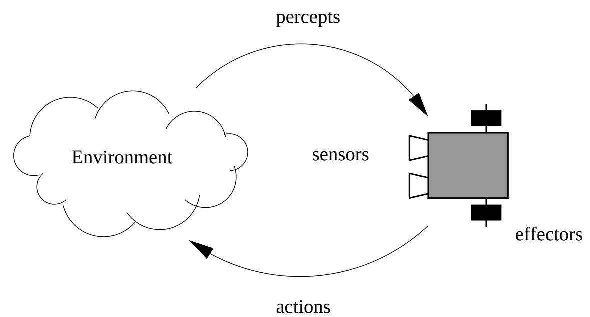
VISION

Introduction

- We have said that the agents are in a feedback loop with their environment:



without saying how the agent perceives its environment.

- In this lecture we will look one way this might be done.

Sensors

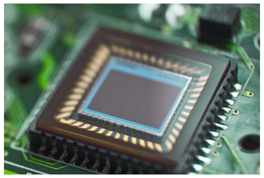
- There are many kinds of sensor:
  - acoustic,
  - infra-red,
  - temperature,
  - pressure,
  - touch,
  - ...
- These give more or less information about an agent's environment.
- Vision is cheap and gives lots of information.

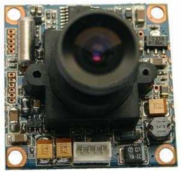
- Although vision seems to be easy for humans, it is hard for machines.

  (as always, remember how long it takes us to learn to "see").
- Reasons include:
  - variable illumination,
  - uncontrolled illumination,
  - shadows,
  - irregular objects,
  - occulsion of objects,
  - noisy sensors,
  - ...
- Typically these problems are worse outside.

The vision process

- The first step is to create an image.
- Use an array of photosensitive devices typically a charge-coupled video camera.





- These devices are the reason vision is now cheap.

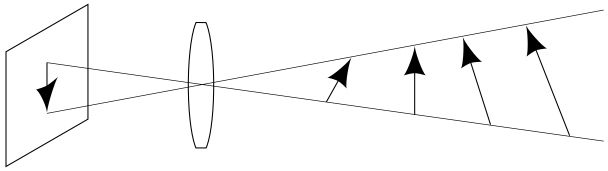
- The lens produces a *perspective projection* of the scene.
- The 3-d scene becomes a 2-d image:

$$I(x, y, t)$$

x and y are the co-ordinates of the array, t is time.

- The image is just an array.
- Well, typically 3 arrays — each with one entry per pixel in the image.
  - Why?
- These must be processed to extract the information that we need.

- The projection is a many-to-one mapping:



- Many scenes can produce one image.
- Noise, bad light etc. also impact the image.
- Thus we need to process the image to extract information.
- Usually we use knowledge about:
  - general properties of objects; and
  - specific objects likely to be in the scene

  to do the extraction.
- Exactly what is extracted depends on what the agent is doing.

- Navigation requires:
  - locations of objects;
  - boundaries of objects;
  - location of openings;
  - surface properties.
- Manipulation requires:
  - locations of objects;
  - size of objects;
  - shapes of objects;
  - composition of objects;
  - textures of objects.
- Other tasks might require colour recognition, classification.

Soccer

- For robot soccer we can do good enough vision quite simply

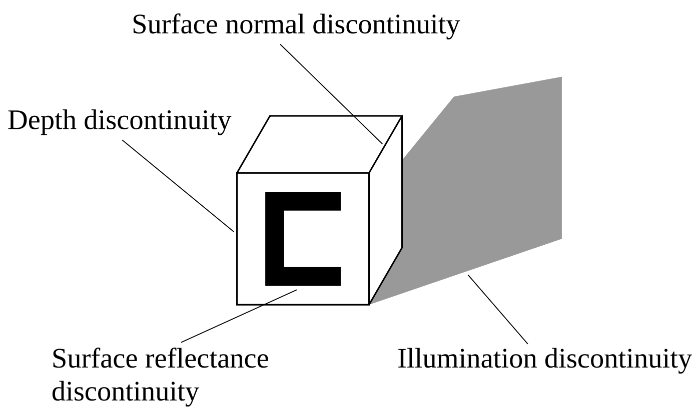
- We just look for blocks of color.

