

Materials:

1. Projectables of Tanimoto figure 10.19 top; page 413
2. Projectables of Winston figures 10-9, 10-10, 10-11, 10-12, 10-13
3. Demonstration of CS112 Project 3 edge detection
4. Projectable of Winston figure 3.1, 3.22
5. Projectable of Winston figure 3.17 with and without a "window"
6. Projectable of Winston figure 3.14

Introduction

- A. We have seen again and again that what is easy for humans (and even animals) is often hard for computers and vice versa. One example of this that we have noted is "common sense" knowledge representation, and another is language understanding. Vision is yet one more area where this pattern holds.
- B. The area of computer vision is a vast one, and the techniques involved draw on a variety of disciplines: AI, electronics, matrix algebra etc.
 1. The present state of the art is mostly a vast collection of techniques, each of which is useful for some part of the task for some classes of problems. No comprehensive overall approach is known.
 2. We only have time to get something of the "flavor" of the field by looking briefly at one of many approaches to a one part of the overall problem.
- C. Computer vision is an important AI problem arising originally from research in robotics. There are many applications where it is important to make use of visual data, typically obtained by a TV camera.
 1. The simplest variant of this problem arises in controlled environments such as industrial production lines. Here it is possible to precisely control the illumination so that shadows do not give rise to confusion.
 2. The more general problem arises from applications where vision must be used in uncontrolled environments (e.g. outdoors.)
 3. Applications of vision have moved beyond robotics, though, to encompass such things as interpretation of satellite data etc.

I. Overview of the Problem

- A. In most cases, the raw data for vision is an array of brightness values.
 1. For example, a black cube viewed face on (so it looks like a square) against a white background might produce an ideal array like the following (where 0 = black and 1 = white):

```

1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1

```

2. In practice, of course, the brightness values are actually real numbers, not booleans. Typical vision systems use a gray scale whereby each point in the picture (pixel) is represented by an 8 to 16 bit integer. For example, using 8 bits, total darkness would be encoded as 0; maximum brightness as 255, and a medium gray as 127.
3. Real problems involve further complications, too. One must deal with the following:
 - a. Noise - random variations in intensity due to atmospheric dust and imperfections in the camera system. That is, the pixel data may not accurately represent what's actually there.
 - b. Variations in light intensity due to differences in surface texture, surface orientation, and illumination. That is, the pixel data may accurately represent the image, but the variation may not be caused by any feature of the object per se.
 - c. Multiple objects
 - d. Shadows
 - etc.
- B. The goal of vision is generally a description of the objects present in the scene:
 1. Their shape
 2. Their size
 3. Their position
 4. Their orientation
 5. Sometimes, their speed and direction of motion (obtained by comparison of multiple images of the same scene taken over time.)
- C. The goal can be arrived at by a series of steps like the following:
 1. Preprocessing to filter out at least some of the effects of noise.
 2. Conversion of the raw data into a line drawing (a set of edges)
 3. Segmentation: identification of the general regions making up the image.

4. Recovering 3-D information from the 2-D image.
5. (Sometimes) Recovering motion information from multiple images.
6. Arriving at an overall interpretation of the image.
7. For the purpose of this lecture, we will focus on steps 1-3: preprocessing, edge detection, and at least some aspects of segmentation.
 - a. The first two portions of the vision problem are called low-level vision. They are some of the best-understood aspects of the problem, and some of the methods developed by AI seem to bear a close resemblance to what actually happens in living creatures. In particular, it appears that the retina of the eye handles this portion of the problem, leaving the remaining aspects to the brain. (This portion of the problem involves only local computation. The global computation arises at the next step.)
 - b. In the third portion, an early technique known as the Waltz procedure was one of the first areas in which constraint satisfaction was used as an AI problem-solving tool.
 - c. Time (and my own knowledge!) will not allow us to say much about higher level steps in the process.

II. Low-Level Vision

--- -----

- A. The goal of low-level vision is to convert raw brightness data into a line drawing, showing the edges of objects in the image.
 1. It appears that, in living creatures, much of this work is done by cells in the eye and optic nerve, before the data reaches the brain.
 2. In present-day AI vision systems, this work is usually done by performing various operations on a matrix of light intensity values, using nested loops to iterate through all the elements of the array, or special parallel hardware, perhaps integrated with the "camera".
- B. There are many ways of approaching low level vision. We consider only one, which is interesting because there is evidence to suggest that it is similar to the approach taken by our own eyes.
- C. The basic idea is this:
 1. First, we reduce the effects of noise by adjusting each pixel's value by taking into consideration the values of its neighbors.
 2. Second, we differentiate the intensity function to get a rate of change of intensity (which will be high near an edge.)
 3. Third, we differentiate the rate of change function. The zeroes of this derivative will correspond to the maxima in the rate of intensity change function.
 4. All of these operations must be done in two dimensions, of course.

