

# Robot Vision

# So far ...

- Following our evolutionary development of robots
  - Behaviour-based robots
    - Simple robots (like insects) that do not need world models
  - SLAM and episodic memory
    - Robots with simple sensory systems that create and use world models

# This time ...

- Adding more complex sensors that are need for more complex tasks

# Robot Vision

- The aim of robot vision is to transform radiation reflected from objects into an internal representation of the objects, appropriate for the robot's task
- Three steps are involved:
  1. Image Formation
  2. Image Analysis
  3. Understanding

# The State of Computer Vision

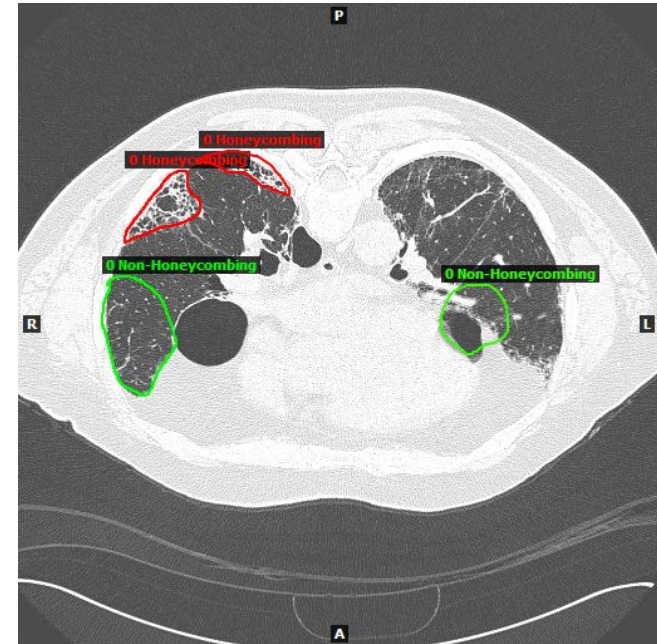
- The general computer vision problem is unsolved
  - Develop a visual system as good as humans
  - No progress for 40 years
- A lot of progress in specific computer vision problems
  - e.g. face recognition used in
    - digital cameras
    - surveillance
    - security
  - e.g. Inspection in Automation
    - e.g. pick and place



# The General Vision Problem

- Maybe there's no such thing
- Maybe our vision is an accumulation of lots of specialised vision systems

# Robot Vision and Image Processing



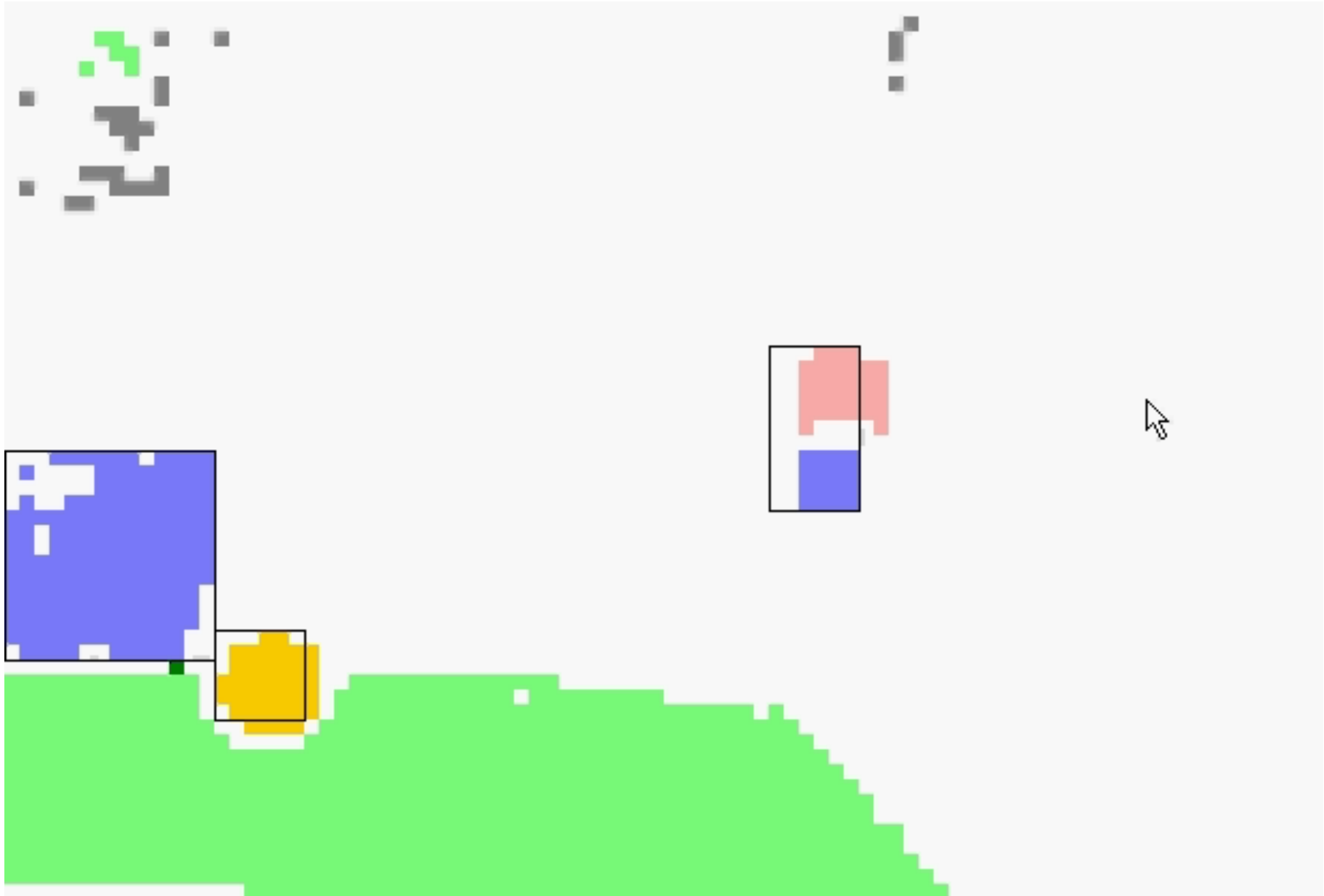
- Robot Vision is in an embedded system and ultimately should work in real time
- Image Processing can be off-line and is not time critical

