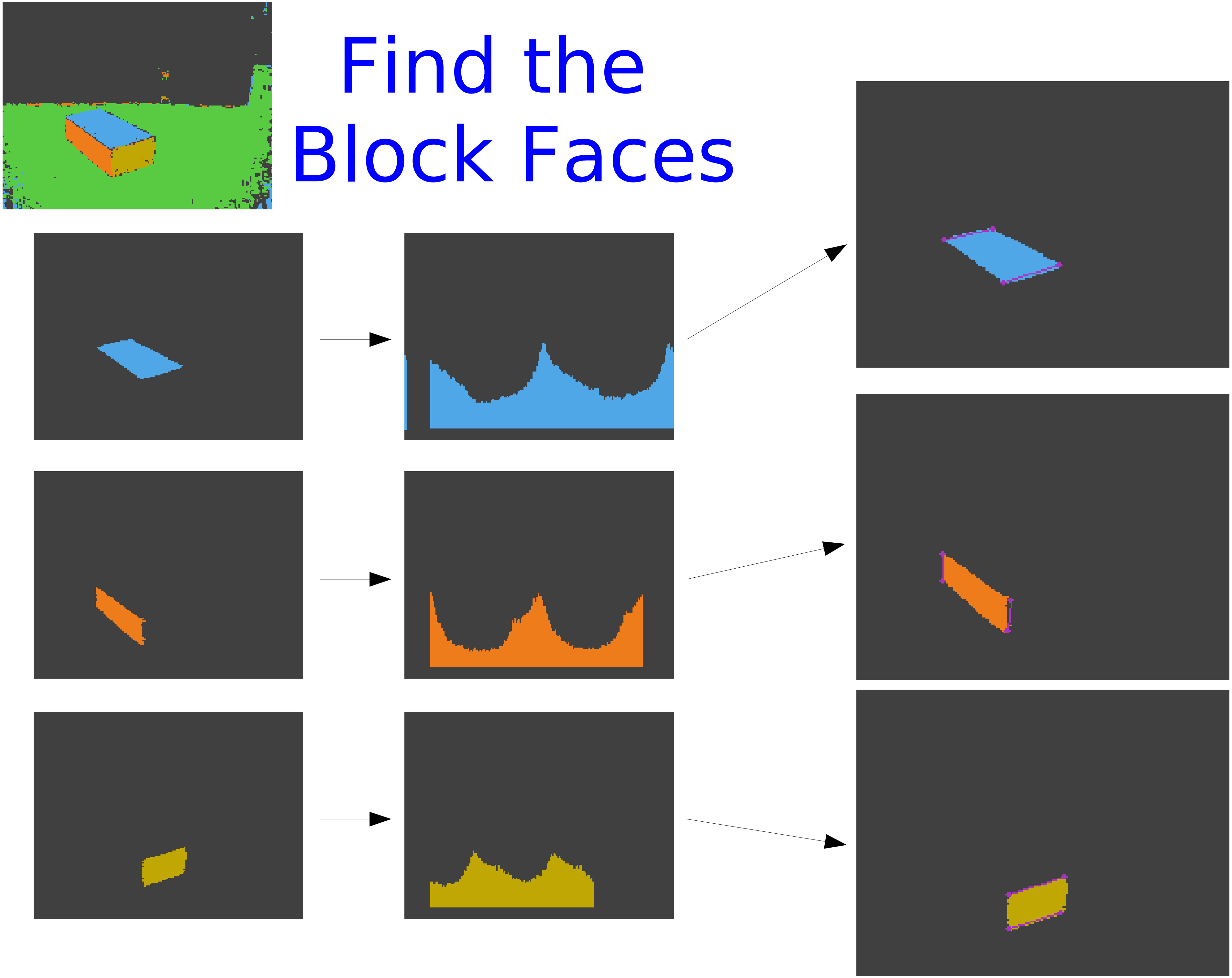


AIBO Blocks World

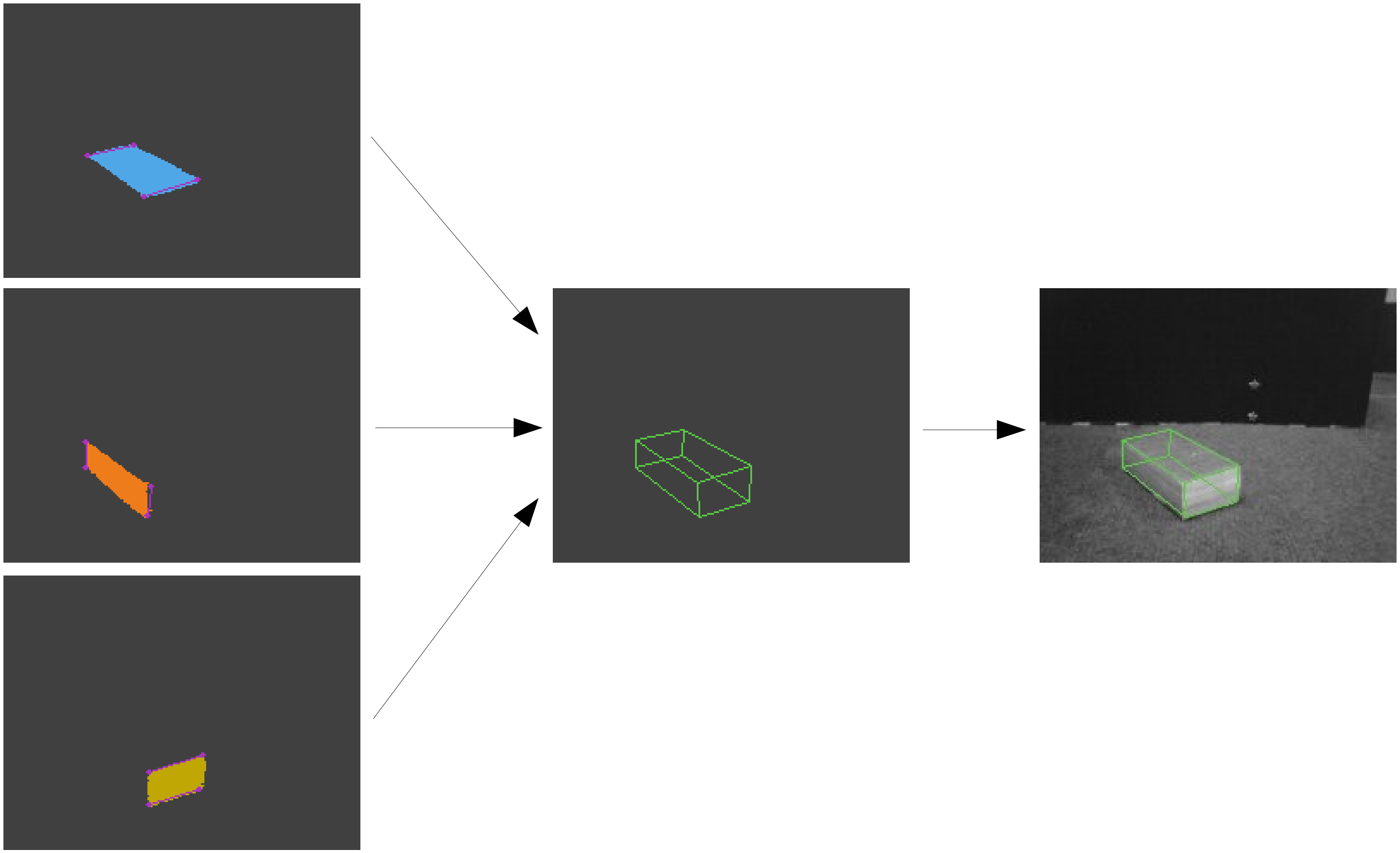
- Matt Carson's senior thesis (CMU CSD, 2006).
- Goal: recover positions, orientations, and sizes of blocks.