5. We now consider each step in more detail. To keep our discussion simple, we will handle the two dimensions by doing each operation separately along each of the horizontal and vertical axes. (In practice, at least the averaging is done two dimensionally.)

D. The effects of random noise can be reduced by replacing the observed intensity value at each point with an average involving the point and its neighbors in either direction.

1. In the simplest case, if we handle each dimension separately, we might consider just the immediate neighbor on either side.

Let I_i be the observed (raw) intensity of pixel i .

Let A_i be the adjusted intensity of pixel i , after averaging.

Then we can calculate $A_i = (I_{i-1} + I_i + I_{i+1})/3$

(Of course, in a real system, we would do this in two dimensions, averaging a pixel with either its 4 immediate neighbors or the 8 pixels surrounding it)

2. A better approach is to use a Gaussian function, in which both near and distant neighbors of the point in question are considered, but with varying weights. In one dimension, we would have

$$A_i = \frac{\sum_{\text{all } j} I_j * e^{-\frac{(i-j)^2}{\sigma^2}}}{\sum_{\text{all } j} e^{-\frac{(i-j)^2}{\sigma^2}}}$$

for some decay constant σ
(where a small value of σ results in giving greater preference to near neighbors)

a. Notice that points near point i get high weight while those far away get small weight.

b. The weight curve has the familiar "bell curve" shape.

TRANSPARENCY: Tanimoto figure 10.19 top figure (one dimensional version)

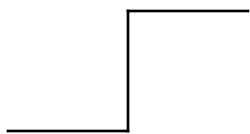
c. By rotating this curve around its center, we can obtain a two dimensional version. This can, in turn, be approximated by an array of weights like the following

TRANSPARENCY: Tanimoto page 413

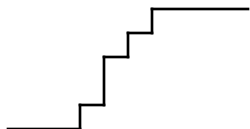
That is, the average intensity at some point would be calculated by multiplying each of the raw intensities in a 5 x 5 grid by the value specified in the table, adding these 25 values together, and dividing by the sum of the weights in the table (5.8). [In practice, the weights used would be divided by this sum to begin with - e.g. the middle value would be $1/5.8 = 0.1724$, producing a table of weights that adds to 1]

E. Recall that our goal is to locate edges. We now consider how to do this.

1. If we consider the intensity function one dimension at a time, then an ideal edge would produce a step change in intensity like the following along one or both directions:

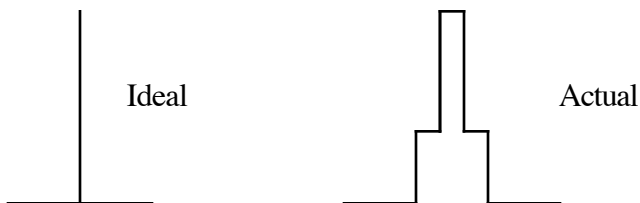


2. Actual edges, however, may look more like this (especially after averaging to eliminate noise):

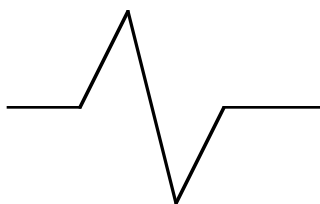


3. Consider what happens if we differentiate the averaged intensity function:

- a. The first derivative would ideally yield a "spike" at the edge; but if the edge is not a step, then it will yield a "bump" at the edge



- b. The second derivative (in the actual case) would be an S shaped curve that crosses steeply through 0 - i.e.



where the 0 crossing corresponds to the maximum value of the first derivative, which in turn corresponds to the point of maximum rate of change in the raw data - i.e. the edge

TRANSPARENCY: WINSTON FIGURE 10-9

- c. The derivatives can be approximated by a method of finite differences. For example, a simple approximation to the first derivative at a given point (where intensity is A_i) is to approximate the derivative as

$$F_i \approx ((A_{i+1} - A_i) + (A_i - A_{i-1})) / 2 = (A_{i+1} - A_{i-1}) / 2$$