# Doggie Cam (1)

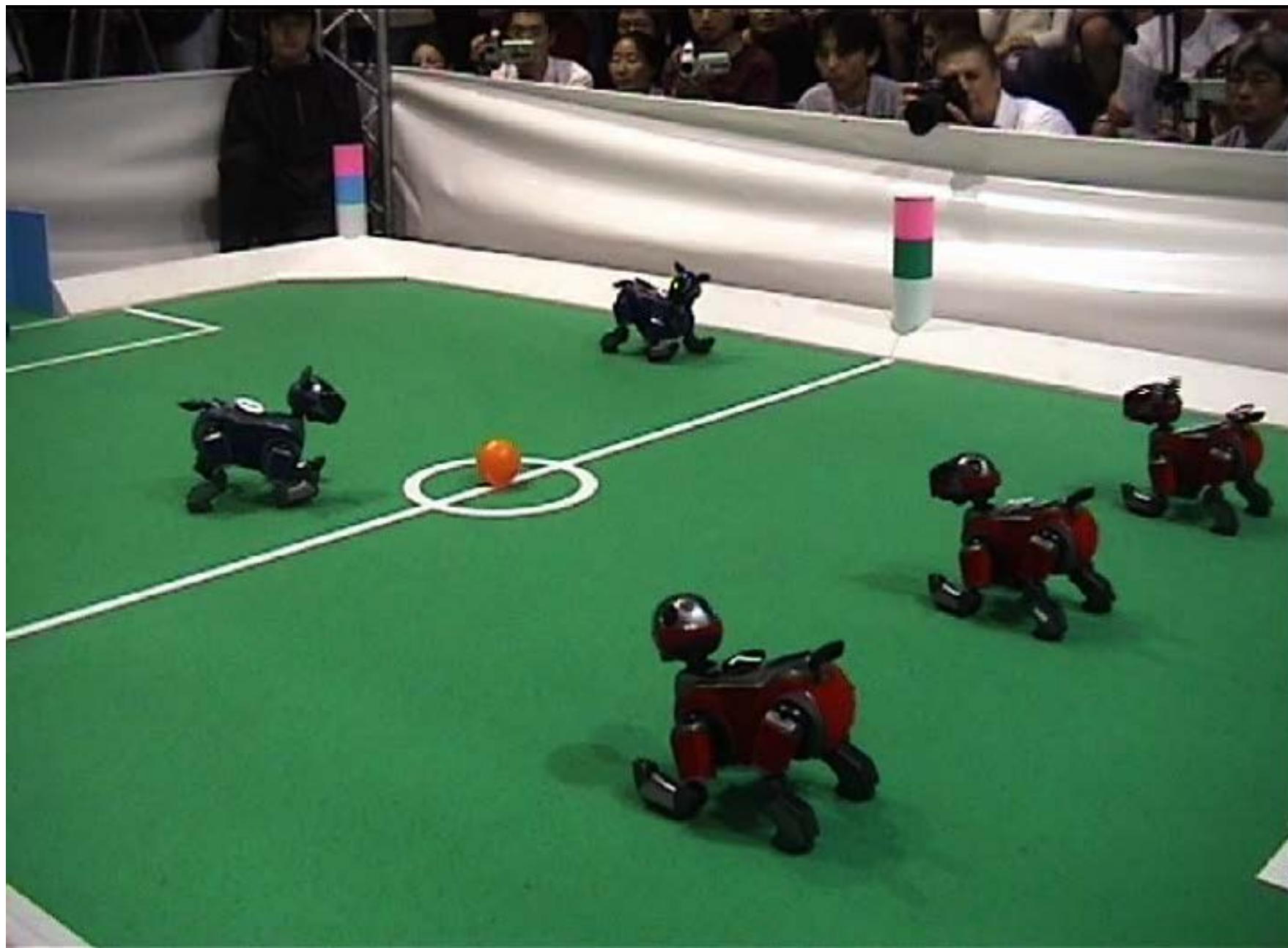




# Object Recognition



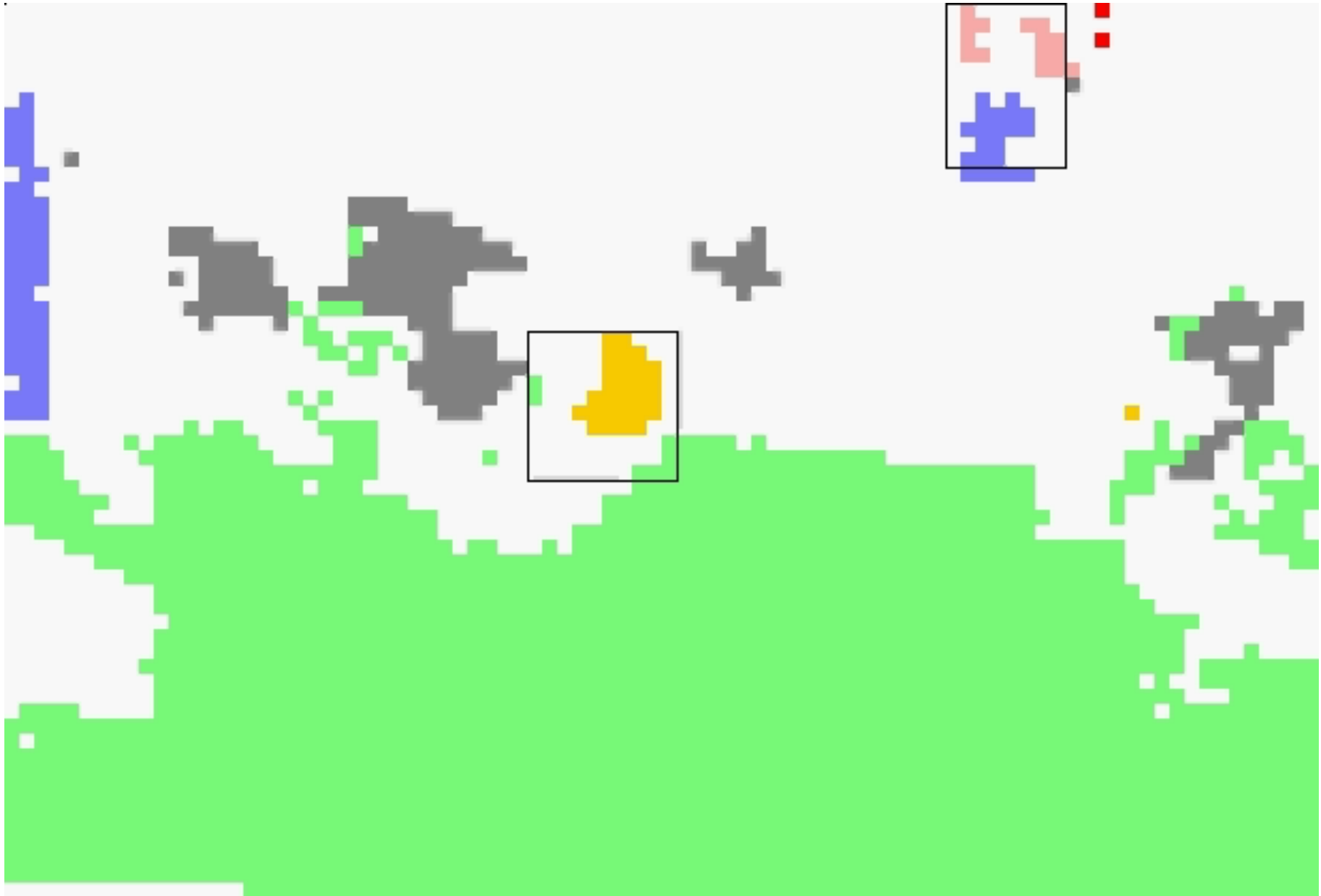
# Computer Vision in Action



# Doggie Cam (2)

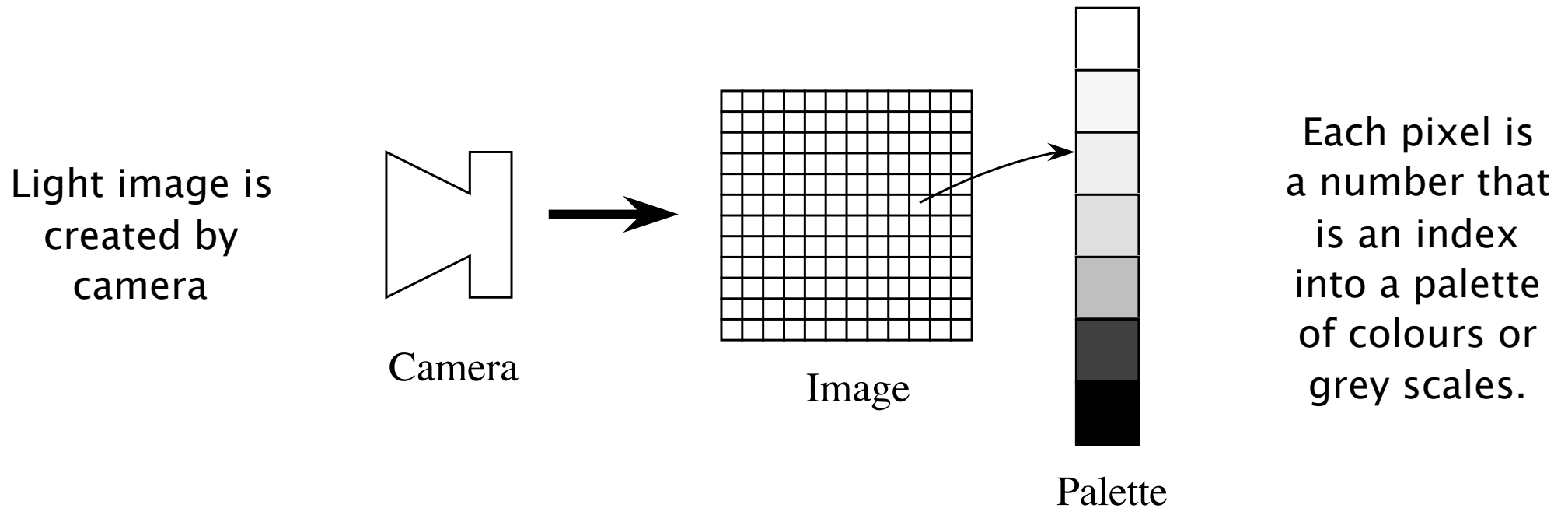


# What the robot sees



# Image Formation

# Image Formation



A "frame grabber" captures camera image and stores it in special purpose memory.