Find the Block Faces

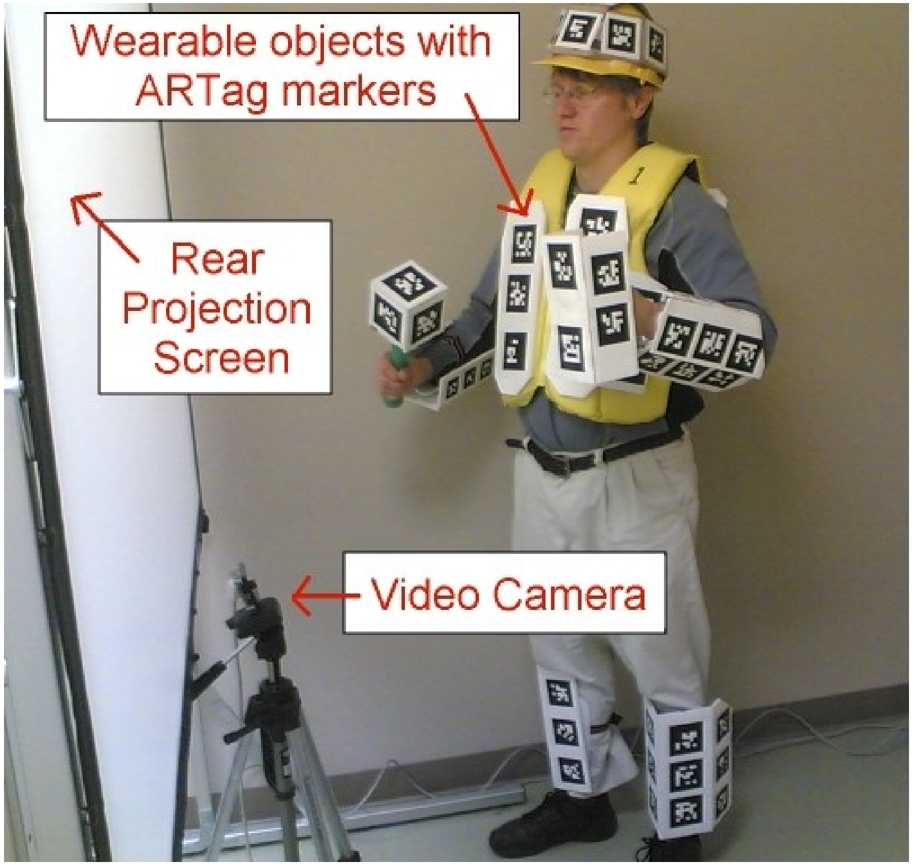


Find the Block From the Faces



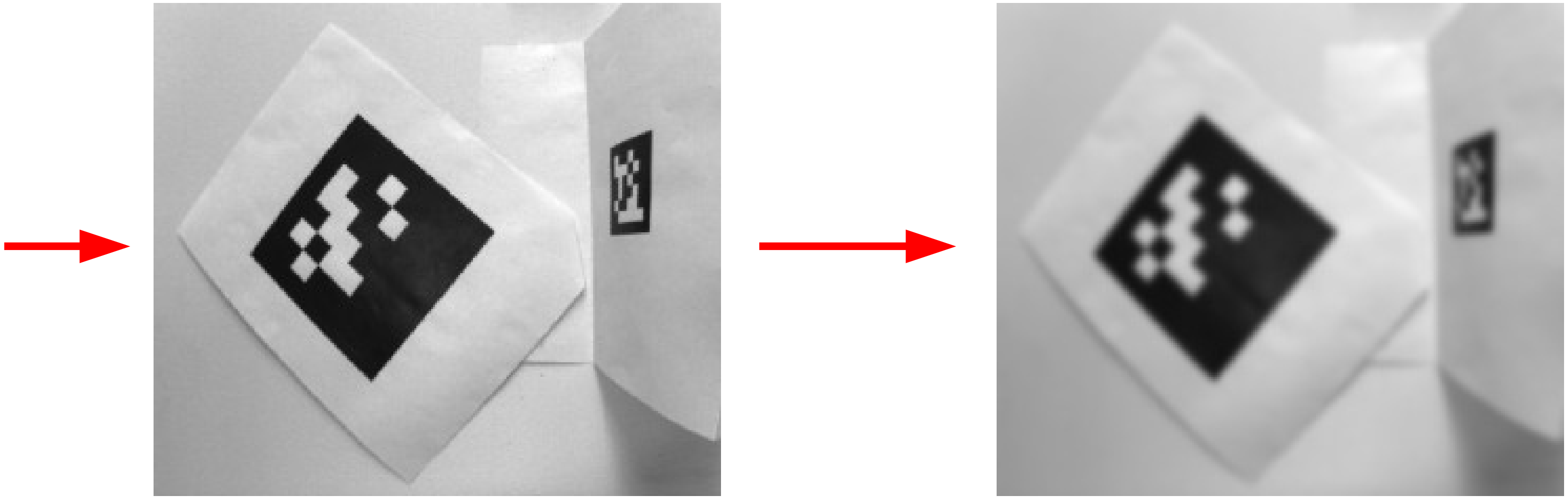
AprilTags

- Robust fiducial markers created by Edwin Olson at the University of Michigan.
- Inspired by ARTag (Fiala) and ARToolkit.

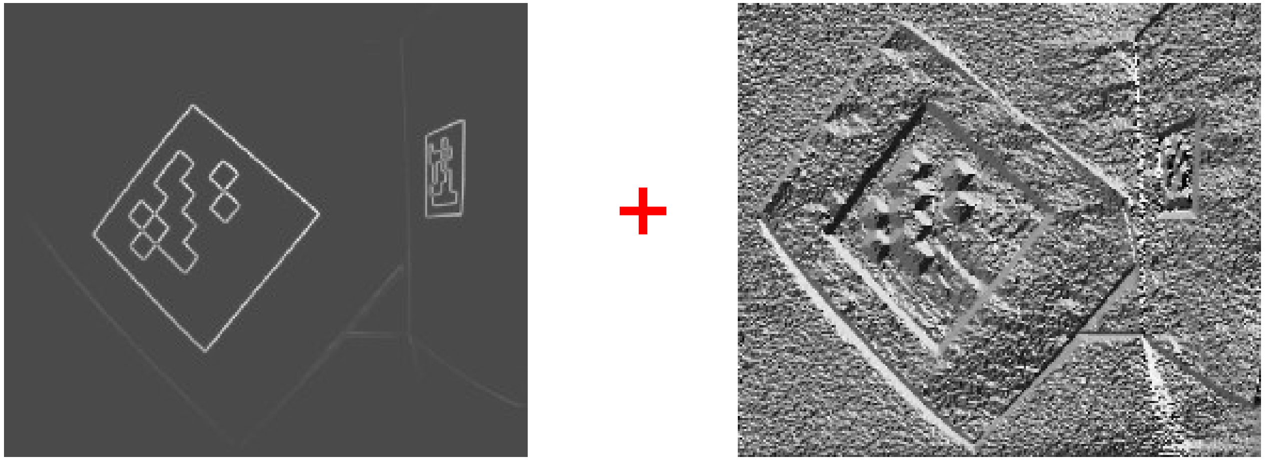


How AprilTags Work (1/4)

1. Convert to greyscale and apply a Gaussian blur.

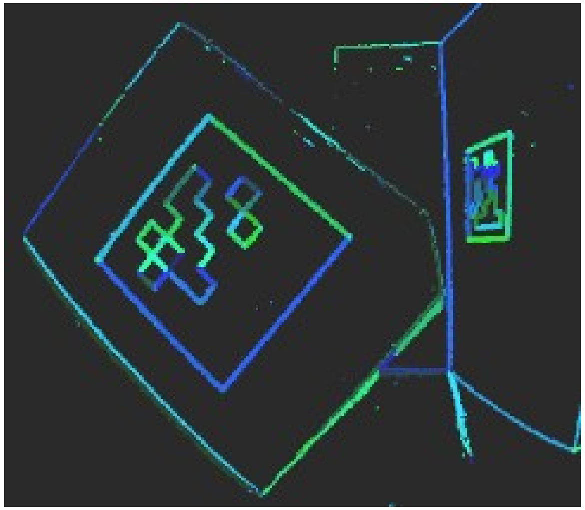


2. Compute gradient at each pixel: magnitude + direction.

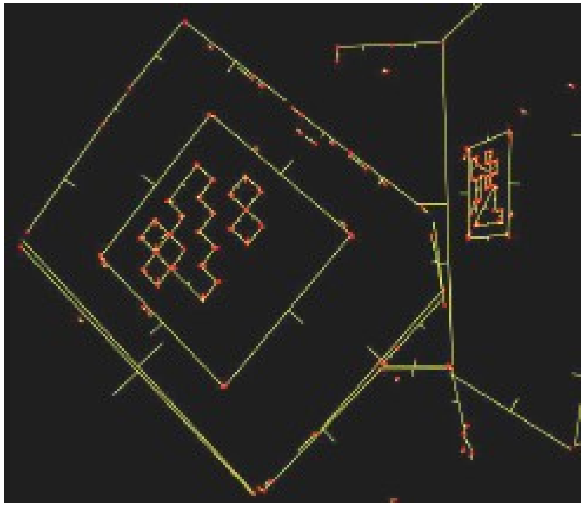


How AprilTags Work (2/4)

3. Generate a list of two-pixel “Edges”.
4. Group aligned edges into Clusters; color indicates gradient direction.

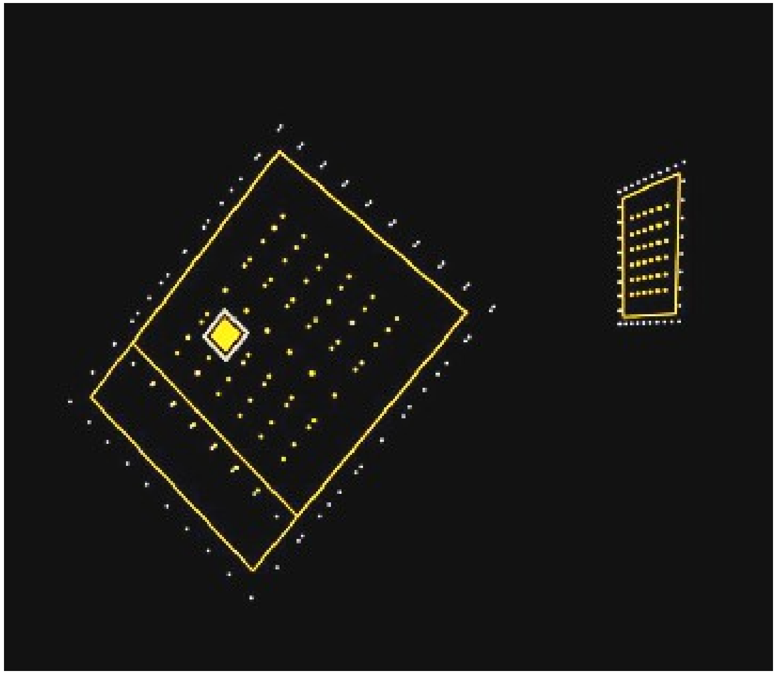


5. Fit lines to the clusters, forming Segments. Notch points toward the bright side of the line.



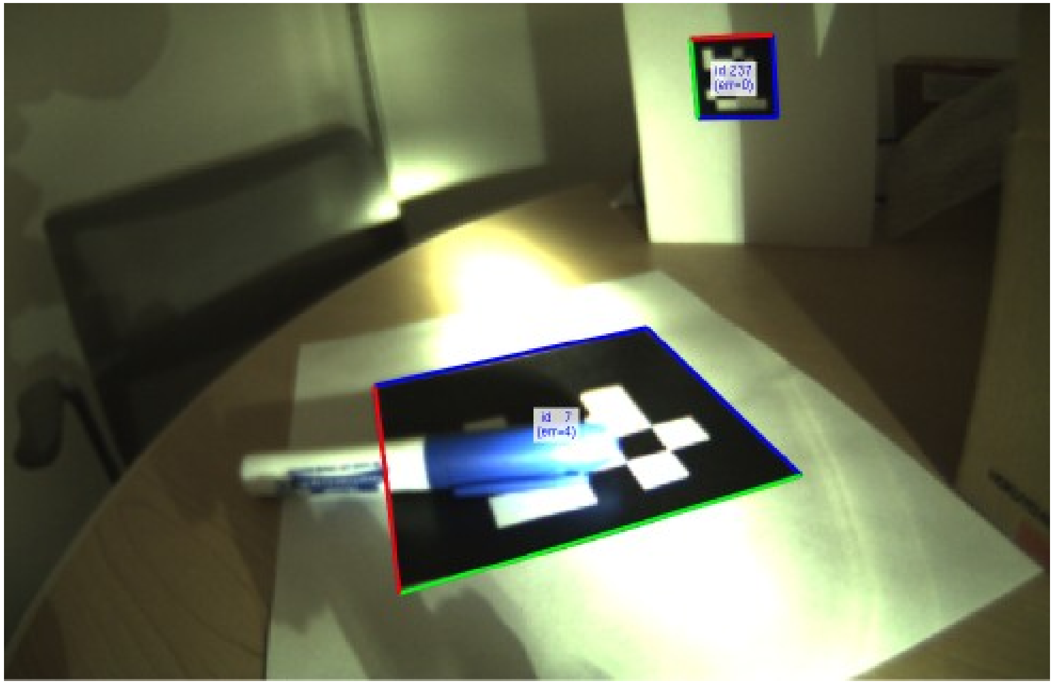
How AprilTags Work (3/4)

