Last lecture when I discussed defocus blur and disparities, I said very little about neural computation. Instead I discussed how blur and disparity are related to each other and to depth – in particular, how blur and disparity vary with accommodation and vergence.

Today I will discuss other sources of image information – called *cues* – namely perspective, texture, and shading. I will briefly describe the information available to the visual system and what problems the visual system is solving when estimating scene depth using these cues. One general idea is that we've consider so far only *depth* of isolated points or small patches around some (X, Y, Z). But our perception of the visual world doesn't represent the world as a set of points or small patches. Rather we group patches together into large surfaces. We don't just perceive depths of points and patches, but rather we perceive the *layout* of scenes – the slants of large surfaces such as a ground plane or wall. We also perceive the 3D *shap*e of objects and whether certain parts of objects are concave or convex and how these local parts of an object fit together.

Perspective and vanishing points

You are all familar with the fact that parallel lines in the world might not appear parallel; when you look at them. The two rails of a train track or the two sides of a road will meet at the horizon, for example. Note that the lines don't actually meet; they only meet "in the limit". The image point where such parallel lines meet is called a vanishing point.

To say that parallel lines meet at infinity just means that if we take two parallel lines and consider where they strike some constant $Z = Z_0$ plane, then the XY distance between the points where the lines intersect a $Z = Z_0$ plane will *not* depend on depth (since the lines are parallel). However, when you project the two points at depth Z_0 into the image, the xy image distance between them will fall as $\frac{1}{Z_0}$.

Although we are most familiar with vanishing points that are defined by lines that lie in a plane such as the above examples, a vanishing points in fact is defined by any set of parallel lines. Consider a hallway, for example. The floor and ceiling and tops of the door frames will all be parallel lines but these lines lie in multiple depth planes. Similarly, the vertical lines in the door frames on the two sides of the hallway lie in different planes.

Images of manmade environments typically have three vanishing points, which correpond to vertical (gravity) and the natural axes of the two "floor plan" dimensions of the buildings or rooms in the environment. If one or two of these axes is perpendicular to the camera/eye axis, then the lines in these directions will be parallel in the image and will meet only at infinity in the image. In the slides I give an example of the McConnell Engineering Build at McGill and I indicate three sets of parallel lines in three consecutive slides. The third set of lines is parallel to the scene gravity axis which is roughly perpendicular to the camera Z axis and so its vanishing point is well outside the image frame.

Vanishing points provide strong cues about 3D. But what information do they convey? Vanishing points allow us to assign a Z component (or extent) to image lines and edges. Detecting a line or edge in an image only identifies (x, y) coordinates in visual direction, but it doesn't tell us about the Z component. By associating an line or edge with a vanishing point, we can identify a slope of that line or edge in depth. I'll have more to say about depth slope soon.

What is the computational problem that the visual system needs to solve when identifying vanishing points? Vanishing points aren't given, but rather they must be found. If a scene contains multiple sets of parallel 3D lines, then the visual system must form groups of these lines and

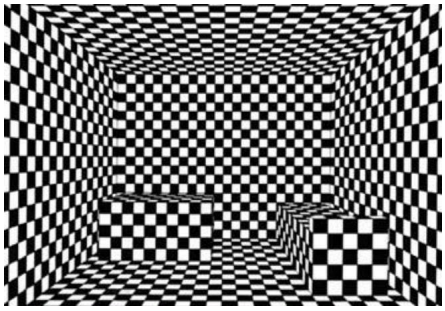
edges, corresponding to different vanishing points. There is a chicken-and-egg problem here. The visual system cannot decide whether a line or edge corresponds to a vanishing point without first knowing where the vanishing point is. And it cannot decide where a vanishing point is, unless it has identified a set of lines or edges that correspond to a common (unknown) vanishing point. Many computational models have been proposed for solving this chicken and egg problem – most of these in computer vision.

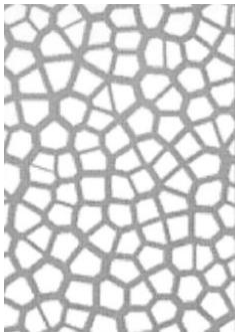
Shape from texture

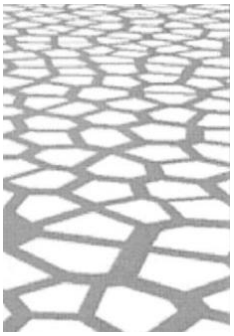
Many surfaces in the world are covered in small surface facets that have a range of sizes. Examples are grass or leaves on the ground, stones, bricks. Sometimes these texture elements are arranged in a regular pattern, for example, floor tiles.