# Image Features

- Contrast
  - dullness or sharpness
  - e.g. if picture contains a lot of white and lot of black
- Dynamic Range
  - how much of grey scale is used)
- Frequency
  - amount of change between pixels on a line

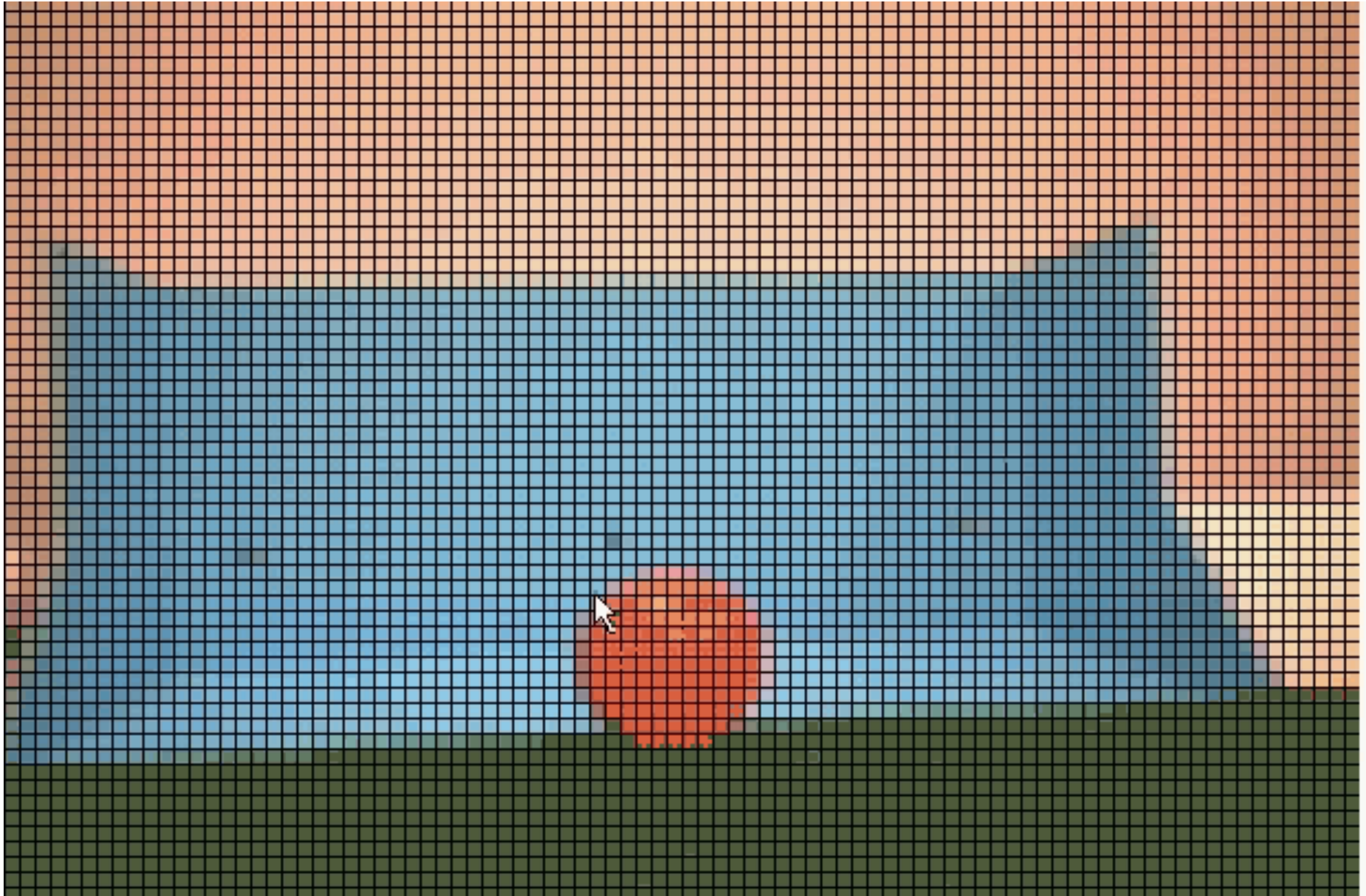
# Object Features

- Illumination (incident light)
- Reflectance (reflected light)
- Depth (distance from camera)
- Orientation (angle of normal to surface)
- Other features:
  - shading
  - colour
  - texture