(where F_i is the approximation to the first derivative at point i , and A_i is the averaged intensity at point i)

d. Likewise the second derivative can be approximated by:

$$S_i \approx ((F_{i+1} - F_i) + (F_i - F_{i-1})) / 2 = (F_{i+1} - F_{i-1})/2$$

(where S_i is the approximation to the second derivative at point i , etc)

- e. A better approximation to the two derivatives would consider several neighbors in each direction with decreasing weight in each case, but we will work with these for simplicity
- f. All three operations (averaging, first derivative, and second derivative) involve only values at the point and one or more neighbors in each direction. By substitution into the formulae above, these can be combined into one operation as follows (for the simple case of one dimension, using only one neighbor each way):

$$S_i \approx - A_{i-3}/12 - A_{i-2}/12 + A_{i-1}/12 + A_i / 6 + A_{i+1}/12 - A_{i+2}/12 - A_{i+3}/12$$

TRANSPARENCY - WINSTON FIGURE 10-10

- g. If we use a gaussian function involving more neighbors in computing the average and each of the derivatives, and if we then rotate the one-dimensional function to get a two-dimensional version (as would happen if we applied the one-dimensional function in all directions through the center point we get what is called a SOMBRERO FILTER. (Because of its resemblance to a Mexican hat.)

TRANSPARENCY - WINSTON FIGURE 10-11

- F. As before, a sombrero filter can be approximated by a weight array. For example, something like this was done for edge-detection in CS112 Project 3.

DEMONSTRATE CS112 Project 3 edge detection

Recall that this was accomplished by using a fairly simple 5 x 5 filter:

```
double [] [] filter = { { -1, -1, -1, -1, -1 },
                        { -1, 1, 1, 1, -1 },
                        { -1, 1, 8, 1, -1 },
                        { -1, 1, 1, 1, -1 },
                        { -1, -1, -1, -1, -1 }
                      };
```

- G. There is evidence that the eyes and optic nerves of living creatures perform just such a filtering operation on the raw data they process.

TRANSPARENCY - WINSTON FIGURES 10-12 and 10-13

These show a comparison between the effect of a sombrero filter and what was actually measured from a monkey being exposed to a moving stimulus (which corresponds to our computational method of scanning across the image from left to right.) (The last prediction is based on delaying the negative portion of the filter in time with respect to the positive, and gives the best fit to the actual data.)

III. Segmentation

--- -----

- A. The result of the first two steps is a line drawing, which must now be interpreted.

TRANSPARENCY - WINSTON FIGURE 3.1

1. To interpret this line drawing, we need to interpret each line appropriately. All of the following physical features can give rise to a line in the line drawing:

a. Actual physical edges

- i. A boundary between an object and the background
- ii. A convex edge in the interior of the object
- iii. A concave edge in the interior of the object

b. Pseudo-edges

- i. Cracks
- ii. Shadows

- c. Markings on the surface of the object - which we may choose to treat as real edges

2. Furthermore, portions of of an edge may be missing from the line drawing, due to occlusion by another object or poor contrast in the image.

3. A number of techniques can be used for this task. We will look at just one: the Waltz procedure.

- a. This procedure is an old one, and has some significant limitations, so it is not a complete solution to the problem.
- b. However, there are key ideas in it that remain very relevant.

- B. The Waltz procedure interprets the line drawing by assigning labels to edges.

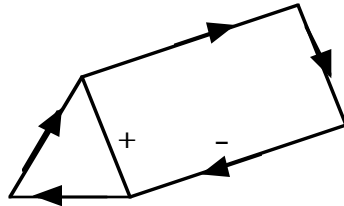
1. In all, there are 11 ways to label a given line in a line drawing:

TRANSPARENCY - WINSTON FIGURE 3-22

- a. Arrows on the boundary edges are drawn so that the object is to the right (and the background to the left) as we follow the edge in the direction of the arrows.

- b. Arrows on the concave and crack edges are drawn in accordance with the way they would be drawn if the object were "pulled apart" along the edge to convert the concave edge or crack into a boundary edge.
- c. Arrows on the shadow edges are drawn pointing into the shadow.

Example: the lines forming the object in the lower-left-hand corner of the example we just looked at (excluding the shadows) can be labelled as follows:



- d. To keep our task simple, we will limit our discussion to a subset of the possible labels by assuming that the objects we are working with have no cracks or shadows. Further, we will not try to distinguish different kinds of concave edge.
 - i. This gives us four basic labels:



- ii. The procedure we discuss can easily be extended to handle all 11 labels - but it would take a computer to keep track of it!