An extreme example is the (computer graphics generated) checkerboard room shown below on the left. Such regular texture patterns on surfaces can convey rich 3D information. These scenes contain parallel lines and so they have vanishing points, but there is more information than that since the lines are *regularly spaced* in 3D and so the distances between the lines in the image varies gradually and systematically with depth. The visual system can potentially relate such gradients in image structure to 3D depth gradients.

Such gradients are defined for random textures as well, such as the examples on the middle and right. The randomness of the sizes and positions of the texture elements and the lack of vanishing points reduces the amount of information about 3D geometry that is available. Yet for those two examples, we get an impression of how depth varies across the image. In the middle panel, the surface appears to be frontoparallel (constant depth) whereas in the right panel the surface appears to slope backwards like a ground plane.







Below see a few photographs of real textures e.g. coins randomly distributed on a plane, and leaves on the ground. In each case, you have a sense of the slope of the plane. In the case of the coins which are all disks, you can use the compression of the coins to tell you something about the slope of the plane. The case with the leaves is more complicated since the leaves have a wide range of sizes.



Slant and tilt

We are considering the problem of perceiving the slope of a ground plane. The slope downward (a ceiling) or upward (a floor) or it may be to the left or right or some intermediate. Consider a general *oblique* plane which depth map

$$Z = Z_0 + AX + BY$$

where XYZ is the viewer's coordinate system which we have been using throughout the course. Note that such a plane always intersects the Z axis, namely at Z_0. (The ground plane $Y = -h$ does not satisfy this property if we are looking at the horizon.)

The *depth gradient* is the direction in 3D in which the depth of the plane is increasing fastest:

$$\nabla Z \equiv (\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y}) = (A, B).$$

The magnitude of the gradient

$$|\nabla Z| = \sqrt{(\frac{\partial Z}{\partial X})^2 + (\frac{\partial Z}{\partial Y})^2}$$

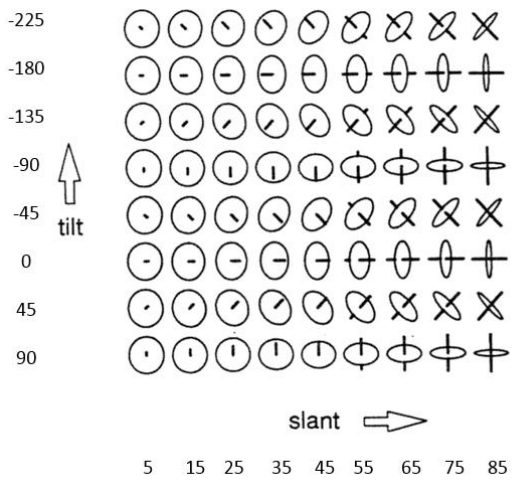
is the tangent, i.e. tan(), of the angle σ between the plane $Z = Z_0 + AX + BY$ and the constant depth $Z = Z_o$. This angle σ is called the *slant*, i.e.

$$|\nabla Z| \ = \tan \sigma.$$

We also define the *direction* of the depth gradient, which is the angle τ such that

$$\nabla Z \ = \ |\nabla Z| \ (\cos \tau, \sin \tau).$$

The angle τ is called the *tilt*. It is the angle from the viewer's X axis to the depth gradient vector $(\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y})$. Tilt τ is only defined when $|\nabla Z| > 0$ since when the plane is frontoparallel (constant Z) we cannot say in which direction it is sloped as it isn't sloped in any direction! The figure below (from Koenderink 1992) shows examples of slant and tilt.

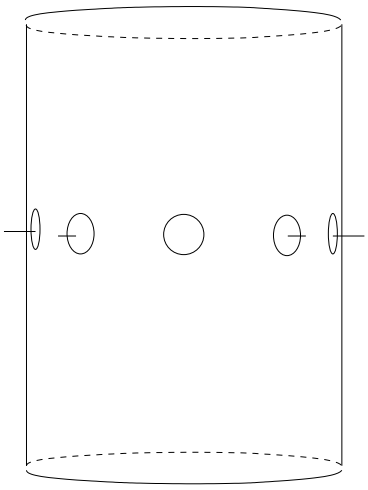


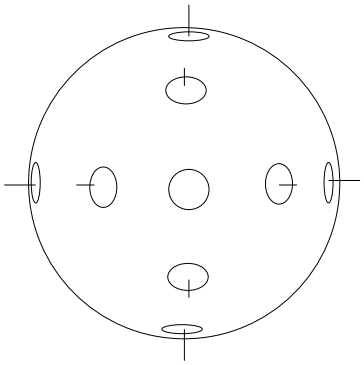
Curved surfaces

Slant and tilt are very commonly used in surface shape perception, and they seem to capture different qualitative aspects of surface orienation in space – i.e. how sloped versus which direction is the slope ? Their usage goes beyond the case of a single slanted plane though. To give you some intuition, here is an illustration of slant and tilt of a cylinder and sphere. We can also talk about the slant and tilt of points on a curved surface. Slant is the angle by which the *local tangent plane* is rotated away from a front facing disk. Tilt is the direction of this rotation.