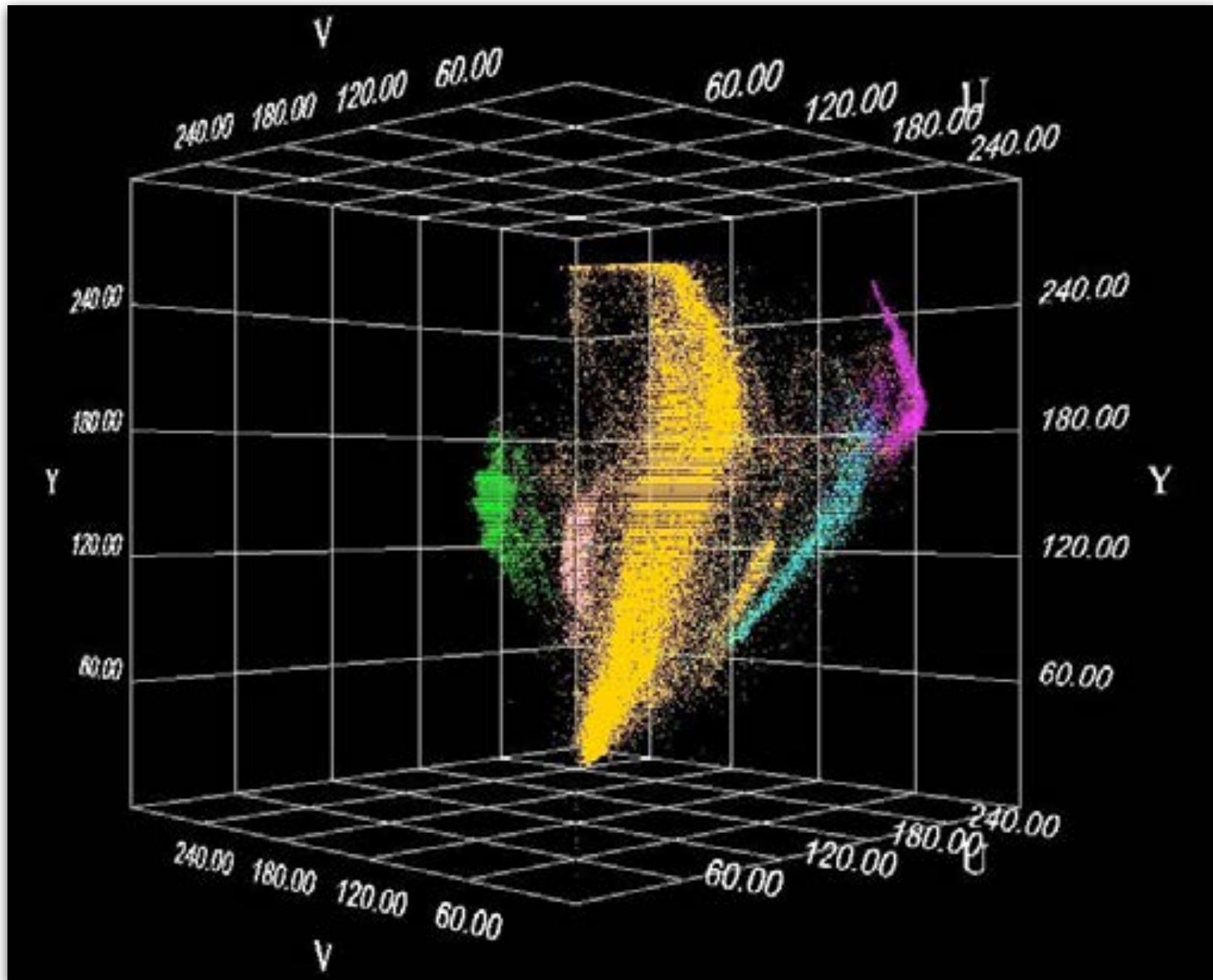


# Colour Recognition

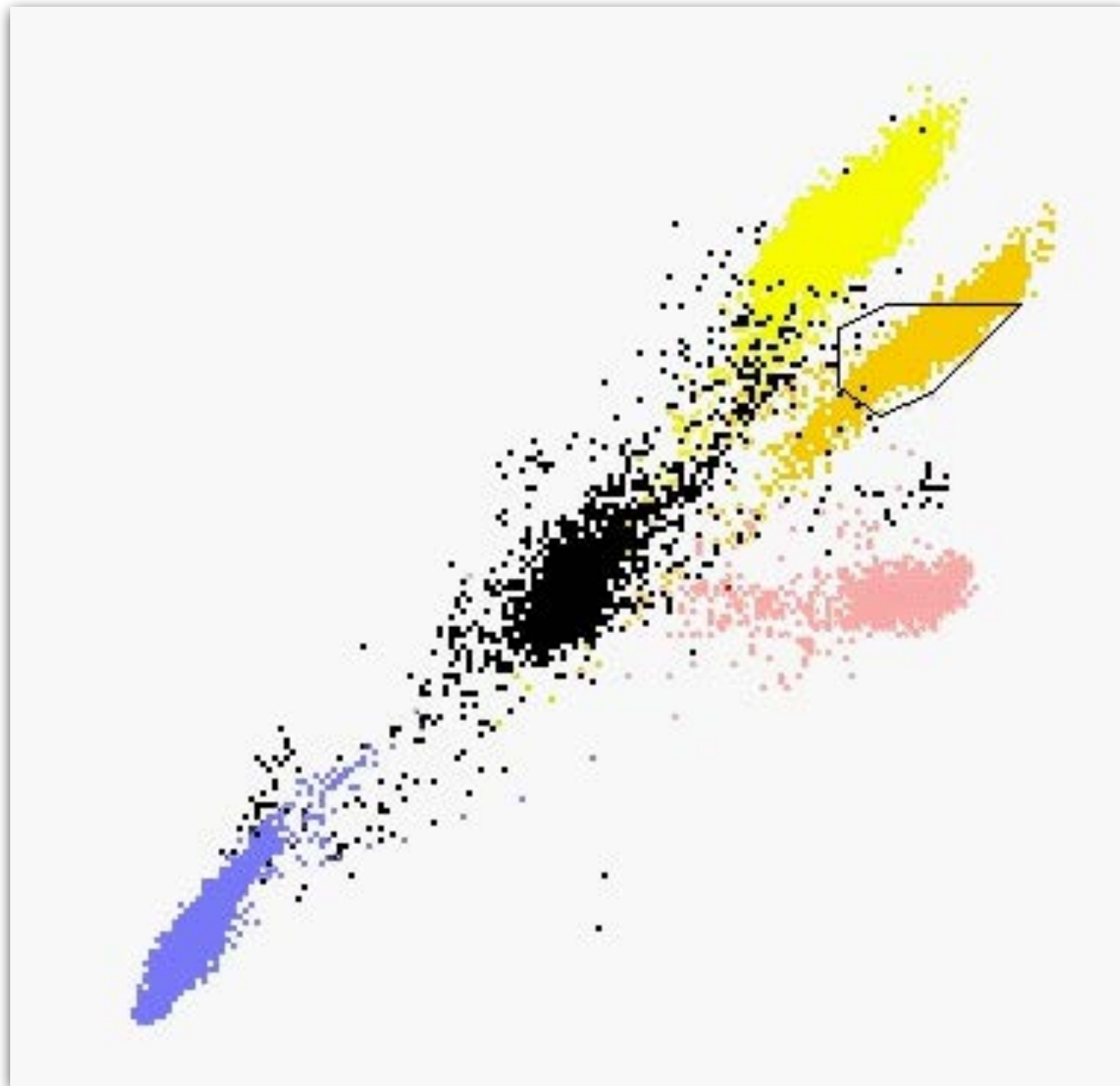
# Camera Image



# YUV Colour Space



# Polygons in UV Plane



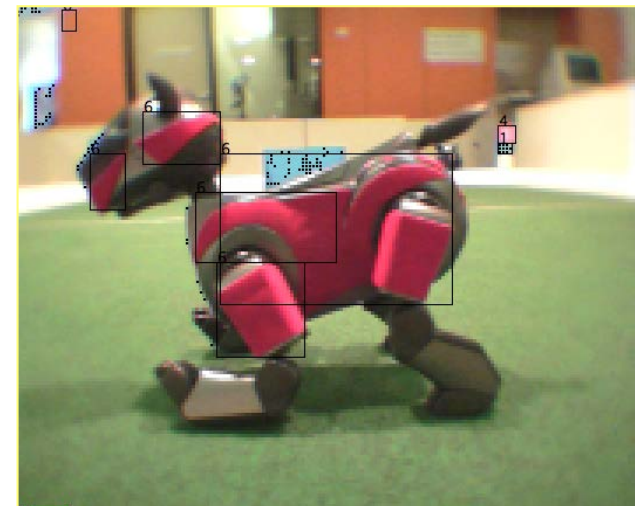
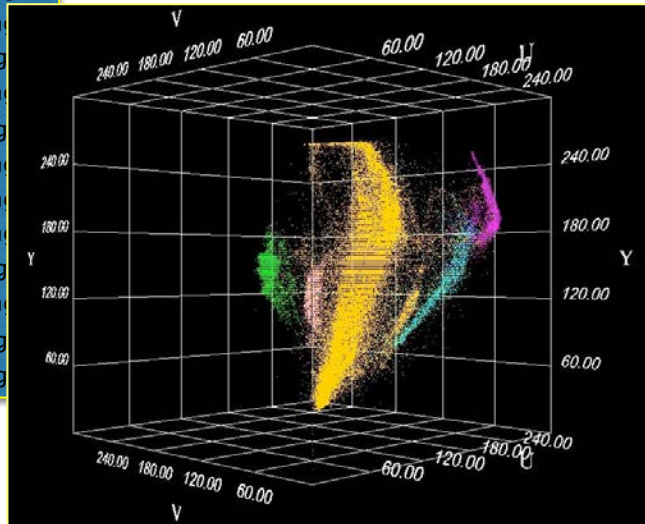
# Learning in Perception

105, 117, 113, orange  
105, 116, 112, orange  
102, 117, 113, orange  
102, 116, 114, orange  
103, 117, 111, orange  
103, 117, 112, orange  
103, 118, 110, orange  
99, 117, 112, orange  
98, 116, 118, orange  
99, 116, 117, orange  
106, 111, 114, orange  
114, 115, 123, yellow  
128, 111, 124, yellow  
150, 112, 121, yellow  
173, 111, 117, yellow  
171, 110, 110, yellow  
145, 112, 108, yellow  
121, 111, 110, yellow  
106, 111, 112, orange  
107, 112, 112, orange  
104, 114, 114, orange  
100, 115, 114, orange  
100, 117, 117, orange  
98, 115, 113, orange  
100, 114, 116, orange  
97, 117, 112, orange  
102, 115, 109, orange  
104, 118, 109, orange  
100, 114, 108, orange  
97, 115, 110, orange  
101, 114, 110, orange  
99, 116, 113, orange  
98, 116, 113, orange

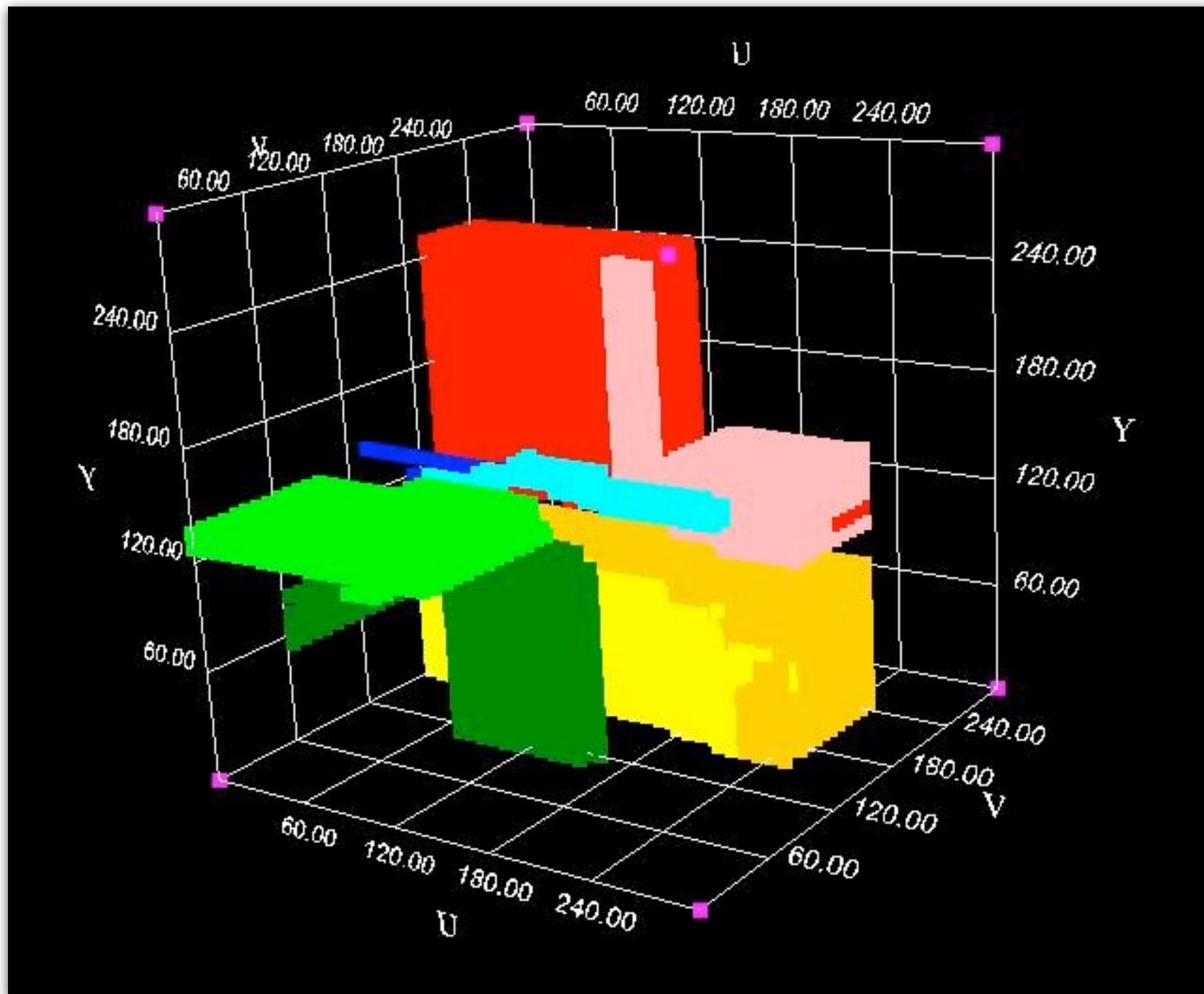
C4.5

Warning: In practice data sets and decision trees are much larger than this example!