Two stages of machine vision

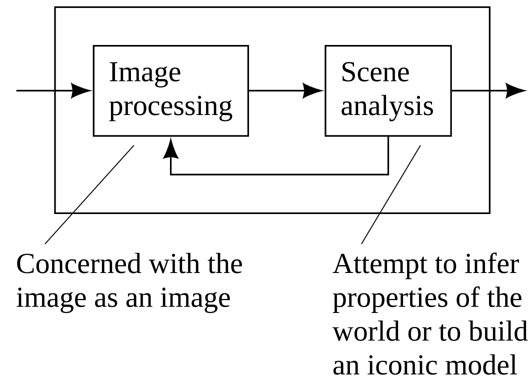
- Unfortunately in many cases we need more sophisticated vision than we can get away with for soccer.

  (In fact we need to do better than this to play soccer well).
- Consider what we need when looking at objects.
- What is an object?
  - doorways, furniture, people, walls, ...
  - animals, plants, buildings, roads.
- Typically man-made environments are easier (sharp edges).
- Two techniques in particular are important.

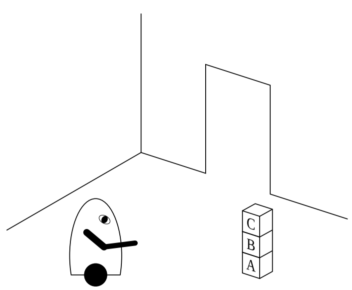
- One looks for *edges*.
- An edge is where intensity or some other property changes.
- Another technique looks for *regions*.
- A region is an areas in which intensity (or some other property) changes only slowly.
- Often changes between regions (across edges) are important in a scene.
- The might mark changes in
  - depth;
  - illumination
  - surface, ...



- Image processing transforms an image into a form that makes it easier to do scene analysis on.



- So in the following example:



- Image processing would eliminate noise and sharpen edges.
- Scene analysis would produce an iconic representation like:

$$On(C,B)On(B,A)On(A,Floor)$$

Image processing

- There are many different aspects to image processing.
- We will look at a couple of these:
  - Averaging;
  - Edge enhancement.
- For averaging, we start with an $m \times n$ array of pixels:

$$I(x,y)$$

where this is light intensity.

- Irregularities (noise) in the image can be removed by smoothing.
- We run an *averaging* window over the image.
- Each pixel in turn is in the centre.
- We compute a weighted sum of all the surrounding pixels.
- This sum replaces the original pixel value (or can be compared with a threshold if we want 1 or 0).
- This process is called *convolution*.
- The bad side-effect of this is to reduce crispness and lose detail.

- A full mathematical description of a discrete version of convolution is:

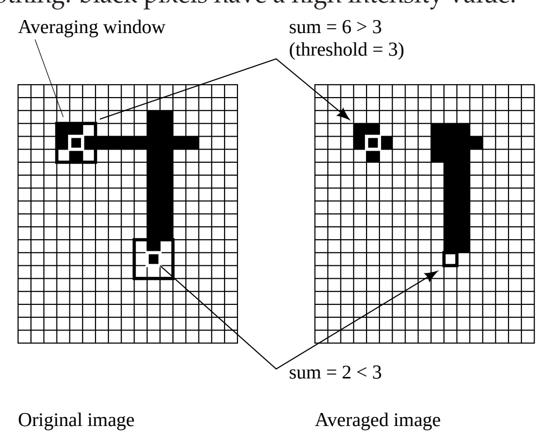
$$
\begin{aligned}
I^*(x,y) &= I(x,y) \star W(x,y) \\
&= \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} I(u,v)W(u-x, v-y)
\end{aligned}
$$

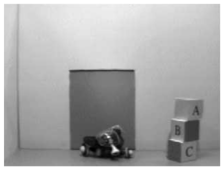
  where $W(u,v)$ is a weighting function.
- But in practice, it isn't so hard to apply this.

- It is fine to think of the formula on the previous slide as:

  replace the value of each pixel by the weighted average of its neighbors.
- We assume that $I(x,y) = 0$ when $x < 0$ or $x \geq m$ and $y < 0$ or $y \geq n$.
  - So we ignore pixels outside the image.
- Often the window is a rectangle where $W(x,y)$ is 1 for pixels in the rectangle and 0 for pixels outside.
- A large rectangle will smooth more than a small one.