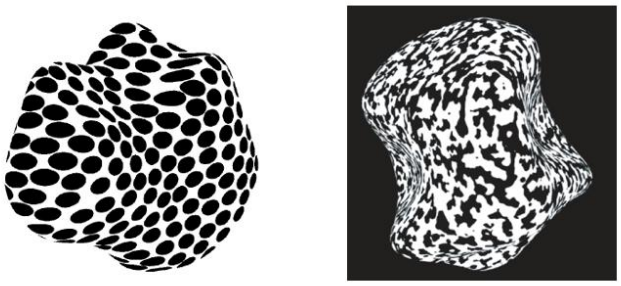
In the cylinder example below, the tilt is 0 deg for the two on the right and 180 deg for the two on the left. The tilt is not defined at the center one since the surface has zero slant there. The slants are about 80, 45, 0, 45, 80 going from left to right.

For the sphere example, the slants are close to 90 deg for four examples near the boundary. (Slants are 90 degrees always at a depth boundary of a smooth surface!) The tilts go from 0, 90, 180, 270 counter-clockwise.





Consider two surfaces rendered below[1] which are smooth random blobby shapes. The texture on the surface tells us something about the surface 3D shape. On the left, the texture elements are elongated disks. On the right, the texture elements are random blotches of various sizes. The surface orientation causes these texture elements to be compressed *in the tilt direction, and by an amount that depends on the slant*. The visual system uses this compression information to perceive shape, and this has been shown in various studies. But we don't know how this processing is done. One limitation is that the visual system does not know what the texture *on the surface* would look like if it were somehow placed on a frontoparallel plane so that there were no view-based distortion. So, the visual system cannot be sure how much of the compression that is observed *in the image* is due to the image projection, and how much of the compression is due to compression that might be present in the original texture itself. For example, on the left, the disk-like texture elements on the surface are not perfectly round disks, but rather they are already elongated i.e. *prior to projection*. This seems to be the case on the left side of the left figure, for example, as the disk texture elements there appear to be horizontally elongated on the surface (as well as in the image).



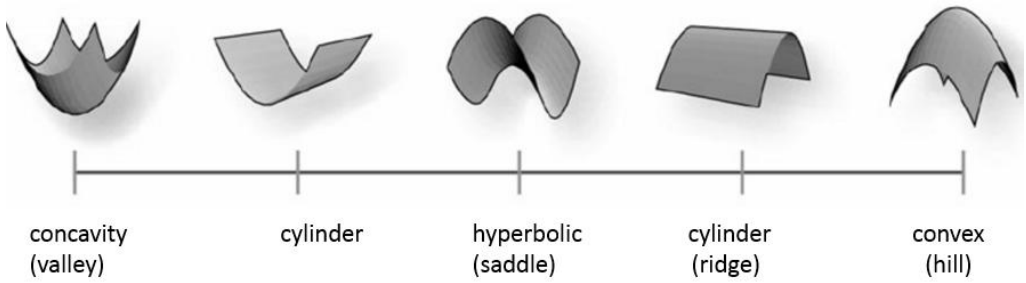
We have been discussing surface orientation (depth gradient, slant, tilt), but these properties seem inadequate for capturing our intuitions about surface shape, in particular, curvature. One can define surface curvature formally in terms of the second derivatives of the surface, and this is part of the topic of *differential geometry* which some of you who have spent more time in the math department may be familiar with. However, differential geometry isn't intended to capture what we perceive about shape and so one can ask if there is a way of (mathematically) defining local shape properties which *does* correspond to our intuitions.

One nice example of how this can be done[2] is illustrated in the figure below. The examples are local surface regions that are (from left to right) concave, an extended valley, a saddle, an extended ridge, or a convex region. We can define a continuum of shapes between the identified ones. This continuum of shapes can be defined mathematically by varying one parameter (which Koenderink and van Doorn (1992) call the *shape index*).

One parameter doesn't define the surface uniquely though, since a surface can curve in two directions at each point. The second parameter has to do with the amount of curvature; think of the scale of the surface in 3D. For the concavity case on the right, compare a golf ball to the planet earth. Both are spheres and have the same shape, but the curvature amounts are quite different.