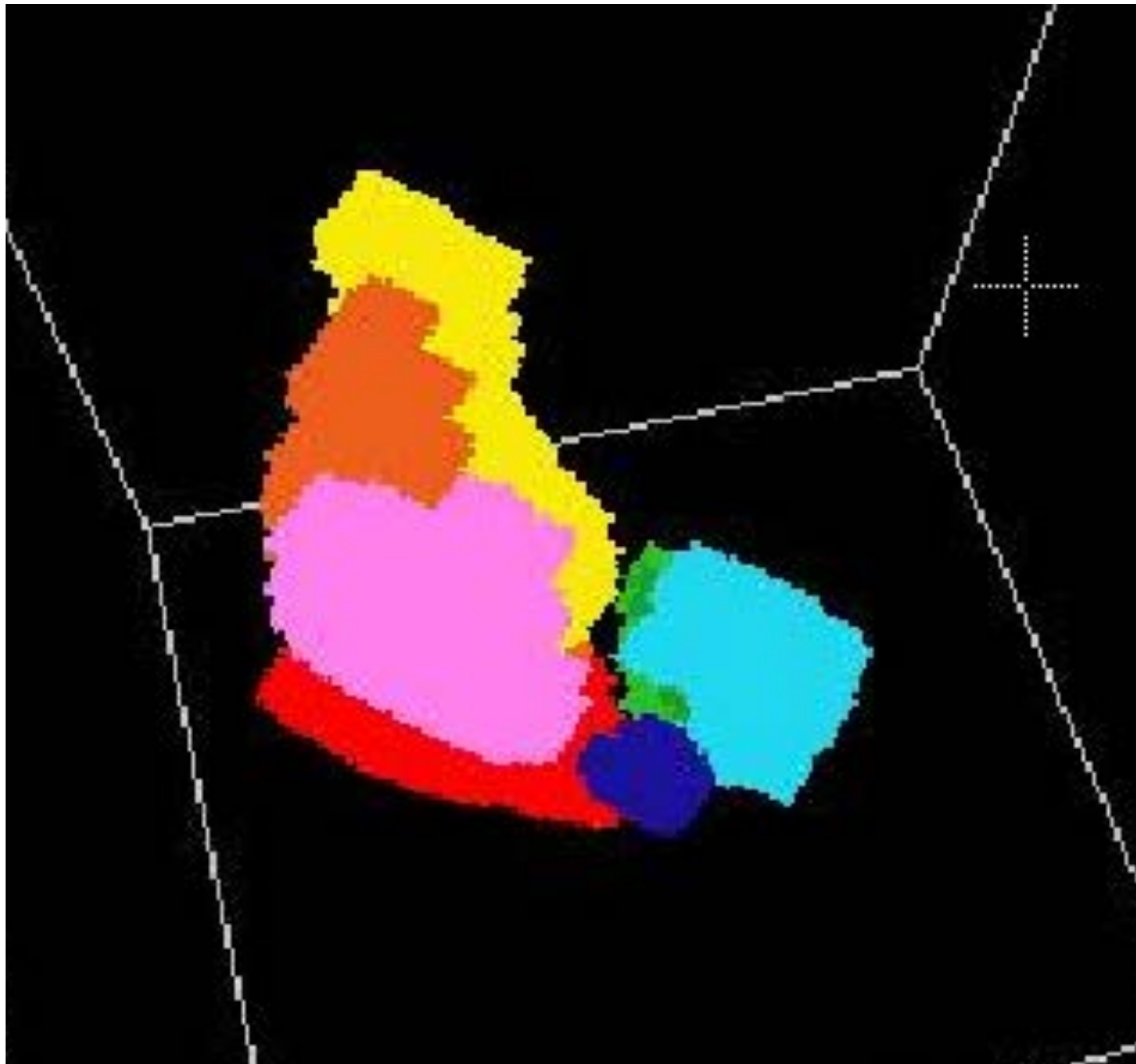
```
if (u <= 107)
  yellow;
else
  if (v <= 100)
    orange;
  else
    if (y <= 136)
      orange;
    else
      yellow;
```



# Colour Classes using C4.5



# Nearest Neighbour



# Binary Vision



# Binary Vision

- The original image is “thresholded”, i.e.

$$\text{new}[x, y] = (\text{old}[x, y] > \text{threshold})$$

- Every pixel brighter than a certain threshold is given a value of 1 otherwise it is zero.
- Easy to process and powerful enough to use in some industrial applications
  - e.g. picking parts from an assembly line.

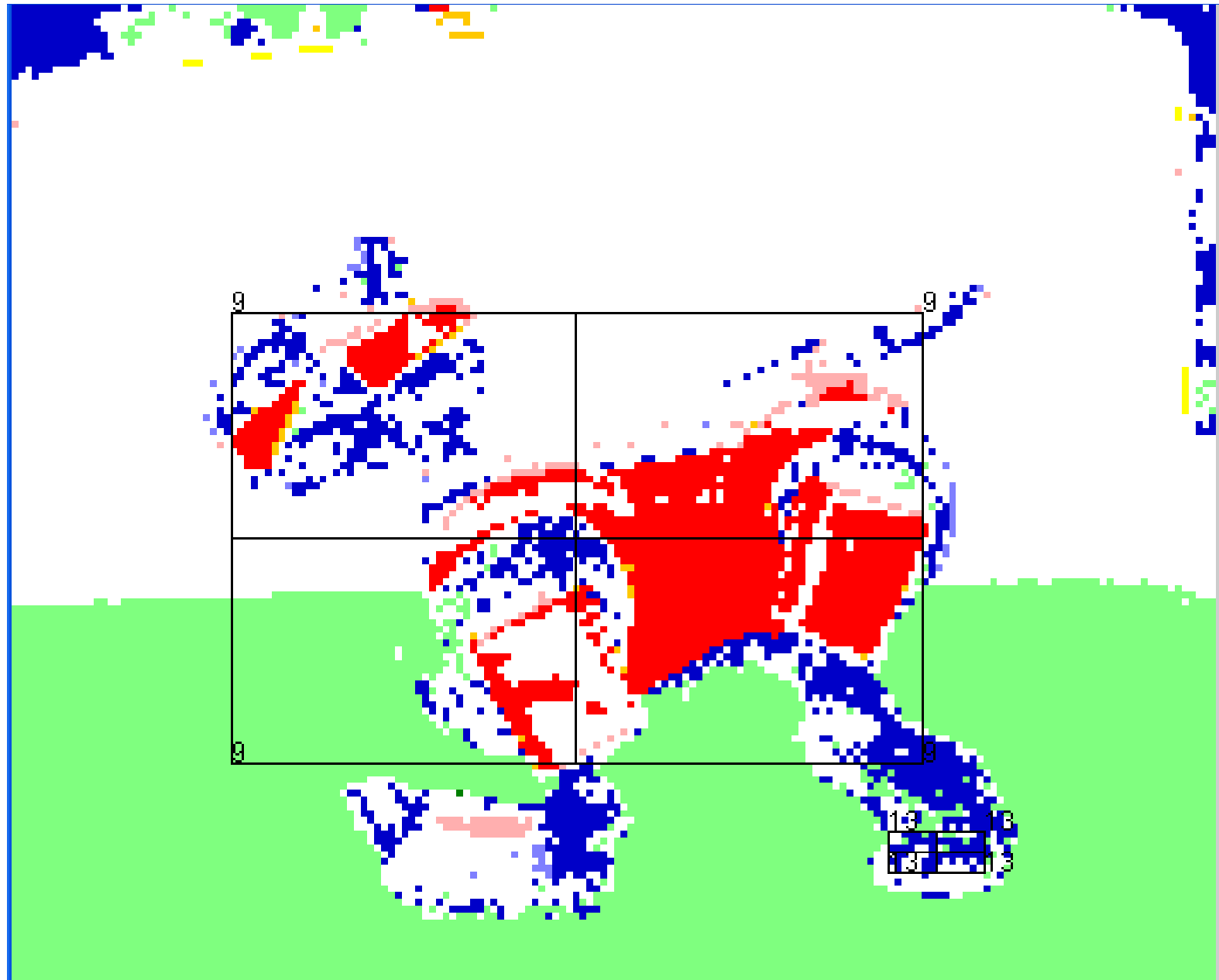
# Binary Vision

- Or use colour to create binary image

$\text{new}[x, y] = (\text{old}[x, y, u] \text{ is orange})$

- Use colour lookup as before to determine colour

# Blob Finding



# Blob Finding

(Connected Component Analysis)