6. For each Segment, find others that begin where this segment ends.
7. Find loops of length 4, called Quads.



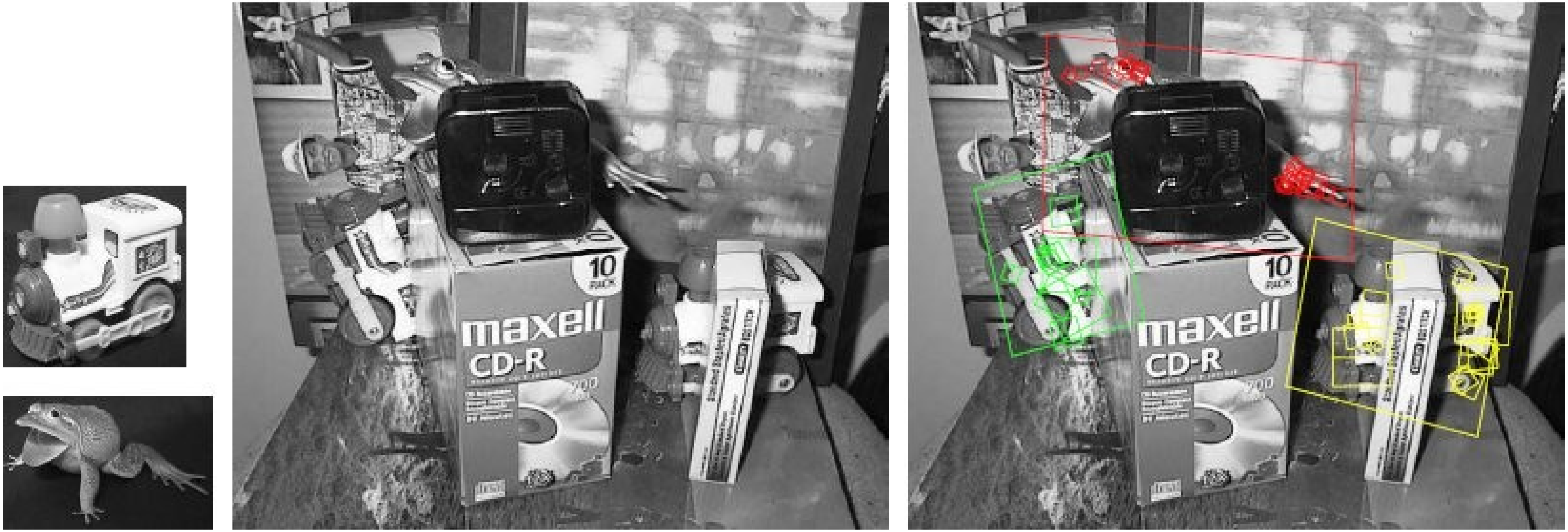
How AprilTags Work (4/4)

8. Decode the Quads by looking at the pixels inside the border to see if they represent a valid tag code.
9. Search for overlapping tag detections and keep only the best ones (lowest Hamming distance or largest perimeter.)



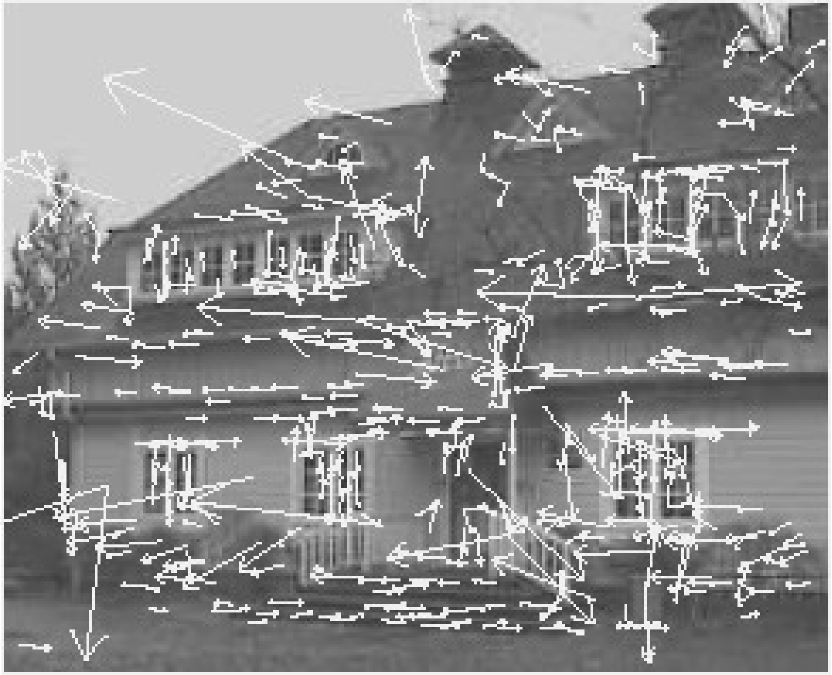
SIFT (Lowe, 2004)

- Scale-Invariant Feature Transform
- Can recognize objects independent of scale, translation, rotation, or occlusion.
- Can segment cluttered scenes.
- Slow training, but fast recognition.



How Does SIFT Work?

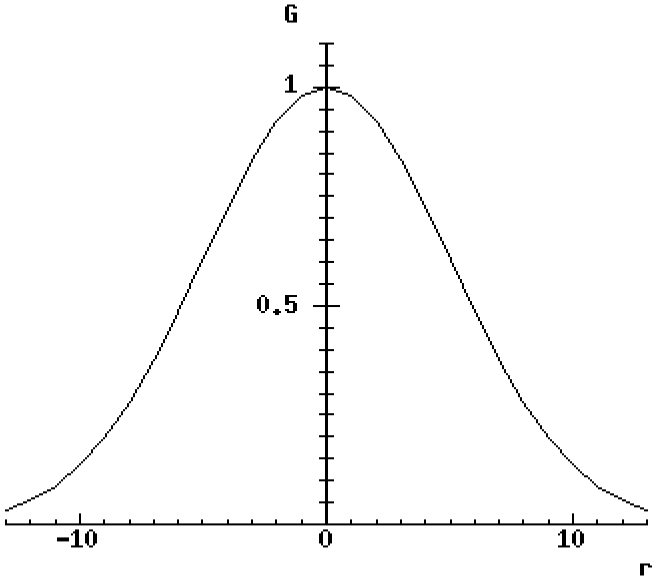
- Generate large numbers of features that densely cover each training object at various scales and orientations.
- A 500 x 500 pixel image may generate 2000 stable features.
- Store these features in a library.
- For recognition, find clusters of features present in the image that agree on the object position, orientation, and scale.

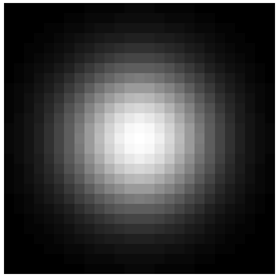


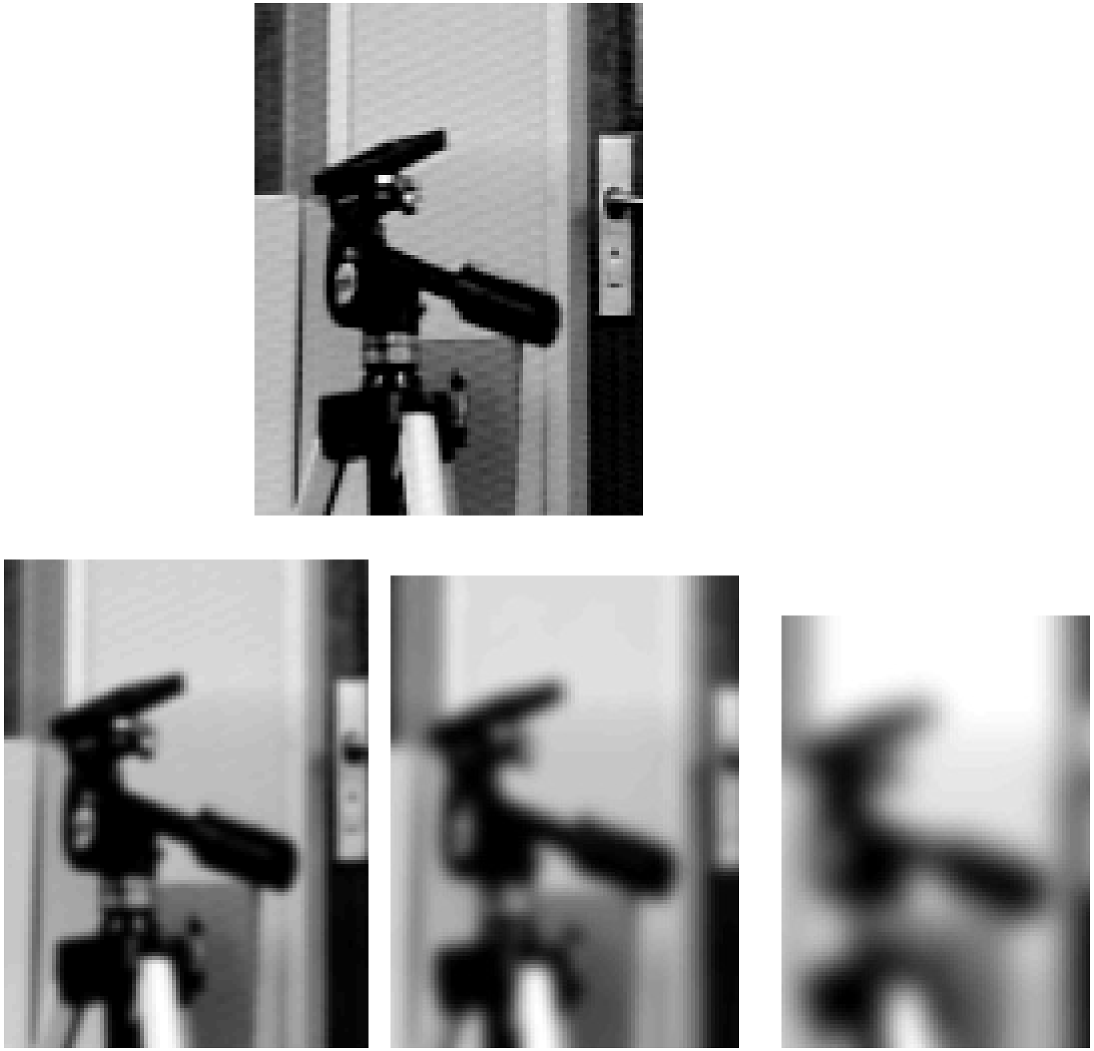
SIFT Feature Generation

1) Scale-space extrema detection
   - Use differences of Gaussians to find potential interest points.
2) Keypoint localization
   - Fit detailed model to determine location and scale.
3) Orientation assignment
   - Assign orientations based on local image gradients.
4) Keypoint descriptor
   - Extract description of local gradients at selected scale.

