Mid-level vision: involves, among other things, the determination of

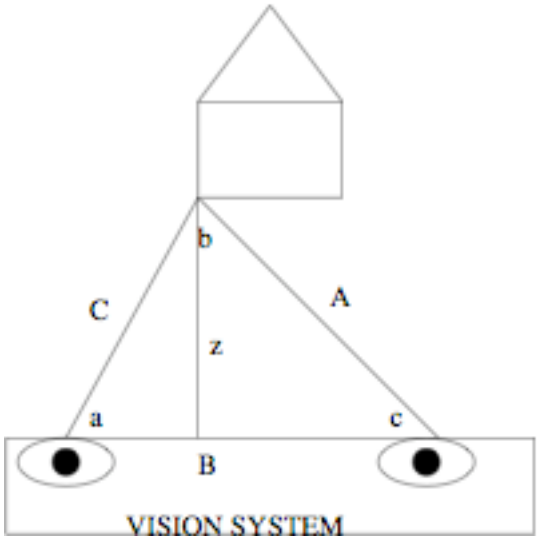
- Distance of objects
- Orientation of objects

Determining distance to an object

There are many techniques for determining distance to an object. One of these is inspired by our own visual system.

Stereo Vision

Say that we have two cameras, as in the figure below. (The cameras are the objects that look somewhat like eyes.) If we focus both cameras on the same part of an object, we can use simple trigonometric techniques to help us determine the distance between the vision system and the object.



Our goal is to calculate z in the figure above. We'll assume that we know B – the distance between the cameras. We'll also assume that we can measure the angles a and c.

The key to calculating z is remembering the Sine Law:

$$\frac{\sin a}{A} = \frac{\sin b}{B} = \frac{\sin c}{C}$$

So in the image above,

$$\frac{C}{\sin c} = \frac{B}{\sin (180 - a - c)}$$

Therefore, C = B sin c / [sin (180 – a – c)]

Since we know the values of B, a, and c, we can calculate C.

Now we can use the fact that

sin a = z / C

and so z = C sin a

Putting this all together:

z = C sin a =

$$\frac{\mathbf{B\ (\sin c)\ \sin a}}{\mathbf{\sin\ (180 - a - c)}}$$

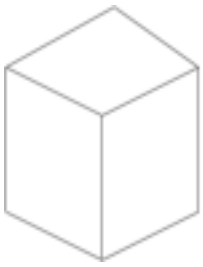
The trickiest part of this is determining the correspondence between a point in one image and a point in a second. (i.e., determining that both cameras are truly focused on the same point on a distant object). One heuristic is to focus on edges/corners.

Determining the orientation of objects in an image

Again, there are many techniques for doing this. We'll consider just one.

Waltz's Algorithm

This algorithm makes the assumption that the world is made up of *polyhedra*. For example:

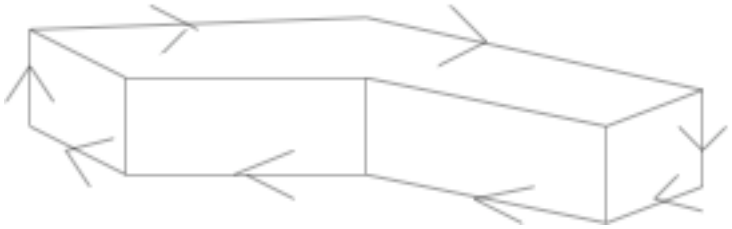


It begins with a line drawing (such as one that would be output by an edge detector). The algorithm labels lines as boundary edges or interior edges. Interior edges can be either convex or concave.

So we have:

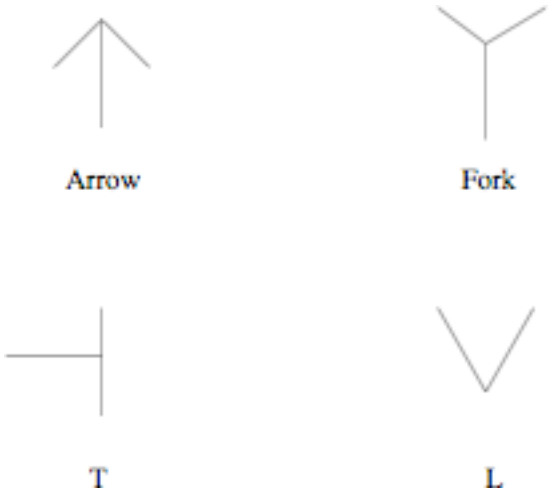
Boundary edges (>)
Interior edges
    Convex (+)
    Concave (-)

To label boundary edges (i.e., edges that separate an object from the background), imagine walking along the outline of the object, always keeping the object to your right:



In addition to assuming that the world is made up of polyhedra, Waltz makes the following simplifying assumption: Vertices are never more than three-faced. There are never more than three faces of a polyhedron coming together at a vertex. (So the top of a pyramid, for example, is something that Waltz's algorithm can't handle.)

Vertices in an image, then, can look like any of the following:



Each line can be labeled in one of four ways: >, <, +, -
This would mean that there are 4^3 possible arrows, 4^3 possible forks, 4^3 Ts and 4^2 Ls, for a total of 208 possible vertex types.

But given the assumptions of a world made up of polyhedra as well as other contraints imposed by the real world, Waltz enumerates 18 true possibilities. (See handout.)

There is only one arrow, for example, with > labels; there is only one fork with + labels.

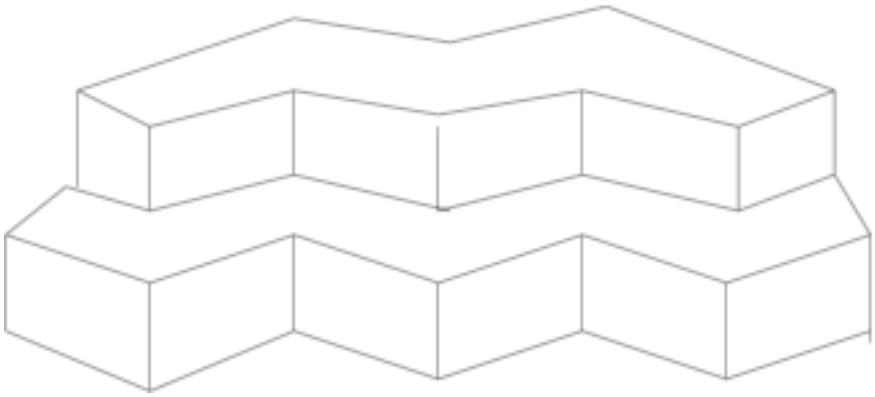
Let's consider the following (very simple) example:

The procedure for labeling can be summarized as follows:

First label the boundary edges as described above.
Then, label the lines that are forced by the boundary edges. That is, when two of three edges from a vertex are already labeled, the third edge can only take on one label.

Based on the labels on the interior edges so far (and the boundary edges), label anything that's now forced. Keep going until all edges have been labeled.

Let's consider a bigger example:



Note that the general technique used in Waltz's Algorithm is called Constraint Propagation.