- Smoothing: black pixels have a high intensity value.



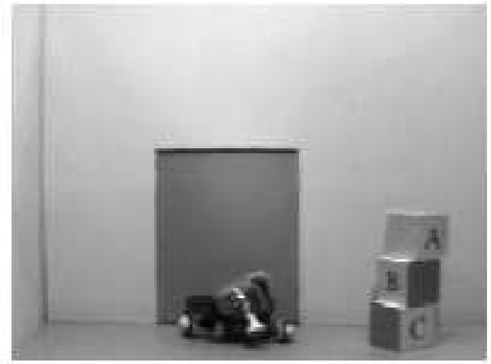


(a) Original image

(b) Width of Gaussian = 2 pixels

(c) Width of Gaussian = 4 pixels

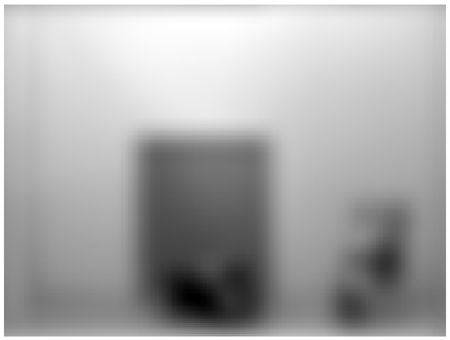
(d) Width of Gaussian = 8 pixels



(a) Original image

(b) Width of Gaussian = 2 pixels

(c) Width of Gaussian = 4 pixels

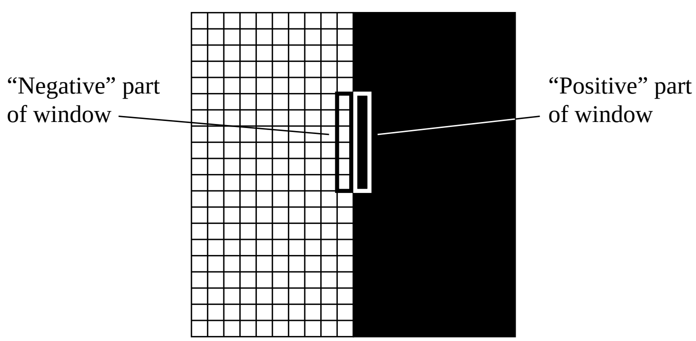


(d) Width of Gaussian = 8 pixels

Edge enhancement

- We often want to extract edges.
- We can then use the edges (as we will see later) in some form of scene analysis.
- One way to extract edges starts by enhancing the edge.
- Let's consider how this is done in 1-dimension.
- Thus $I(x, y)$ varies only along the x dimension.
- We do this once again by convolution.

- In this case we convolve with a part-negative, part-positive window:



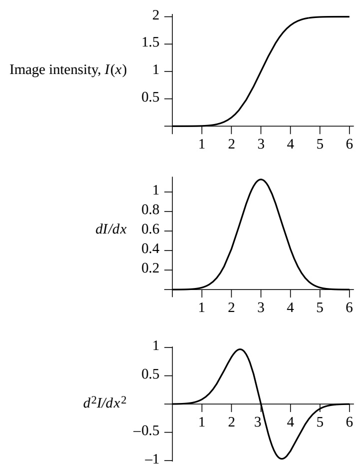
- If we convolve this in the x direction over an image, we get a peak where an edge is aligned along the y direction.
- This is rather like taking the derivative:

$$\frac{dI}{dx}$$

of the intensity wrt x.
- We get something even more distinct if we take the second derivative:

$$\frac{d^2I}{dx^2}$$

- If the intensity changes smoothly, then we get something like...



- The steeper the change in intensity, the narrower the peak.
- The edges are when:

$$\frac{d^2I}{dx^2} = 0$$

- This is when the second derivative crosses the y axis.

- As with smoothing, we don't have to understand the math to find edges.
- We an just look for pixels where the intensity changes.
- Usually produces a prety good approximation to the edge of objects.

Scene analysis

- Once the image has been processed, we can extract information from it.
- Because of the many-to-one mapping in image-formation, we either need to use additional information or have additional images (as in stereo vision).
- We can use either very specific information
  - Likely contents of a scene

  or very general information
  - Surface reflectivity
- Then the form of analysis depends upon what kind of information we want to extract.