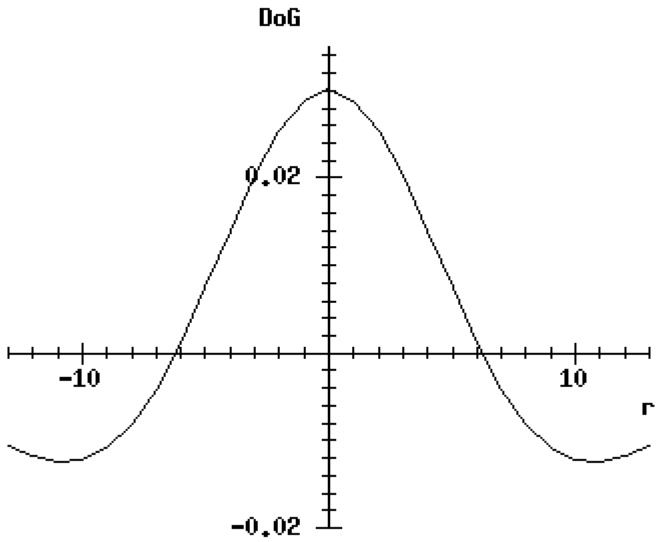
Gaussian Smoothing

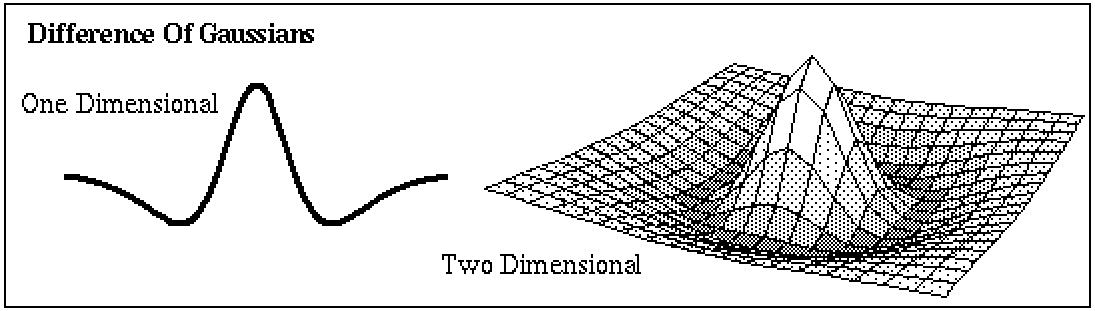


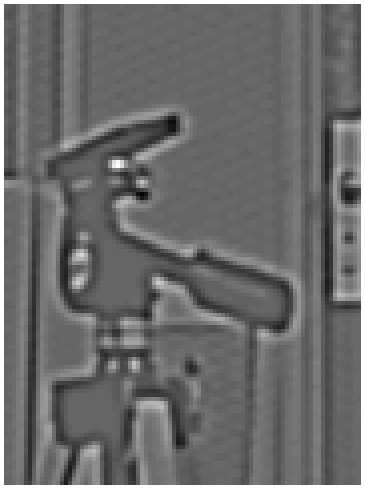




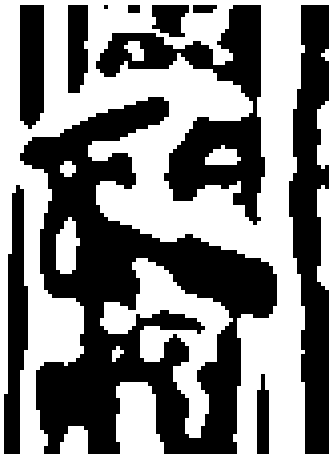
Difference of Gaussians: Edge Detection





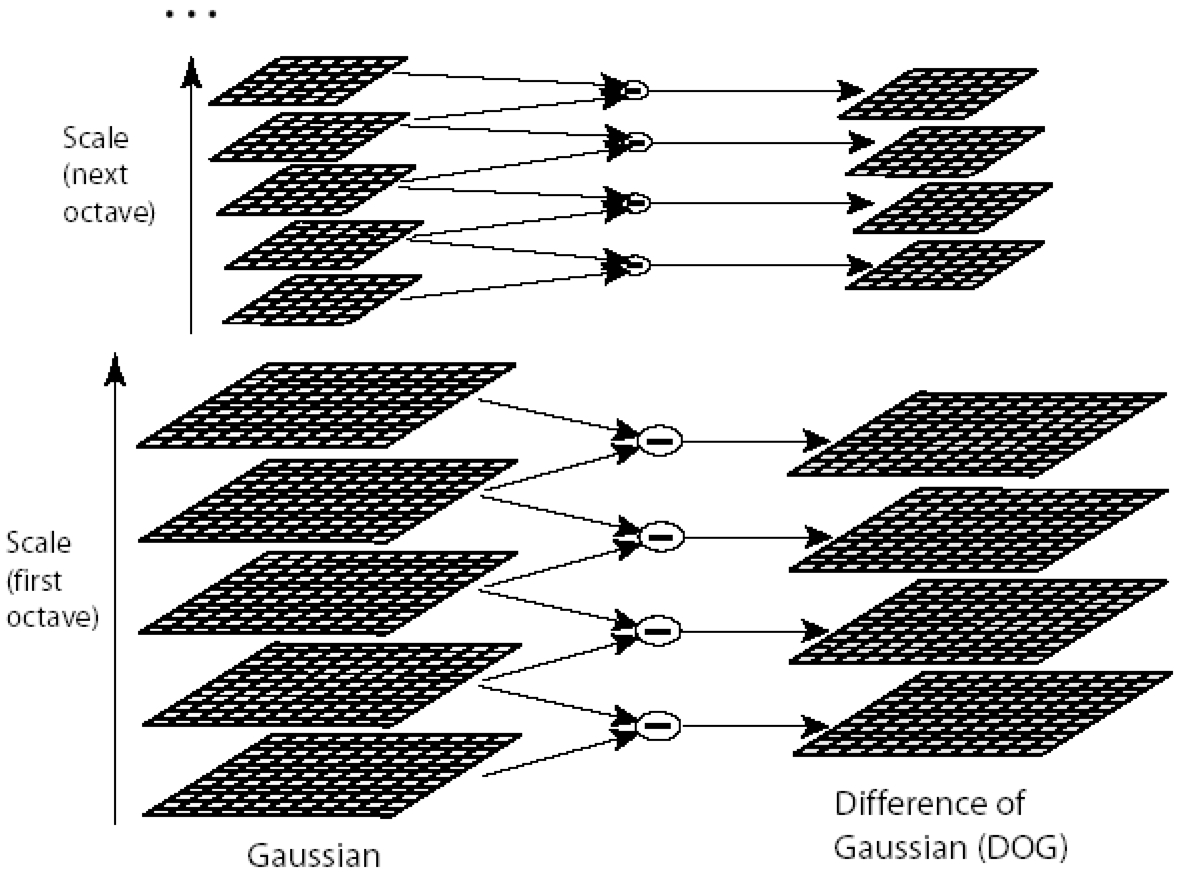


Difference of Gaussians



Zero Crossings = Edges

Scale Space


. . .
Scale
(next
octave)
Scale
(first
octave)
Gaussian
Difference of
Gaussian (DOG)

Scale Space Extrema

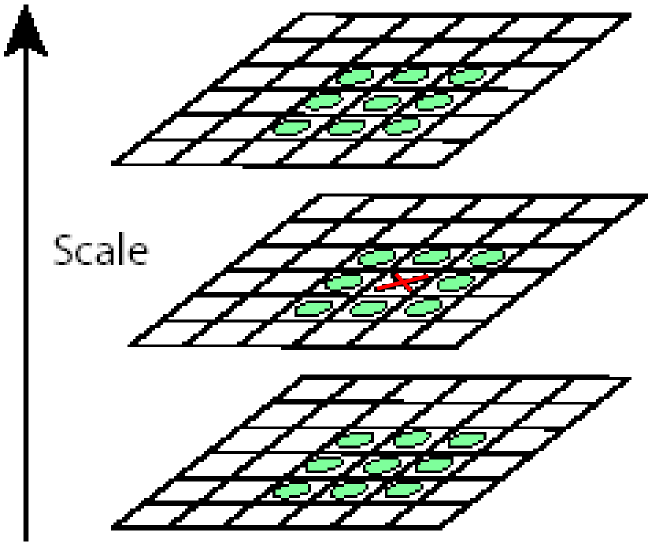


Figure 2: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3x3 regions at the current and adjacent scales (marked with circles).