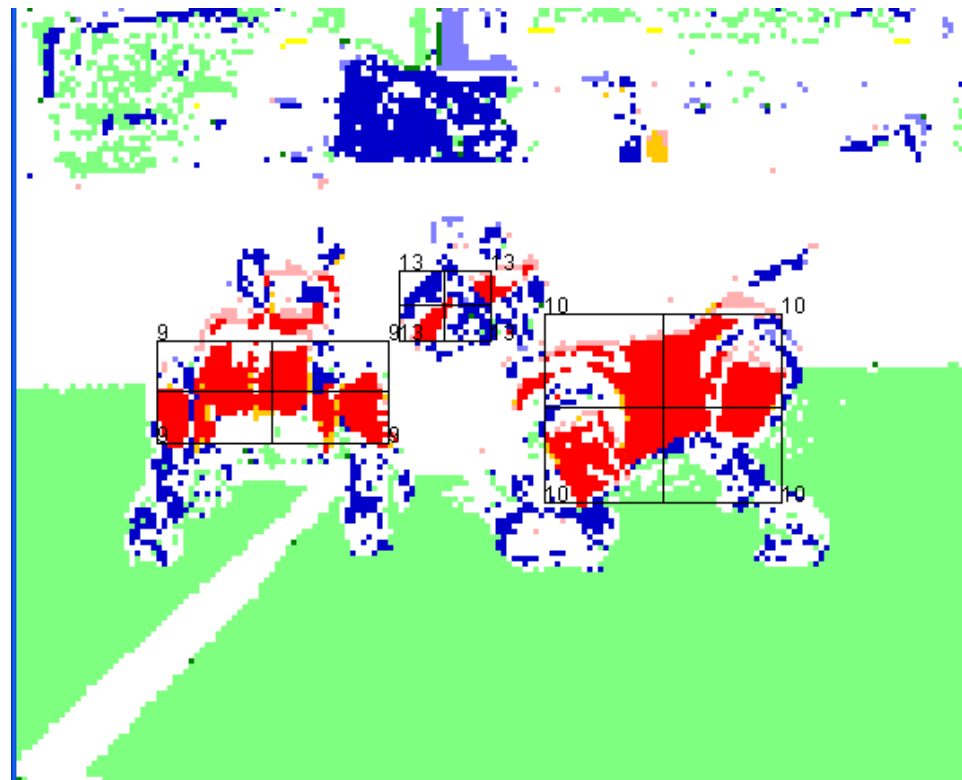
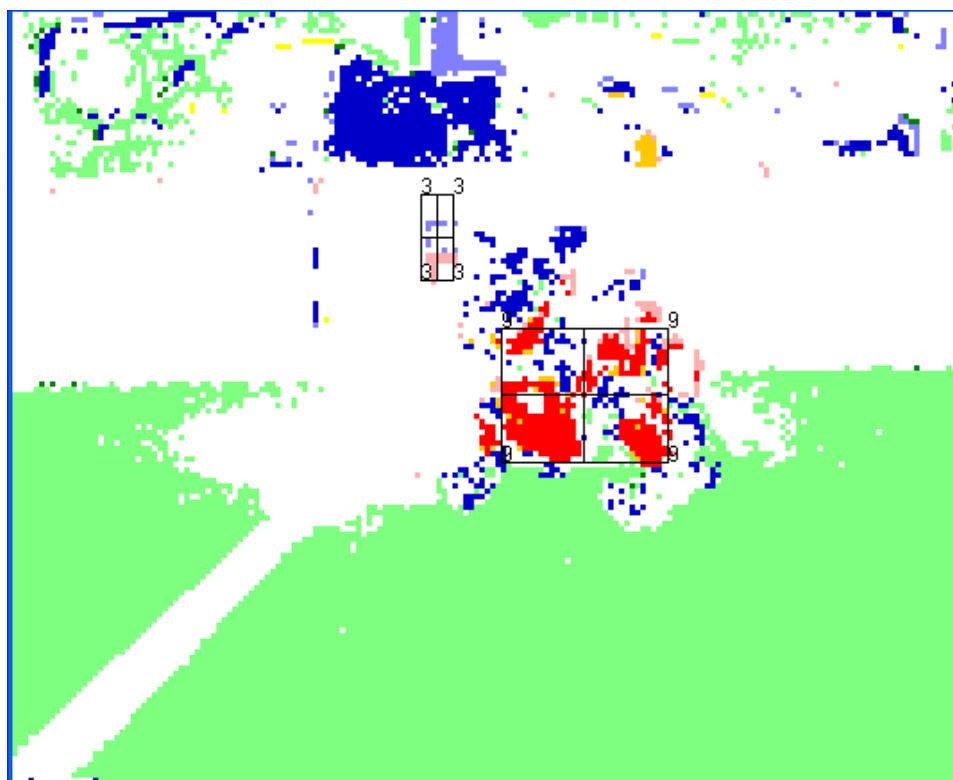
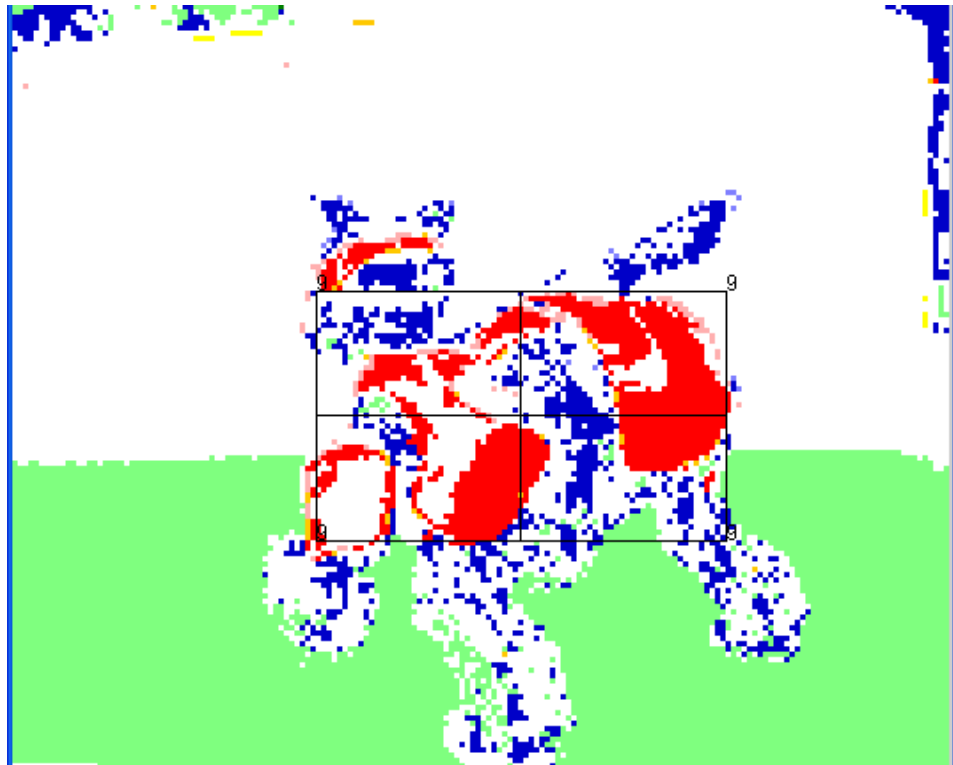
- How do we find a region in an image?
  - e.g. find the orange ball
- Assign a different number to every connected component of an image
- Requires two passes.
- First scan a 2 x 2 window over the binary image and observe the pattern:

**C**   **B**  
**D**   **A**

- Scan along each row from left to right, starting at the top.
- When we inspect cell A, cells B, C and D have already been labelled.