- 2. In labelling a line drawing, purely local computation is not sufficient. In some cases, the correct interpretation of a given line may depend on information from a line far away from it in the drawing.
 - a. Example: Project figure 3-17 in Winston viewed through a window.

There are two interpretations that we can accept: either the upper points can be "out" or the lower points can be "out".

But when we see the whole picture, only the latter interpretation is possible. (Project without window)
 - b. Waltz's procedure provides a way of taking global factors into account in a controlled fashion.
- 3. Waltz's procedure transforms the problem from one of labelling lines to one of labelling junctions between lines. A complete junction specification includes its shape plus the labelling on each of the lines comprising it; thus, a solution to the junction labelling problem is also a solution to the line labelling problem.

C. Waltz's procedure relies on two physical constraints to make the problem tractable:

1. Though many combinations of line labellings at a junction are combinatorially possible, only certain combinations are physically possible.

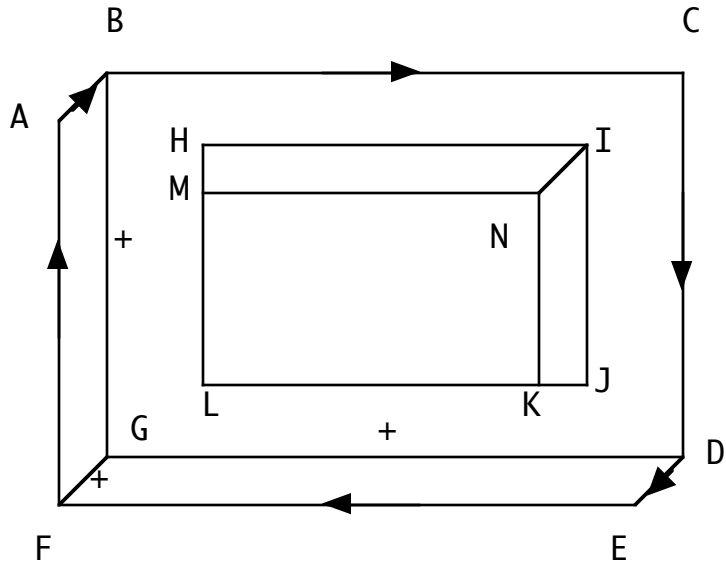
Ex: Winston figure 3-14 gives the possible set of junction labellings in a restricted environment where we impose the following limitations:

- No shadows or cracks are allowed; only real edges. (Thus, only the four line labels we mentioned earlier are needed.)
- All junctions are formed from at most three faces. (The pyramids of Egypt are ruled out.)
- The viewing angle is not singular; what we see would not be drastically altered by a slight movement of position.

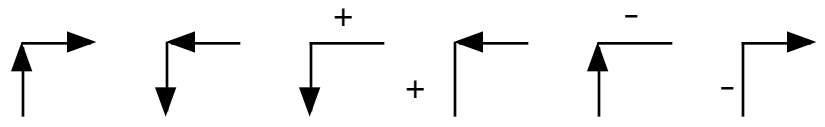
(These classes of junctions are called, respectively, L's, forks, T's, and arrows - another write uses the term V's, Y's, T's, and W's.)

Note that real vision systems must deal with a much more complex set of conditions, including shadows, cracks, and junctions formed from more than three faces. The set of possible labellings of junctions would be very hard to enumerate by hand, but has been done by computation.

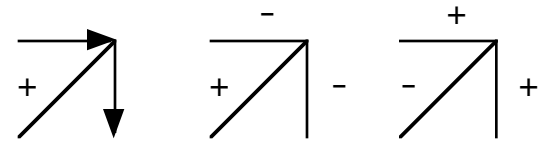
2. Each line connects two junctions. The labellings at the two junctions must both assign the same label to the line.
3. The problem of finding a set of consistent line labellings could be viewed as a search problem, with a particular assignment of labels being a state. In this case, the goal would be to find a physically consistent set. However, the size of the search space would be very large - e.g. with just the 4 labels and 10 lines, we would have 4^{10} (> 1 million) states to consider; and with 20 lines, we would have over a trillion.
4. Waltz's procedure dramatically reduces the search space size by using CONSTRAINT PROPAGATION.
 - a. At any given time in the procedure, each junction is either unvisited, or else it has associated with it a set of possible labellings. (Ultimately, the set associated with each junction will become a singleton.)
 - b. Initially, we associate with a junction the set of all possible labellings for junctions of that type (L, fork, arrow etc.) which are consistent with the labellings on its already-visited neighbors.
 - c. Constraints propagated from neighbors allow us to eliminate certain elements from the set of possible labels for the junction, until we are (hopefully) eventually left with just one.