Filtering the Features

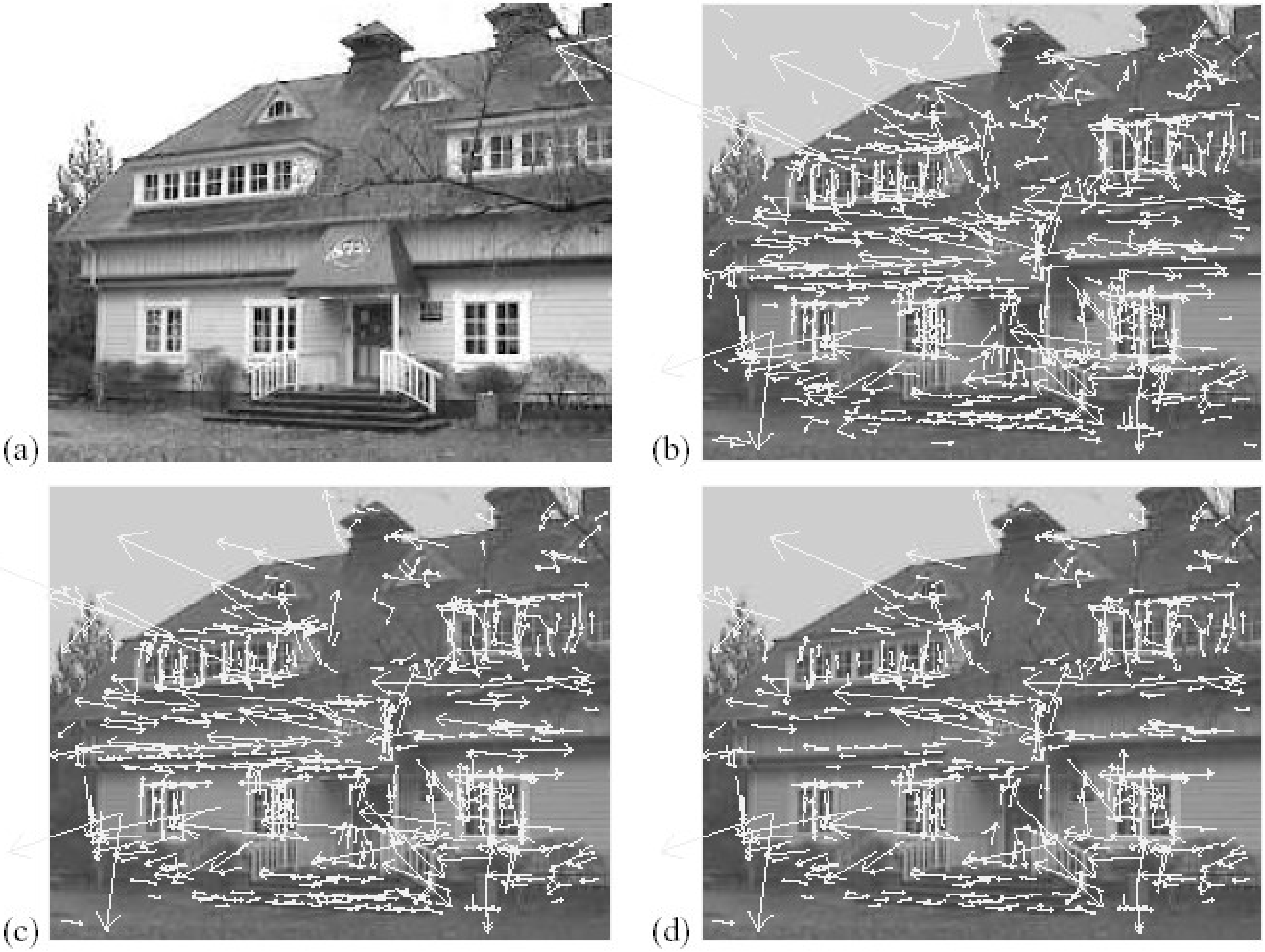


Figure 5: This figure shows the stages of keypoint selection. (a) The 233x189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

Keypoint Descriptors

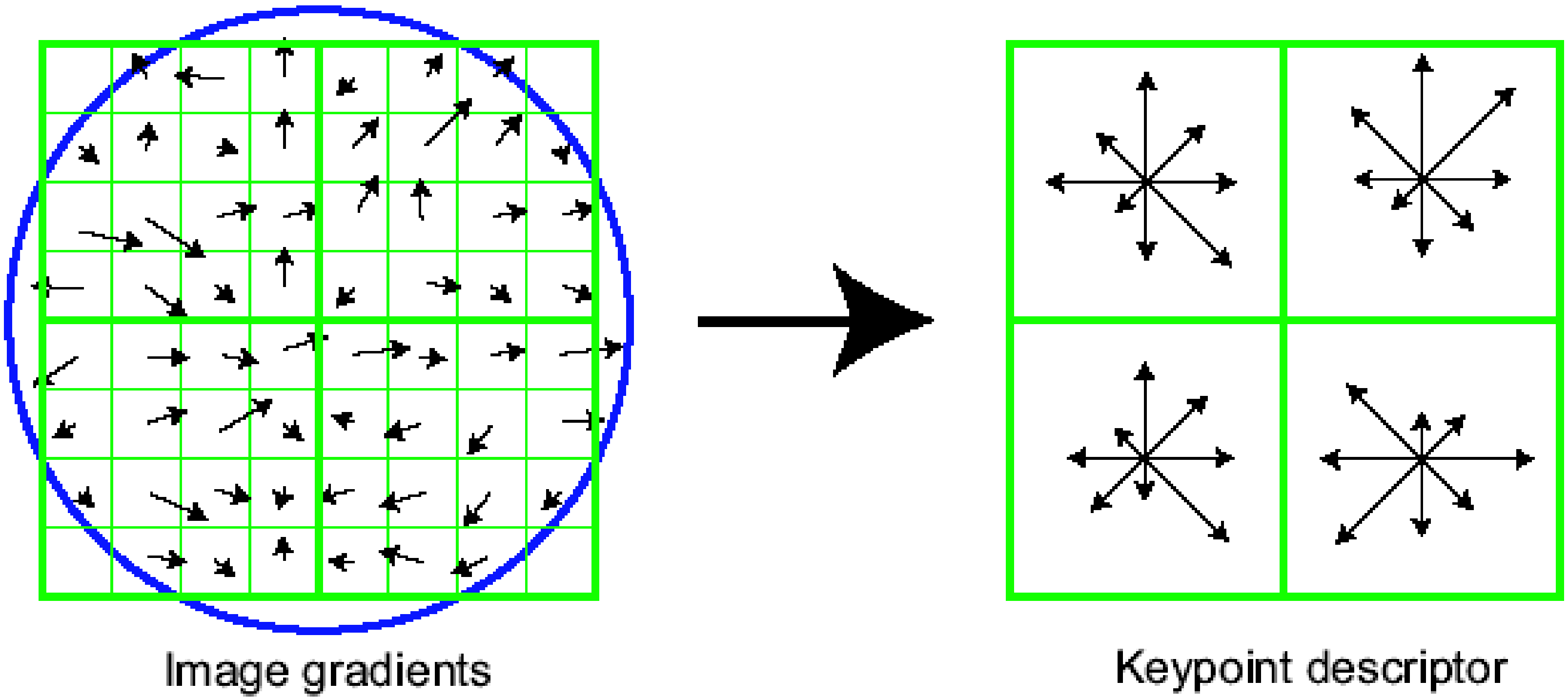


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

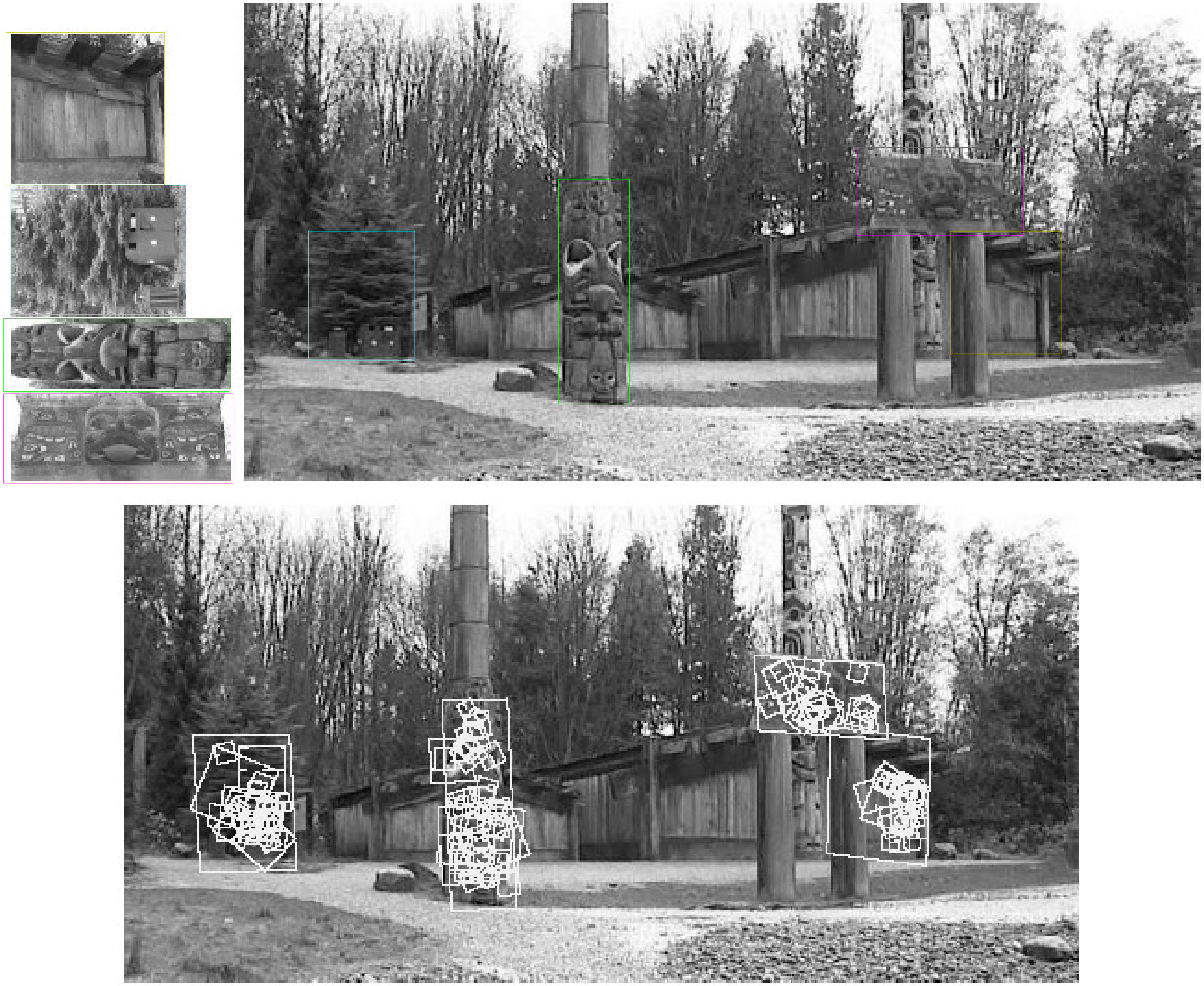
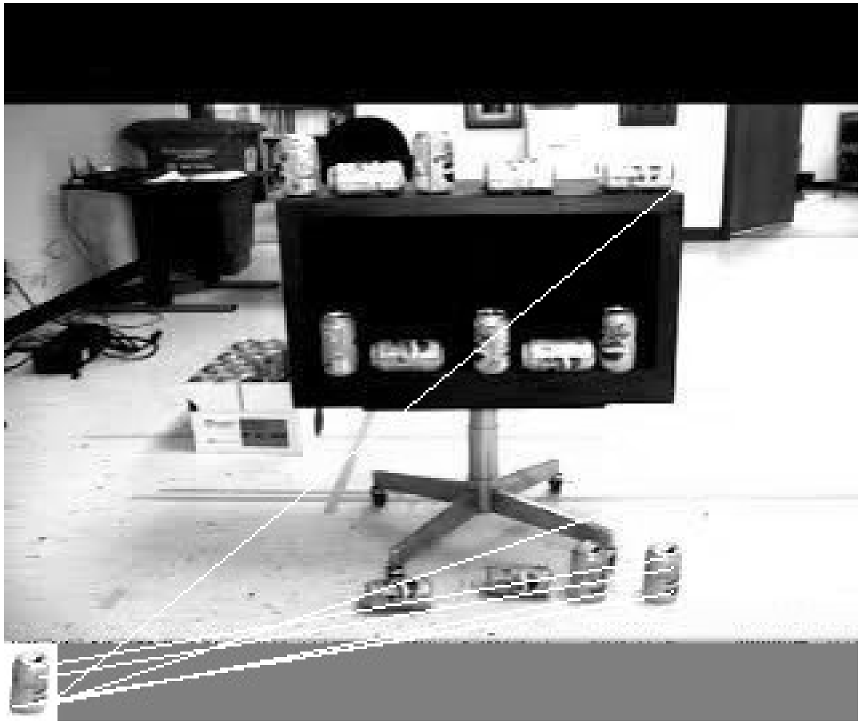