[1] papers by Jim Todd, Roland Fleming, and colleagues

[2] Koenderink and van Doorn 1992



The slides show an example of a 3d model of a face which uses a color map to indicate the type of local shape (shape index) and the amount of curvature ("curvedness"). For example, there are just two convex spherical regions on the surface: the top of the head and the tip of the nose. Note that the curvedness is quite different in these two cases.

Surface tangent plane and unit surface normal

To define slant and tilt at different points on a general curved surface, we consider the tangent plane at each visible point. As we saw above, the tangent plane will change from point to point along the surface. For any visible point (X_p, Y_p, Z_p) on the surface, the surface in the neighborhood of that point can be approximated as a planar depth map (the *tangent plane*):

$$Z(X_p + \Delta X, Y_p + \Delta Y) = Z_p + \frac{\partial Z}{\partial X} \cdot \Delta X + \frac{\partial Z}{\partial Y} \cdot \Delta Y$$

where $(X, Y) = (X_p + \Delta X, Y_p + \Delta Y)$. Note that this is different from the equation of a plane $Z = Z_0 + AX + BY$ mentioned earlier, since (X_p, Y_p) is not necessarily $(0, 0)$. [ASIDE: I did not make this distinction originally in the slides, but I have now changed them.]

It is often useful to talk about the unit vector that is perpendicular to the local tangent plane. This is called the *local surface normal*. Let's derive what this vector is. Not surprisingly, it depends on the depth gradient of the tangent plane.

Taking Z_p from the right side to the left side of the above equation, we get

$$\Delta Z = \frac{\partial Z}{\partial X}\Delta X + \frac{\partial Z}{\partial Y}\Delta Y$$

or

$$(\Delta X, \Delta Y, \Delta Z) \cdot (\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y}, -1) = 0 \ .$$

Since this inner product relationship holds for any step $(\Delta X, \Delta Y, \Delta Z)$ in the tangent plane of the surface, it follows that the vector $(\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y}, -1)$ is perpendicular to the surface and hence this vector is in the direction of the *surface normal*. We rescale this vector to unit length, and call it as the *unit normal vector*

$$\mathbf{N} \equiv \frac{1}{\sqrt{(\frac{\partial Z}{\partial X})^2 + (\frac{\partial Z}{\partial Y})^2 + 1}} (\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y}, -1) \ .$$

Notice that we are not considering steps $(\Delta x, \Delta y)$ in the image, but rather we are considering steps on the surface, or more specifically, on the surface tangent plane.

Shape from shading

Suppose we have a surface defined by a depth map $Z(X, Y)$. We have in mind here a curved surface. If we illuminate this surface by a parallel source such as sunlight from some 3D direction $\mathbf{L}$, then the amount of light reaching a surface patch depends on the orientation of the patch. If the patch faces the light source directly, then the maximum amount of light reaches that patch. As the patch orienation is rotated away from the light source direction, the amount of light that reaches the patch falls off. Lambert's law says that the amount of light reaching the patch depends on the cosine of the angle between the normal of the patch and the direction to the light source:

$$I(X, Y) = \mathbf{N}(X, Y) \cdot \mathbf{L}.$$

Here I have normalized the intensities so that the point that faces directly to the source has value 1. Also note that there is no dependence in this model on the distance from the light source to the surface. Essentially we are assuming that the light source is very far away, like sunlight.

The computational problem of *shape from shading* then goes as follows: given an intensity image, estimate a surface $Z(X, Y)$ such that the surface normal satisfies the above model. There are a number of reasons why this problem is difficult. First, the vision system needs to estimate the light source direction. People have come up with methods for doing so, which I won't go into. Let's just assume for now some hypothetical direction $\mathbf{L}$. The second reason the problem is still difficult is that the surface normal has two degrees of freedom at each point, namely it is some direction on a unit hemisphere facing the light source. The intensity $I(X, Y)$ only specifies the angle between the normal $\mathbf{N}$ and the light source, but there are many possible $\mathbf{N}$ that have this angle.

There are other versions of the shape from shading problem, and I'll return to them next lecture. For now, let's just think about what problem we are solving here. I said that the problem was to estimate the surface normal at each point. You can imagine that, with the estimates of surface normals, you can estimate the surface depths $Z(X, Y)$ by fitting oriented patches together. You could also estimate the surface shape (see shape index on previous page) by piecing patches together. But notice that this discussion is pure handwaving. It does not say *how* to estimate the surface normals. Frankly there are far more unknown aspects of this problem than known aspects – and this is despite decades of experiments and theorizing. (This is in remarkably sharp contrast to the situation of binocular stereo, which relatively well understood.)