Interpreting lines

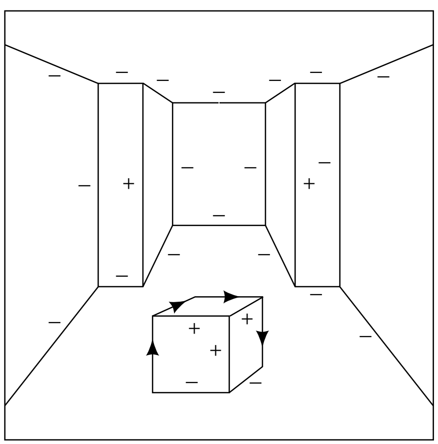
- When dealing with images which include rectilinear objects we need to identify and handle lines.
- We can create these by fitting lines to edges or region boundaries.
- This can then be post-processed to:
  - merge small sections of line
  - eliminate odd sections
  - join sections together
- Then it is ready for interpretation.
- One technique for doing this is as follows.

- It works for scenes where all surfaces are planar and no more than three intersect at a point.
  - Trihedral vertex polyhedra
- There are only three ways two planes can meet in an image.
- One is where one plane occludes another

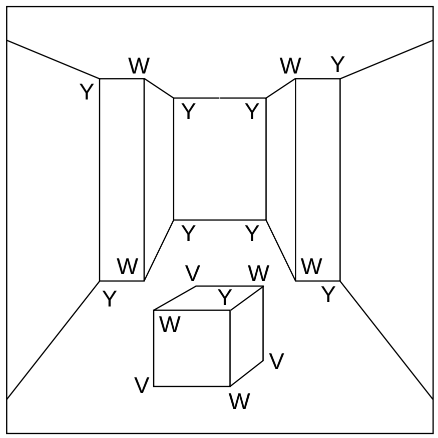
  (marked by an arrow where the occluding plane is to the right of the arrow)
- Alternatively two planes can make a *blade* where both planes are visible and the edge is convex

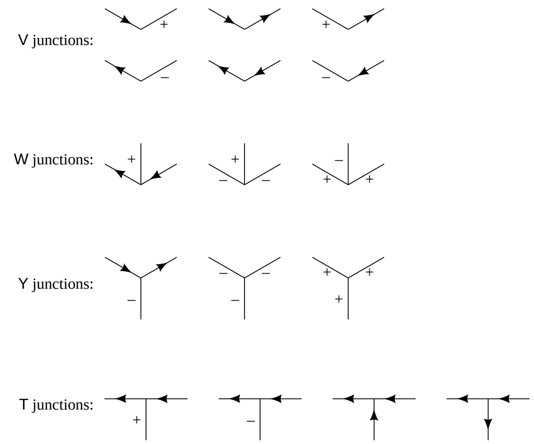
  (marked by a +)
- Or they can make a *fold* where the edge is concave

  (marked by a -)



- A typical thing to do is to try and label the lines in the image to try and describe the scene.
- That is, to come up with the the +, -, $\rightarrow$ labelling working from the raw lines.
- It turns out that this can be done as long as the image is not pathalogical.
- We start by identifying if the vertices are:
  - V;
  - W;
  - Y; or
  - T;





- Then we try to come up with a consistent labelling knowing what the possible labellings of each type of junction are.
- When we do this, an edge must have the same labelling at both ends.
- This allows us to use *constraint satisfaction* to put the labelling together.

- In the constraint satisfaction framework we talked about before, the variables are the vertices.
- The domains of the variables are the sets of labels from two slides ago.
- Constraints come from the fact that edges between vertices can have only one label.
  - The edge has to be the same at both ends.
- A consistent set of values for variables is an interpretation of the scene.

- This kind of information, although not *that* extensive, does help.
- For example, a robot could head towards vertical folds to find corners.
- Or avoid them so as not to get stuck there.
- A robot could circumnavigate obstacles by skirting vertical blades.

Summary

- In this lecture we looked a little at some of the issues in machine vision.
- Our motivation was the common use of cameras as a sensor for intelligent agents.
- We considered three techniques:
  - Smoothing.
  - Edge detection
  - Constraint-based scene interpretation.
- The last of these demonstrates a practical use of constraint satisfaction from the first half of this lecture.
- This just scratches the surface of machine vision, but it is also enough to get you started (if that is ever necessary).