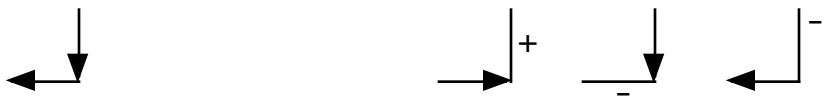
4. This finishes the outer edges. We now must plunge into the interior. If we start with H, all 6 "L" labellings are initially possible.



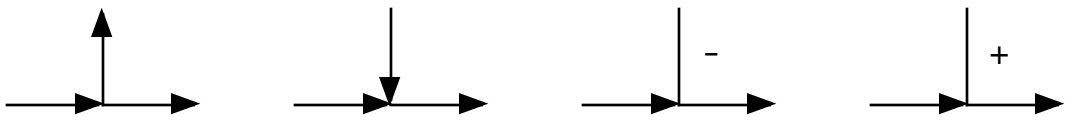
5. Likewise, for I all 3 arrow labellings are initially possible, since each is consistent with some labelling for H.



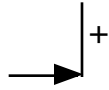
6. At J, we can eliminate 2 of the 6 "L" labellings as inconsistent with the set of labellings for I. (There is no labelling for I that has a boundary edge going from J to I.) That leaves the following possibilities:



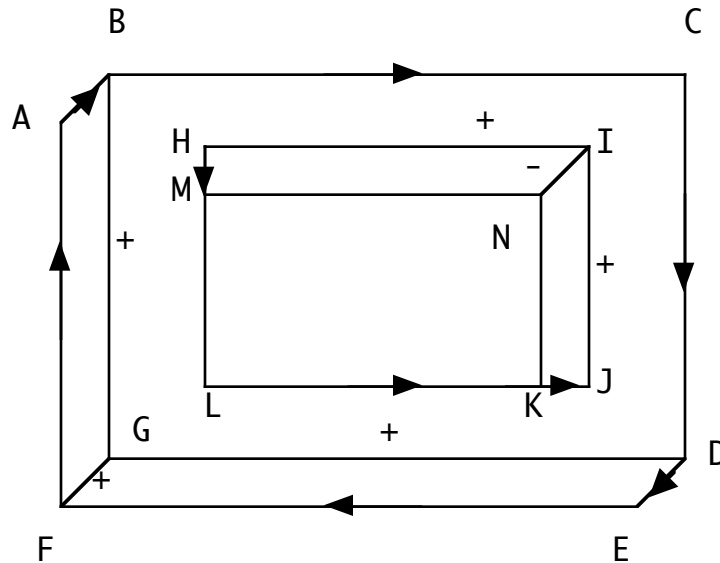
7. At K all 4 T's are possible



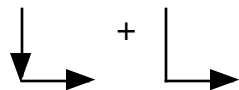
But now, we can propagate some constraint back to J. The two L's having a boundary from J to K are out, as is the L having the J-K edge a -, since K constrains the J K edge to be a K to J boundary. Thus, the labelling for J has been fixed as:



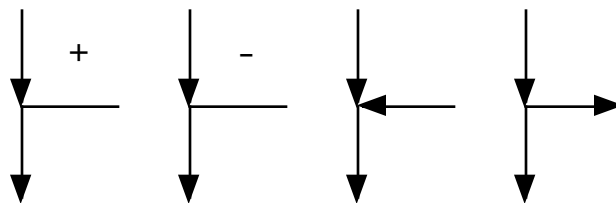
Further, the fixing of J's labelling forces the labelling for I, which in turn forces the labelling for H, leading to:



8. At L, only two labellings are possible consistent with K:



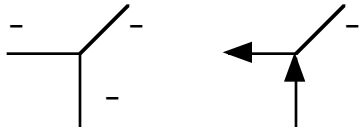
9. At M, all four "T"'s are initially possible:



However, since all of M's labellings have ML a boundary edge, this eliminates one possibility for L, forcing the remaining one as the only possibility.