Figure 13: This example shows location recognition within a complex scene. The training images for locations are shown at the upper left and the 640x315 pixel test image taken from a different viewpoint is on the upper right. The recognized regions are shown on the lower image, with keypoints shown as squares and an outer parallelogram showing the boundaries of the training images under the affine transform used for recognition.

Real-Time SIFT Example

Fred Birchmore used SIFT to recognize soda cans.

http://eyecanseecan.blogspot.com

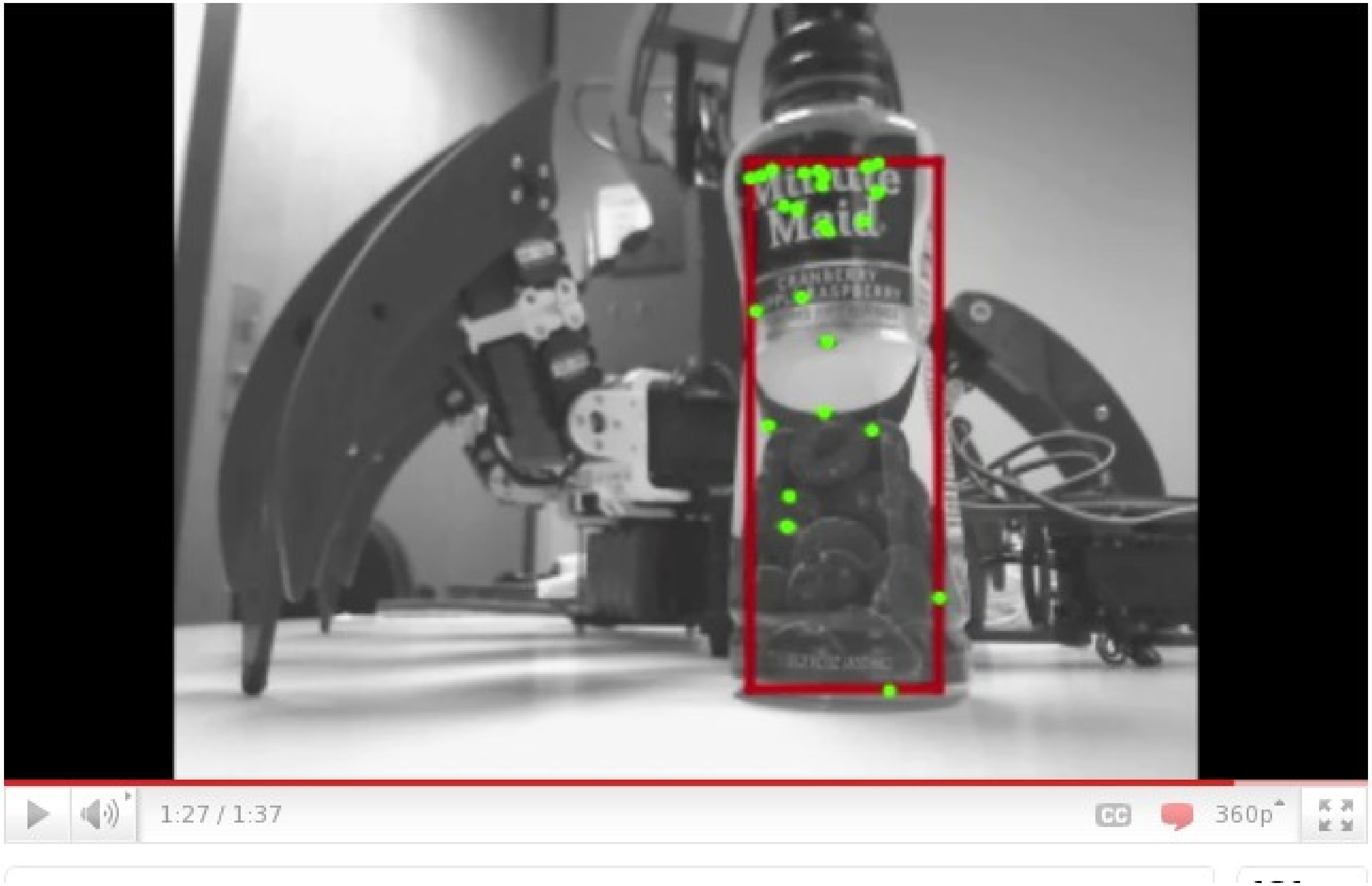


See demo videos on his blog.

SIFT in Tekkotsu

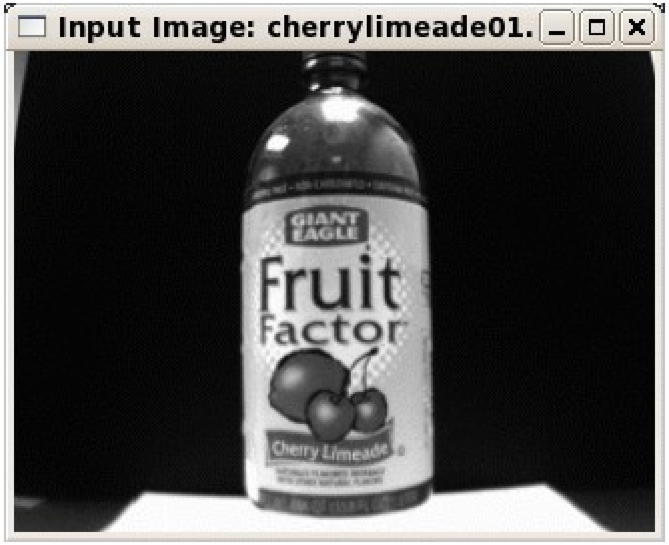
- Xinghao Pan implemented a SIFT tool for Tekkotsu:
  - Allow users to construct libraries of objects
  - Each object has a collection of representative images
  - User can control which SIFT features to use for matching
  - Java GUI provides for easy management of the library
- How to integrate SIFT with the dual coding system?
  - Object scale can be used to estimate distance
  - Match in camera space must be converted to local space

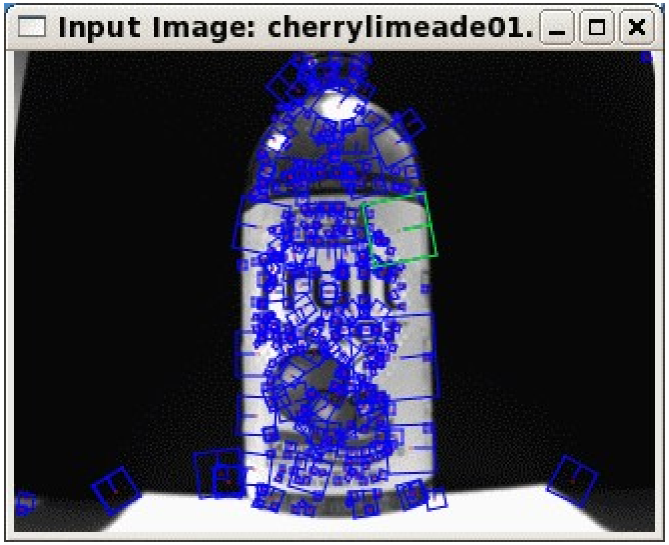
Tekkotsu SIFT Video

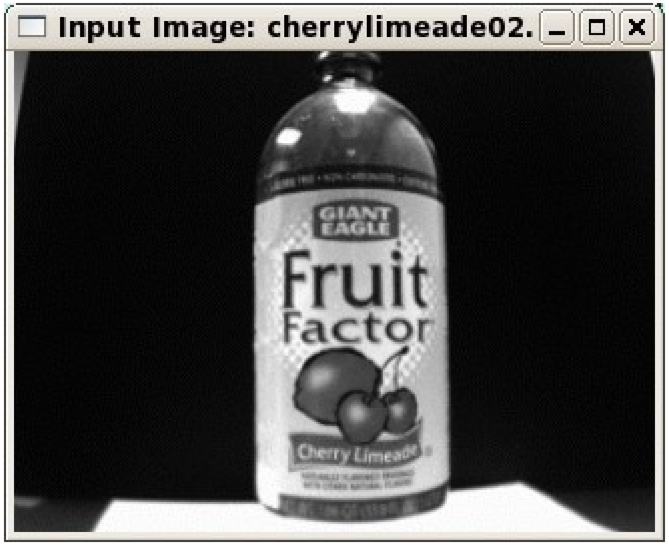


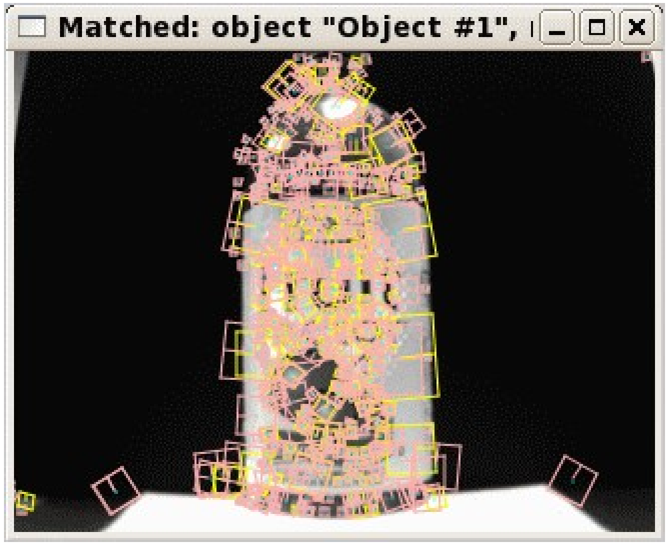
http://www.youtube.com/watch?v=2QVSTtjenCs