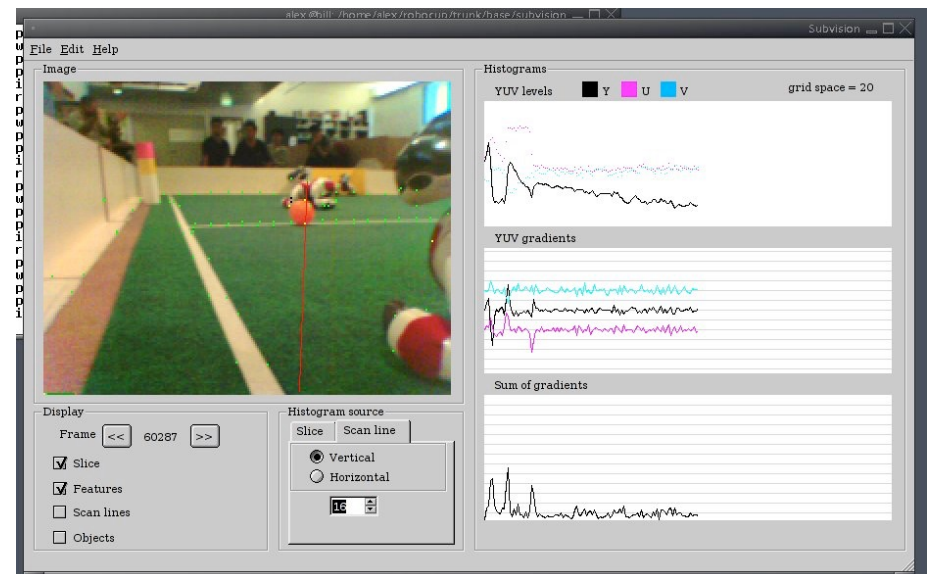
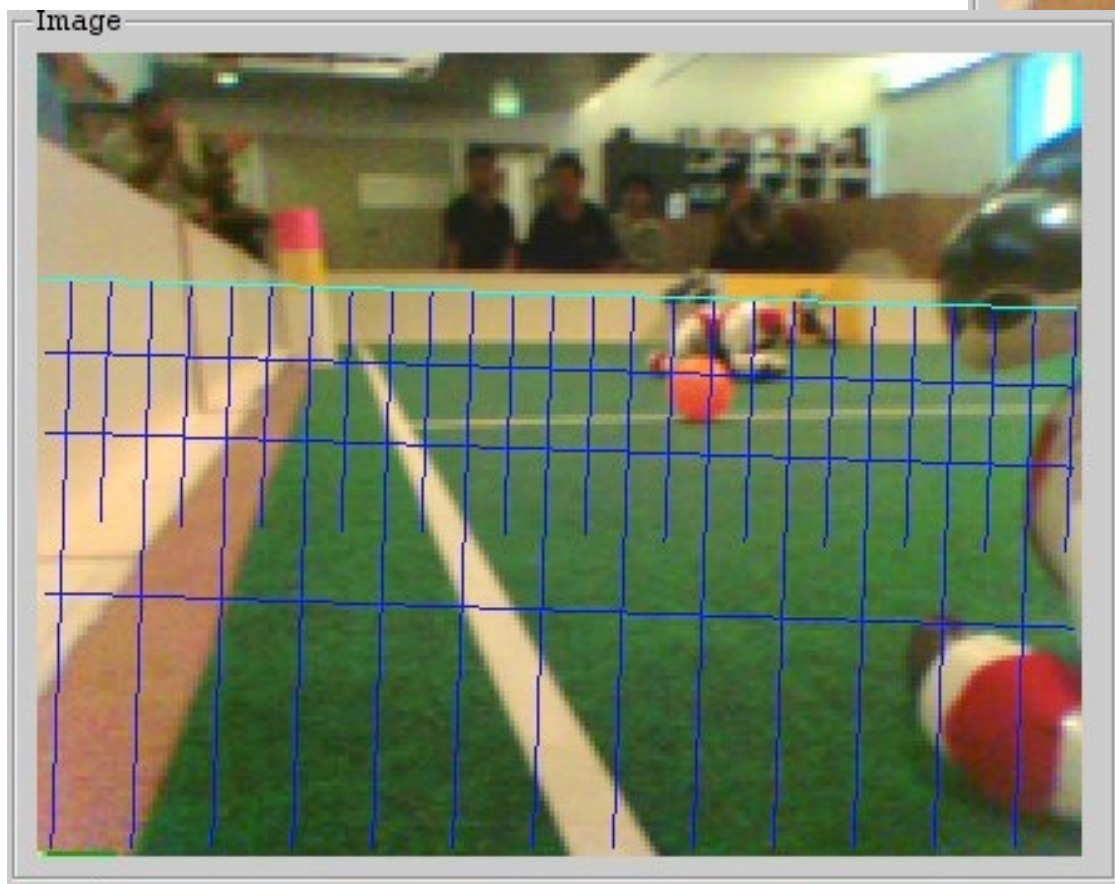
# Edge Detection



# Edge Detection

- The first step in robot vision system is to recognise the intrinsic features of the image.
- Some of the most important features are edges.
- There is evidence that the human vision system has edge detectors.

# Looking for Edges



# Edge detection requires several steps

1. Smoothing and sharpening of the image to remove noise.
2. Finding the edges by filtering the image.
3. Connecting lines from the edges found in the previous step.

# Smoothing

- Real images always contain noise.
- Smoothing tries to remove isolated bright and dark regions of a picture.
- One smoothing method is to replace the brightness value at each pixel by average brightness of eight neighbours.
- Called local averaging.
- Has side-effect of blurring image.

# Smoothing Example

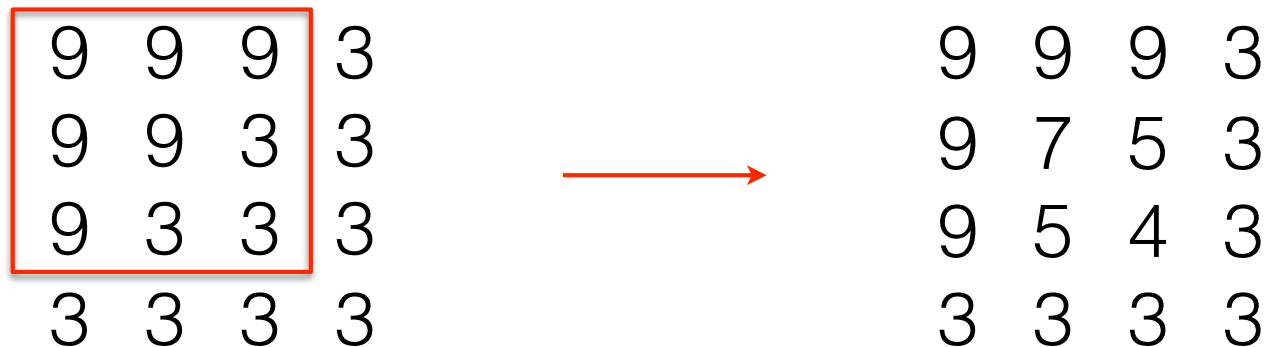
- Given a simple 4 x 4 picture matrix:

9	9	9	3
9	9	3	3
9	3	3	3
3	3	3	3

- Smooth this matrix using a local-averaging technique and a 3 x 3 pixel window.

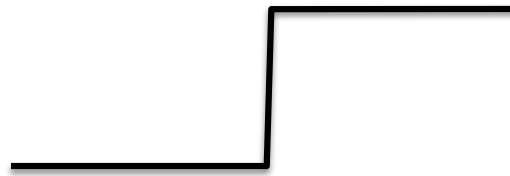
# Solution

- There are four 3 x 3 pixel windows in the matrix.
- Replace middle value in each window by average of all the values in the window.

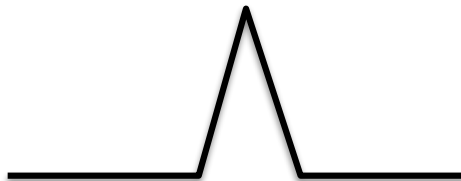


# Edge Detection Operators

- The ideal edge can be graphed on a grey-scale as:



- To find the edges in a grey-scale image we calculate the first-derivative of the adjacent grey-scale values, i.e. the gradient.



# The Roberts Cross Operator

The Roberts cross operator approximates the first derivative.

$$R(i, j) = \sqrt{[i(m+1, n+1) - i(m, n)]^2 + [i(m, n+1) - i(m+1, n)]^2}$$

where  $i(m, n)$  is the image intensity of pixel  $(m, n)$ .



# Roberts Operator Example

- Replace grey-scale values with values obtained using the Roberts operator.
- If a Roberts value cannot be obtained for given pixel, replace that pixel with an X.

9	9	9	3
9	7	5	3
9	5	4	3
3	3	3	3

# Roberts Operator Example (Solution)

- Applying the Roberts operator to each 2 x 2 window in the picture gives:

2.0 4.5 6.3 X

4.5 3.0 2.2 X

6.3 2.2 1.0 X

X X X X

- Apply threshold to get a binary image.

# Thresholding

With a threshold of 4:

0	1	1	X
1	0	0	X
1	0	0	X
X	X	X	X

With a threshold of 6:

0	0	1	X
0	0	0	X
1	0	0	X
X	X	X	X

- In both matrices connecting ones gives the edge.
- 6 is better.

