Linear shape from shading (continued)

Recall the linear shading model from last class, where the l_Z component of the light source was small in comparison to $|(l_X, l_Y)|$. In order to estimate shape from shading using this linear model, we would need to know the direction of the (l_X, l_Y) vector. How might we estimate it? The technique is simple – and I'll explain later this lecture. But to understand why it works, we first need to understand the relationship between Z and its derivatives, the light source direction, and the surface irradiance gradient.

According to the linear shading model,

$$E(X,Y) \;\approx\; -l_Z + \frac{\partial Z}{\partial X} l_X + \frac{\partial Z}{\partial Y} l_Y$$

and so the gradient of irradiance along the surface is

$$(\frac{\partial E}{\partial X}, \frac{\partial E}{\partial Y}) = \begin{bmatrix} \frac{\partial^2 Z}{\partial X^2} & \frac{\partial^2 Z}{\partial X \partial Y} \\ \\ \frac{\partial^2 Z}{\partial X \partial Y} & \frac{\partial^2 Z}{\partial Y^2} \end{bmatrix} \begin{bmatrix} l_X \\ l_Y \end{bmatrix}. \tag{1}$$

The 2×2 matrix of second derivatives of Z is the Hessian $\mathbf{H}$. Let's have a closer look at the Hessian.

The Hessian is symmetric and so its eigenvalues are real and its eigenvectors $\mathbf{u}_1$ and $\mathbf{u}_2$ are orthogonal:

$$\begin{bmatrix} \mathbf{u_1} & \mathbf{u_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 Z}{\partial X^2} & \frac{\partial^2 Z}{\partial X \partial Y} \\ \\ \frac{\partial^2 Z}{\partial X \partial Y} & \frac{\partial^2 Z}{\partial Y^2} \end{bmatrix} \begin{bmatrix} \mathbf{u_1} & \mathbf{u_2} \end{bmatrix}$$

Thus the Hessian can thus be diagonalized

$$\mathbf{H} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T,$$

where $\mathbf{u}_1$ and $\mathbf{u}_2$ are the columns of $\mathbf{U}$. [Keep in mind that the $\mathbf{U}$ and $\mathbf{\Lambda}$ will vary with position $(X, Y, Z(X,Y))$ along the surface.]

The diagonal elements of $\mathbf{\Lambda}$ are the second derivatives of Z in the directions of the eigenvectors, $\lambda_i = \frac{\partial^2 Z}{\partial u_i^2}$ for $i = 1, 2$. This is perhaps not obvious. To see why it is so, take a point (X_0, Y_0) and expand $Z(X,Y)$ to second order around this point. Letting $(\Delta X, \Delta Y) = (X - X_0, Y - Y_0)$, and $(u_1, u_2) = (\Delta X, \Delta Y)\mathbf{U}$, we get

$$\begin{aligned} Z(X,Y) \;&\approx\; Z(X_0,Y_0) + \nabla Z\,|_{X_0,Y_0} \cdot (\Delta X, \Delta Y) + (\Delta X, \Delta Y)\mathbf{H} \begin{bmatrix} \Delta X \\ \Delta Y \end{bmatrix} \\ &=\; Z(X_0,Y_0) + \nabla Z\,|_{(X_0,Y_0)} \mathbf{U}\mathbf{U}^\mathbf{T} \begin{bmatrix} \Delta X \\ \Delta Y \end{bmatrix} + \frac{1}{2}(\Delta X, \Delta Y)\mathbf{U}\mathbf{U}^\mathbf{T}\mathbf{H}\mathbf{U}\mathbf{U}^\mathbf{T} \begin{bmatrix} \Delta X \\ \Delta Y \end{bmatrix} \\ &=\; Z(X_0,Y_0) + (\mathbf{U}^T \nabla Z\,|_{(X_0,Y_0)}) \cdot (u_1, u_2) + \frac{1}{2}(u_1, u_2)\mathbf{U}^\mathbf{T}\mathbf{H}\mathbf{U} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \end{aligned}$$