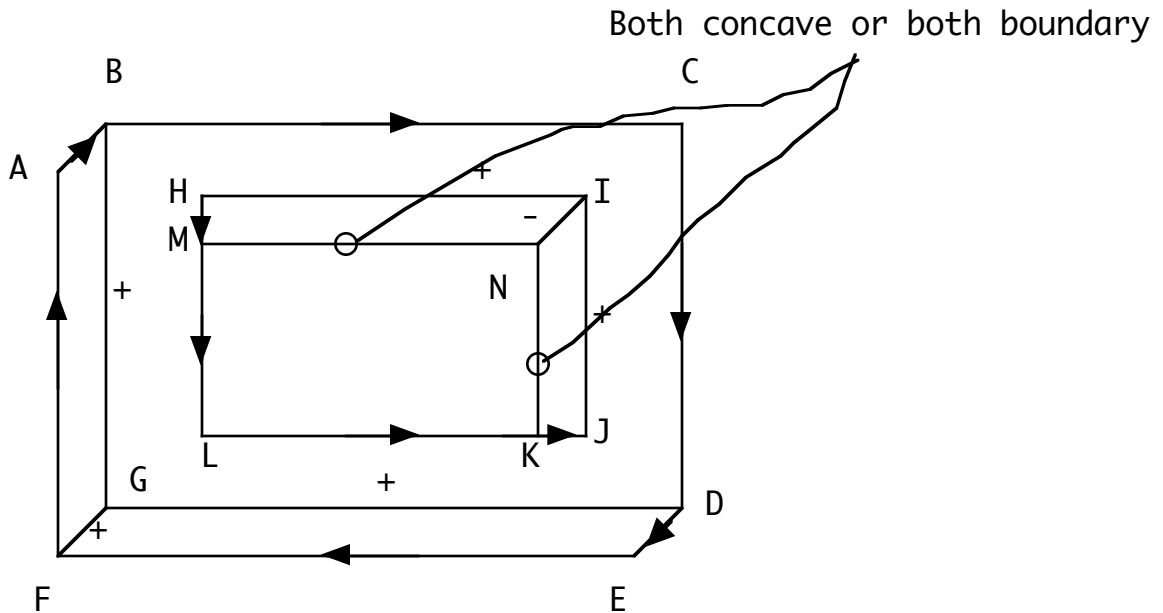


10. Finally, at N only two labellings are consistent with I:



This, in turn, reduces possible labellings of K and M to two each.

11. At this point, our labelling looks like this:



we can go no further. The Waltz procedure has uncovered a genuine ambiguity - is the center area (bounded by K,L,M,N) a bottom or a hole? Further data would be needed to settle this. (E.g. if this area were a different color or shading from the background, it's probably a bottom; if the same as the background, probably a hole.)

KLMN

12. Notice, then, that a problem that could have led to combinatorial explosion (18 possible labels for each of 14 junctions) has been converted to a straightforward procedure involving no real search at all.

E. A line drawing labelled using the Waltz procedure can then be analyzed into individual objects by tracing the boundary edges.

IV. Higher Level Aspects of Vision

-- ----- ----- ----- -- -----

- A. When we move to higher level steps in computer vision, our task becomes much more complex. We can mention just a few issues and techniques.
- B. One key task is to reconstruct the original 3D objects from a 2D image.
 - 1. A technique like the Waltz procedure can determine whether given edges are convex or concave.
 - 2. Variations in brightness are a key tool.

Example: viewed head-on, a sphere and the end of a cylinder both look like circles. But the brightness of the end of a cylinder would be uniform, while the brightness of the sphere would vary due to differences of surface direction. If the source of light were directly behind the observer, then the center of the sphere would be brighter than the edges, because the center would reflect most incipient light back to the observer while the edges would reflect some in other directions.
 - 3. Perspective can also help. Converging lines (that are assumed to be parallel in the actual object) imply increasing distance from the viewer. However, this relies on domain knowledge to tell us when to expect that a given set of lines are actually parallel in the physical scene we are studying.
 - 4. Multiple images can also provide 3D information if taken of the same scene from different angles. However, here we must deal with the CORRESPONDENCE PROBLEM - determining which components in one image correspond to given components in the other.
- C. An even harder problem is to analyze multiple images of the same object to recover information about motion, etc.
- D. In practical vision systems, the higher level problem is made tractable by a set of domain-specific expectations. For example, a vision system that is doing inspection of parts coming off a production line relies on the fact that the objects it is seeing are expected to have a certain shape.
- E. Because much of higher level vision relies on domain knowledge, the general vision problem has been called AI-Complete - solving it is tantamount to solving the whole problem of AI! [Notice that it is like natural language processing in this regard - two things which humans and even non-human creatures do intuitively prove to be very challenging problems for AI!]