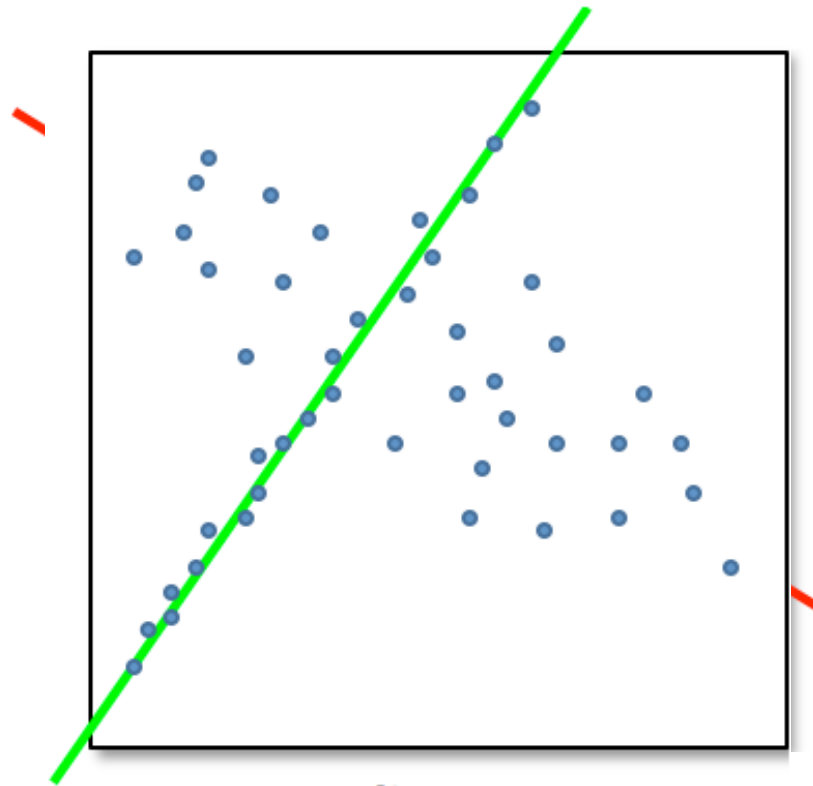
# Line Finding

# RANSAC

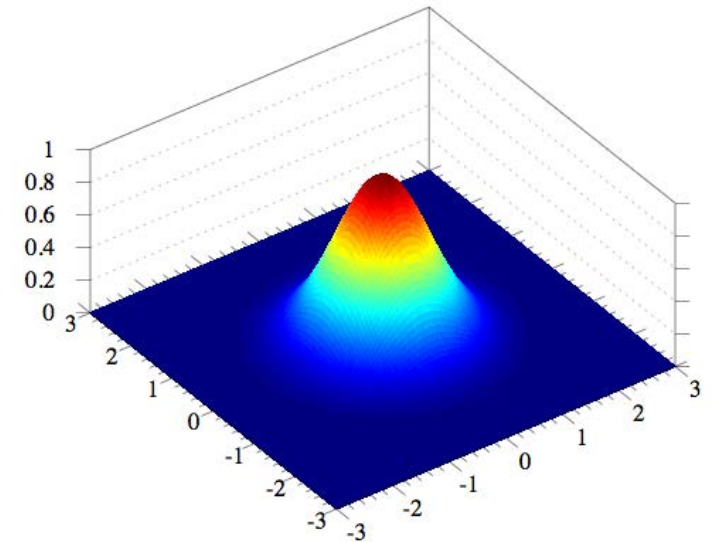
(RANDOM SAMPLE CONSENSUS)

1. Select randomly the minimum number of points required to determine the model parameters.
2. Solve for the parameters of the model.
3. Determine how many points from the set of all points fit with a predefined tolerance  $\epsilon$ .
4. If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold  $\tau$ , re-estimate the model parameters using all the identified inliers and terminate.
5. Otherwise, repeat steps 1 through 4 (maximum of  $N$  times).

# RANSAC



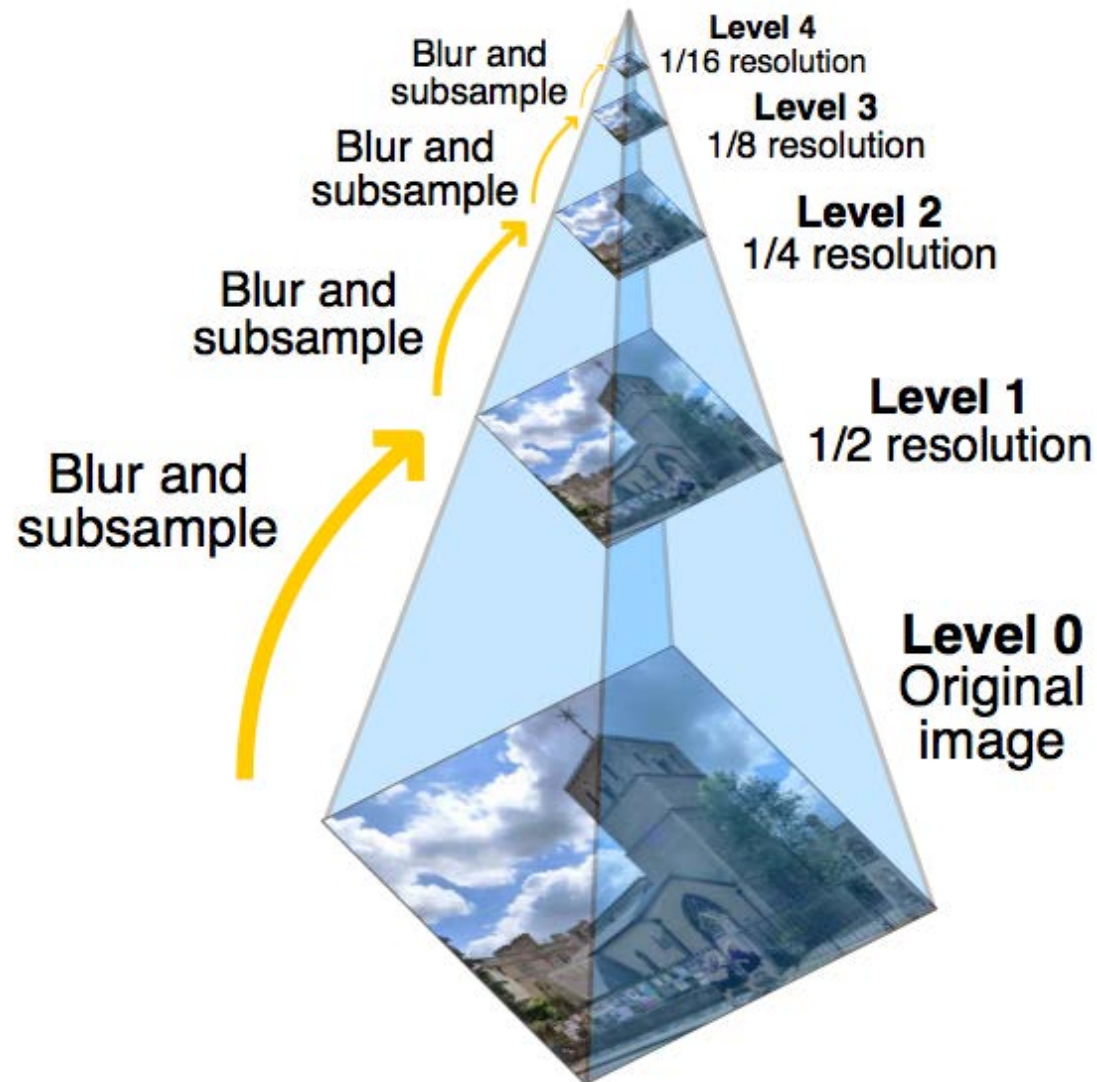
# Gaussian Blur



Weighted average  $G(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

0.00000067	0.00002292	<b>0.00019117</b>	0.00038771	<b>0.00019117</b>	0.00002292	0.00000067
0.00002292	0.00078634	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
<b>0.00019117</b>	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	<b>0.00019117</b>
0.00038771	0.01330373	0.11098164	<b>0.22508352</b>	0.11098164	0.01330373	0.00038771
<b>0.00019117</b>	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	<b>0.00019117</b>
0.00002292	0.00078633	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00000067	0.00002292	<b>0.00019117</b>	0.00038771	<b>0.00019117</b>	0.00002292	0.00000067

# Image Pyramid





# SIFT Features

## (Scale-Invariant Feature Transform)

1. **Build a Difference of Gaussians pyramid.** Take the difference of adjacent pairs of images from the scale-space pyramid.
2. **Find the extrema points in the Difference of Gaussians pyramid.** Each point in the pyramid has 26 neighbours, eight on the same scale and nine each in the above and below scale. The points that are the minimum or maximum of their 26 neighbours are the extrema points. Typically found near corners and edges in an image at a given scale.
3. **Rejection of unstable extrema points.** Points with a low absolute value in the Difference of Gaussians pyramid or points that are too edge-like are unstable in the presence of noise. Edge points are similar to nearby points.
4. **Orientation assignment.** Each remaining key-point is assigned a principal orientation. This is the most prominent gradient direction of a small neighbourhood of pixels around the key-point.
5. **Generation of key-point description vectors.** For each key-point, a neighbourhood of pixels is used to build an array of histograms of gradients. The histograms are normalised to the principal orientation of the key-point to make the descriptor vector rotation invariant. The histogram is normalised on gradient magnitude of the pixel neighbourhood to increase robustness to changes in contrast and lighting. The result is a 128-dimensional description vector for each key-point.

# SIFT Features