SIFT Tool

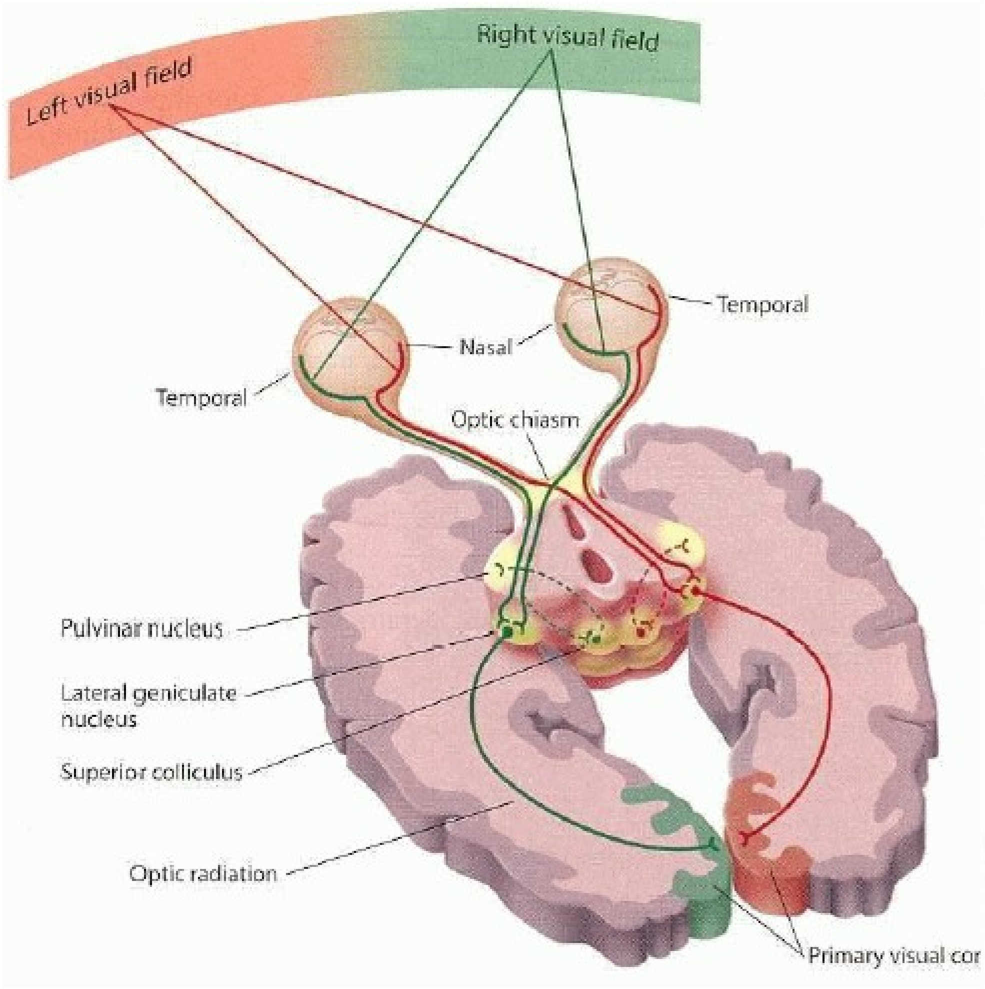

Input Image: cherrylimeade01.
GIANT EAGLE
Fruit Factor
Cherry Limeade

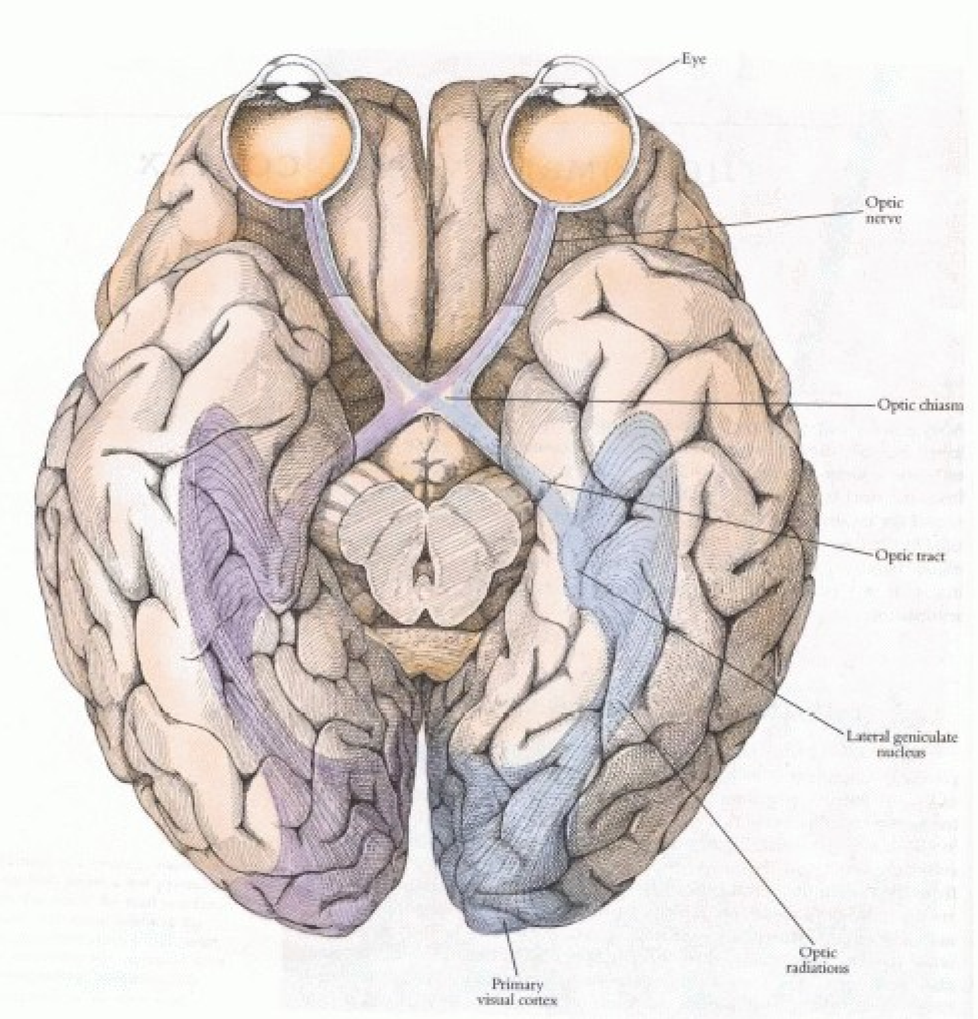

Input Image: cherrylimeade01.