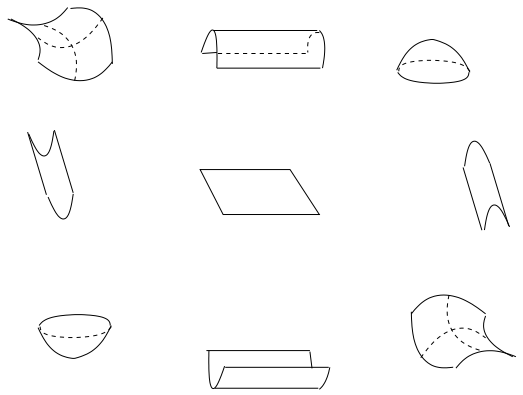
and so writing $Z(u_1, u_2)$ in a Taylor expansion about $(u_1, u_2) = (0, 0)$, we get

$$Z(u_1,u_2) = Z(0,0) + \nabla Z\,|_{(0,0)} \cdot (u_1, u_2) + \frac{1}{2}(u_1, u_2)\mathbf{\Lambda} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

from which it follows immediately that

$$\mathbf{\Lambda} = \begin{bmatrix} \frac{\partial^2 Z}{\partial u_1^2} & 0 \\ 0 & \frac{\partial^2 Z}{\partial u_2^2} \end{bmatrix}.$$

The eigenvalues λ_1 and λ_2 can be either negative, positive, or zero. The figure below illustrates the various combinations – think of it as a 2D shape space which is defined by the two eigenvalues λ_1 and λ_2. Cylinders occur when one eigenvalue is zero: a cylindrical hill occurs when the non-zero eigenvalue is positive, and a cylindrical valley occurs when the non-zero eigenvalue is negative. Saddle points occur when the two eigenvalues are of opposite sign (top left and bottom right). A hill (top right) or valley (bottom left) occurs when the eigenvalues are of the same sign.



The shape space just defined is closely related to the local *curvature* of the surface. Those of you who will continue on to Siddiqi's course in the Winter semester will learn more about this.

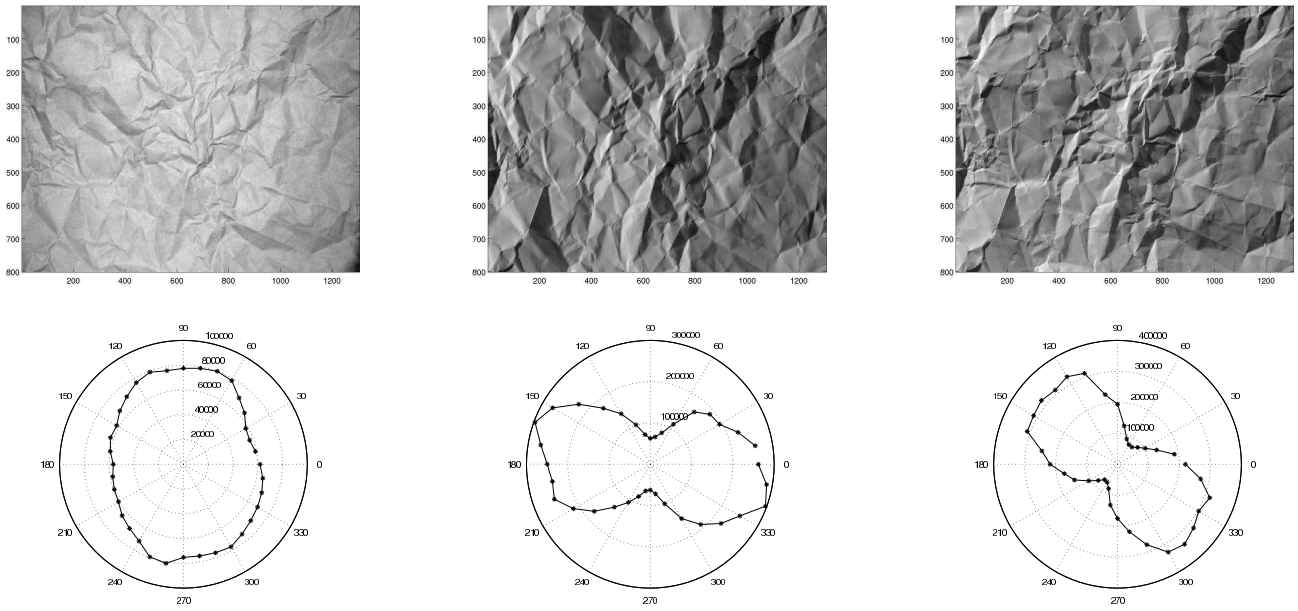
Estimating the light source direction (l_X, l_Y)

Eq. (1) implies that there is a relationship between the irradiance gradient, the vector (l_X, l_Y), and the eigenvectors and eigenvalues of the Hessian $\mathbf{H}$. In Exercise 2, I plan to ask you to work out some of these relationsships. For now, let's just note what happens when we take a surface that has relatively small gradients $|\nabla Z|$ and we illuminate it from various directions. The figure belows shows three images of piece of an approximately Lambertian white paper that has been crumpled up and then uncrumpled so that it nearly flat. The paper is viewed under perspective from about one meter. In each image, the light source is positioned in the same $Z = 0$ plane as the camera. In the leftmost image, the light source is directly "above" the camera (namely $Y > 0$). In the middle image the light source is off the left of the camera ($X < 0, Y \approx 0$). In the rightmost image, the lightsource is to the left and above the camera.

For each image, I have computed the image intensity gradient at each point[1], then computed the direction θ of the gradient, have summed up the gradient magnitudes within small theta bins, and then plotted using Matlab's `polar` command.

[1]To do this experiments properly, I really should have undone the camera's non-linear response. Instead I am just taking the raw image intensities.

Notice how the image gradients first image in the first image are much more uniformly distributed over direction than are the second and third. The reason is that the l_X, l_Y components are much less in the first image, and so the l_Z component (quadratic shading) plays more of a role. For the second and third images, there is a dominant gradient orientation. This points roughly in the direction of (and directly away from) the (l_X, l_Y) component. The reason why this effect occurs will be clear once you work through the Exercise question.



just above camera from left from top left

Shape from shading on a cloudy day

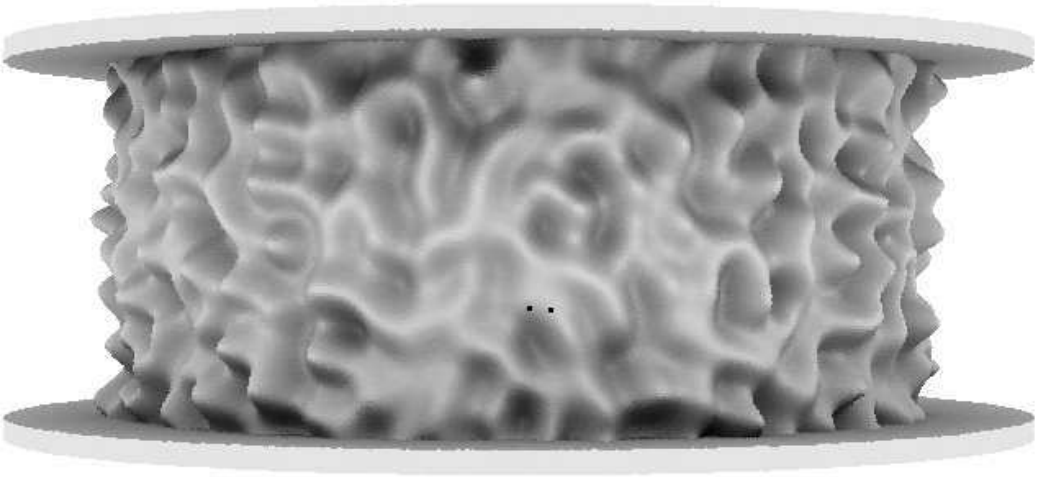
Let's now turn to a different shading problem, which I introduced in my PhD thesis and presented at ICCV in 1993[2]. Again suppose we have a Lambertian surface of uniform reflectance. Now, however, rather than having a unique light source direction as on a sunny day, we instead have a diffuse light source such as on a cloudy day. We assume that the radiance from the sky is constant L_{src} and is much greater than the radiance coming from other surfaces in the scene. So we ignore the radiance from surfaces in the scene, and approximate the surface irradiance as