Input Image: cherrylimeade02.
GIANT EAGLE
Fruit Factor
Cherry Limeade


Matched: object "Object #1",

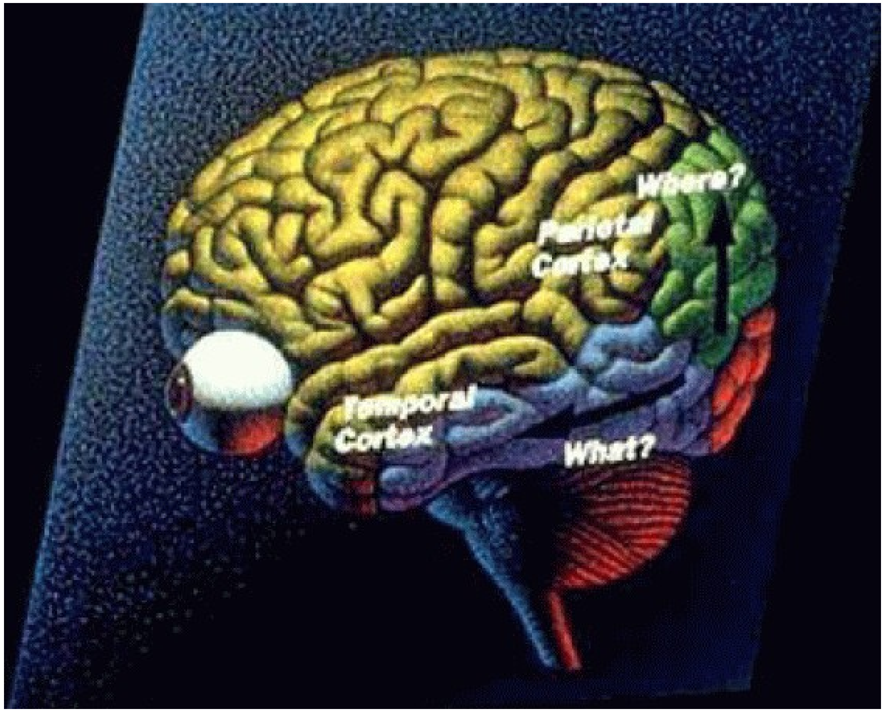
Object Recognition in the Brain

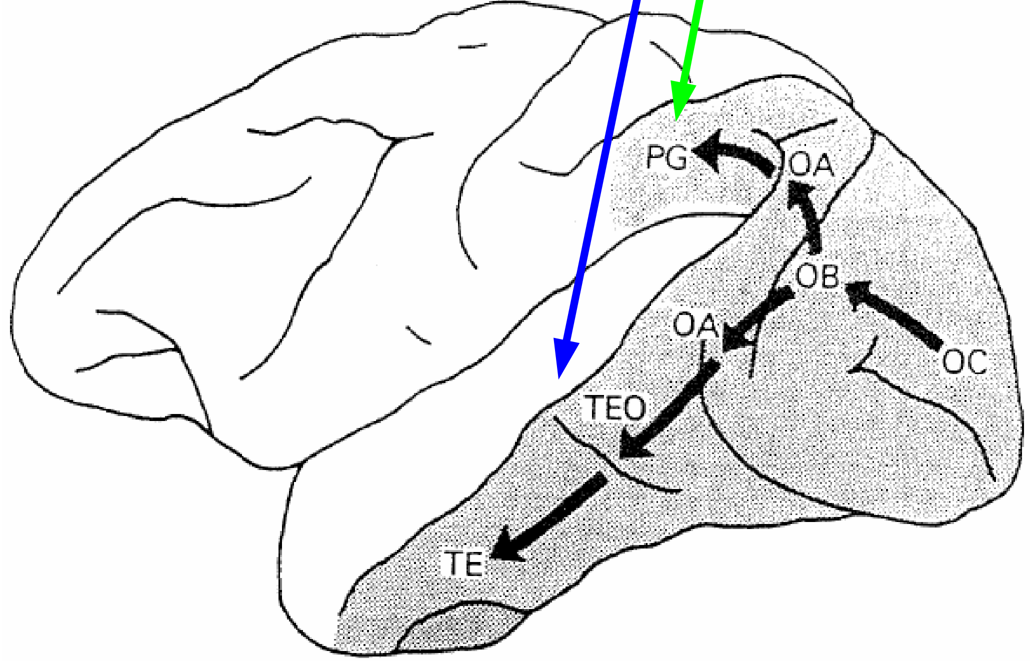




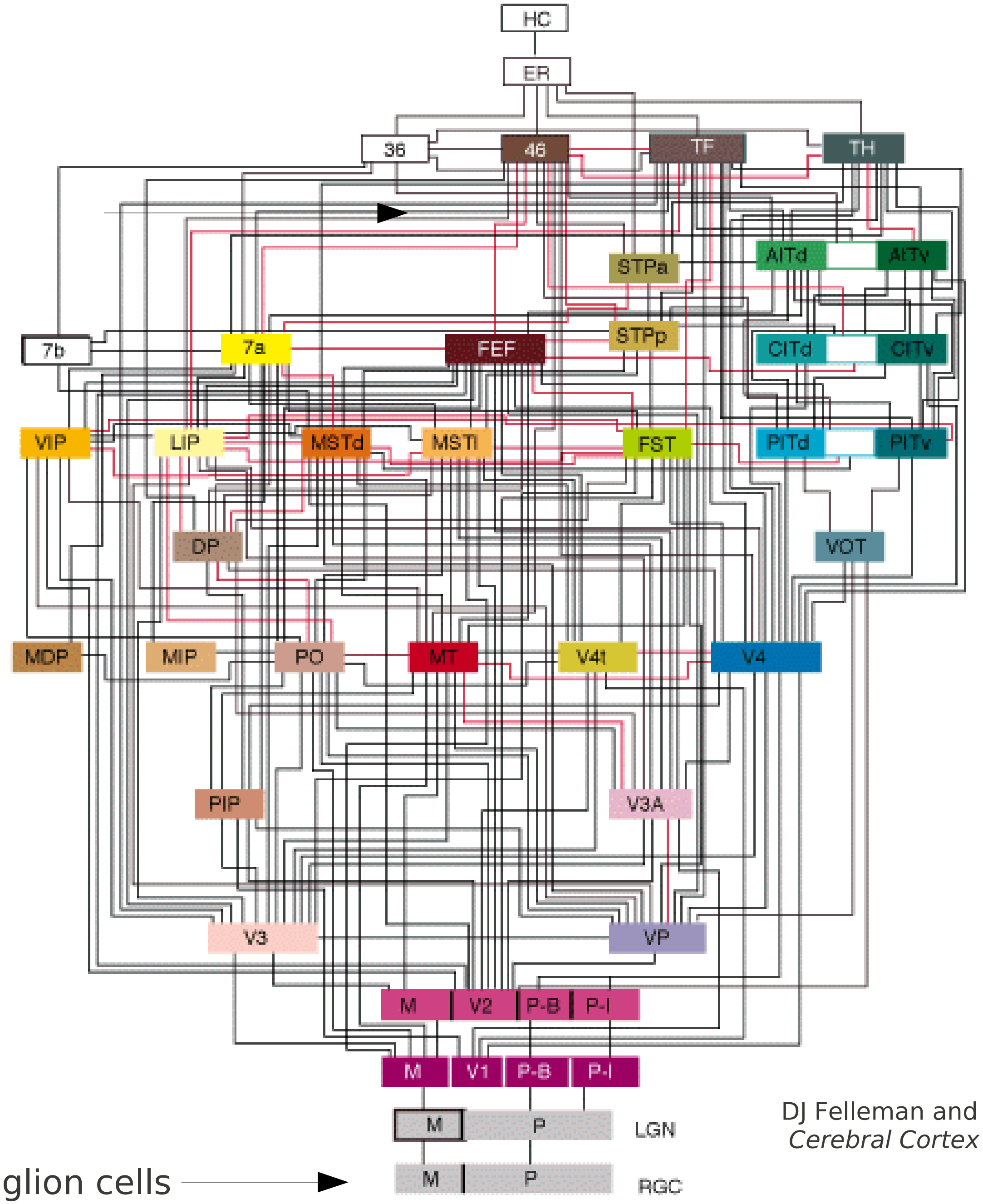
Object Recognition in the Brain

- Mishkin & Ungerleider: dual visual pathways.
  - The dorsal, “where” pathway lies in parietal cortex.
  - The ventral, “what” pathway lies in temporal cortex.
  - Lesions to these areas yield very specific effects.





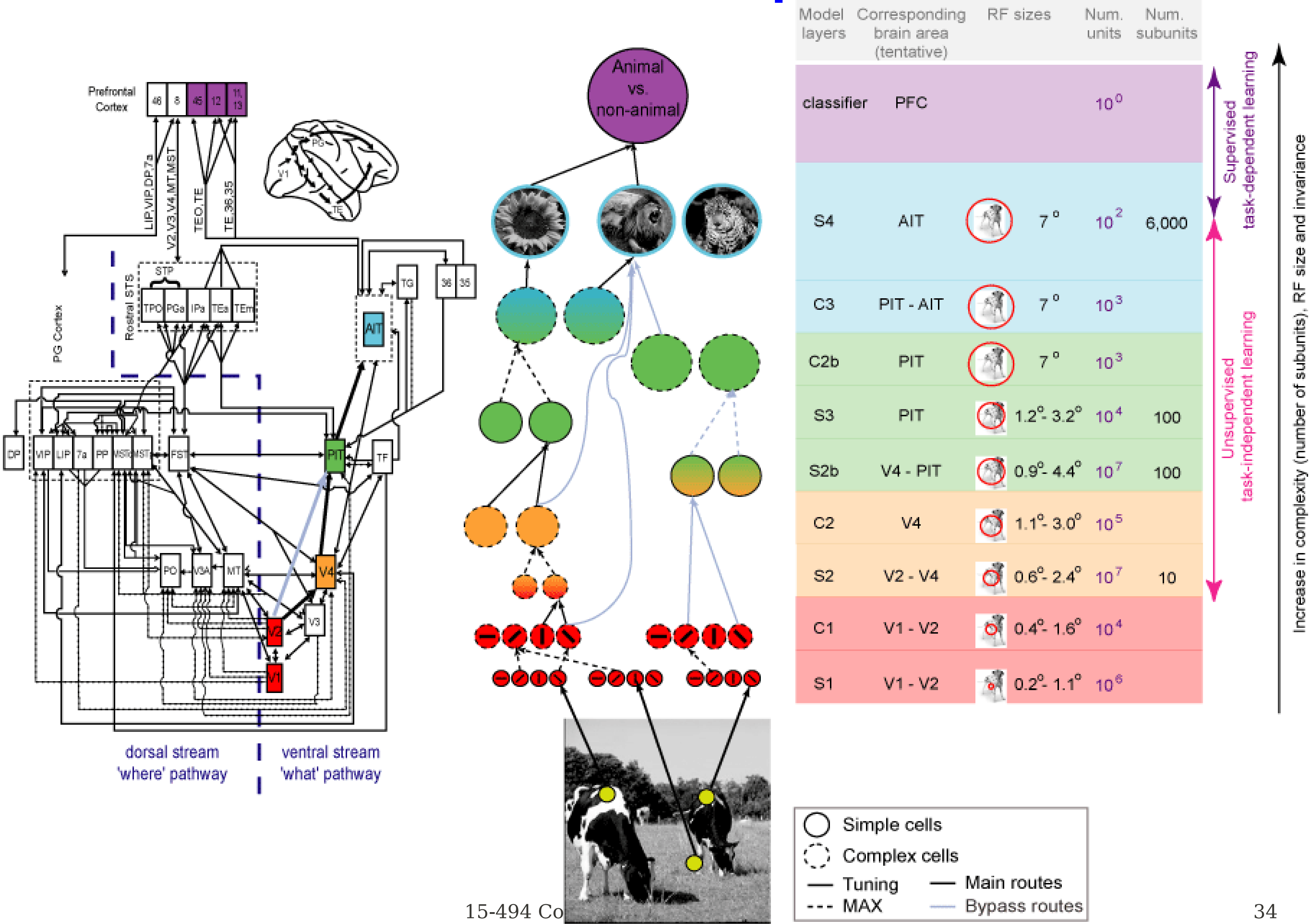
The Macaque "Vision Pipeline"



RGC = retinal ganglion cells

DJ Felleman and DC Van Essen (1991),
Cerebral Cortex **1:**1-47.

Serre & Poggio (PAMI 2007): Model Based on Temporal Cortex



Model layers	Corresponding brain area (tentative)	RF sizes	Num. units	Num. subunits
classifier	PFC		10^0	
S4	AIT	7 °	10^2	6,000
C3	PIT - AIT	7 °	10^3	
C2b	PIT	7 °	10^3	
S3	PIT	1.2°- 3.2°	10^4	100
S2b	V4 - PIT	0.9°- 4.4°	10^7	100
C2	V4	1.1°- 3.0°	10^5	
S2	V2 - V4	0.6°- 2.4°	10^7	10
C1	V1 - V2	0.4°- 1.6°	10^4	
S1	V1 - V2	0.2°- 1.1°	10^6	

To Learn More About Computer and Biological Vision

- Take Tai Sing Lee's Computer Vision class, 15-385.
- Take Tai Sing Lee's Computational Neuroscience class, 15-490.
- There are many books on this subject. One of the classics is “Vision” by David Marr.