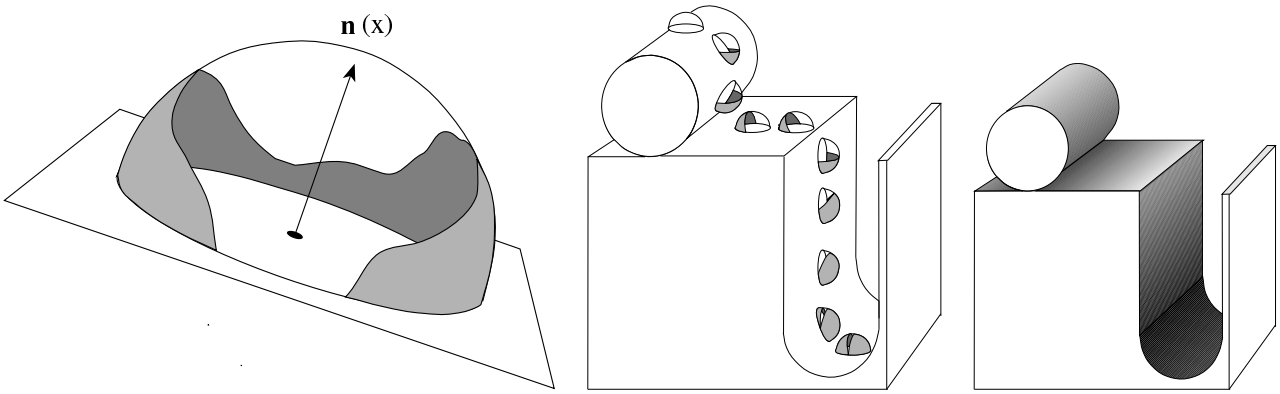
$$E(X,Y) \;=\; L_{src} \int_{\mathbf{l} \in \mathcal{V}(\mathcal{X},\mathcal{Y})} \mathbf{n}(\mathbf{x}) \cdot \mathbf{l} \; d\Omega$$

where $\mathcal{V}(X,Y)$ is the set of directions in which the sky is visible from $(X, Y, Z(X,Y))$. Notice that the irradiance now depends both on the surface normal variations as well as on the amount of sky that is visible.

[2]See a longer version of this paper, "Shape from shading on a cloudy day," which appeared in the Journal of the Optical Society of America in 1994

Below is an image of a bumpy surface rendered with computer graphics, under the diffuse lighting model just given. There are noticable intensity variations on the surface which are due to the hill vs. valley shapes. Below are three sketches illustrating where the shading comes from. The hemispheres in the two sketches on the left are partitioned into two regions – a solid color which represents directions in which the sky is *not* visible, and the remainder which are directions in which the sky is visible. Notice that with the example of the cylinder lying on the ground, the points on the of the cylinder sees the entire sky, but as you move toward the contact between the cylinder and the ground, there is less light arriving at the surface and so it is darker.





Having a model of shading is just the first step. We would like to come up with an algorithm that estimates the surface shape from such shading. The first algorithm I came up was based on an approximation of the above lighting model, in which the surface irradiance only depends on the solid angle of the sky that is visible. To obtain such an approximation, one can replace the factor $\mathbf{n} \cdot \mathbf{l}$ by $\frac{1}{2}$ which is its average value over the hemisphere[3], giving:

$$E(X,Y) \approx \frac{L_{src}}{2} \int_{\mathbf{l} \in \mathcal{V}(\mathcal{X},\mathcal{Y})} d\Omega. \tag{2}$$

[3]You can calculate this using spherical coordinates

The integral just gives us the solid angle of the sky that is visible, and hence *we model the surface irradiance as some constant times the solid angle of the visible sky.*

The task now is to come up with an algorithm for computing surface shape, given the surface irradiance function.[4] This shape from shading problem is challenging to solve because the visibility of the sky is not a *local* property of the surface shape (like the surface normal) but rather it is a *global* property. In particular, the sky visibilty $\mathcal{V}(X,Y)$ at a surface point $(X,Y,Z(X,Y))$ can be affected by scene points that are far away, which can "cast shadows" i.e. block the sky.

The insight I had for an algorithm for solving this shape from shading problem was to think of the visibility function not just on the surface but also in the 3D space above the surface. In particular, when the visibility function $\mathcal{V}(X,Y,Z)$ is defined in 3D space, it has strong local constraints, which I called *local visibility constraints*: If the sky is visible from some point (X,Y,Z) in space and in a direction $\mathbf{l}$, then the sky also will be visible from the nearby point $(X,Y,Z)+r\mathbf{l}$ and in direction $\mathbf{l}$. The algorithm for computing shape from shading that I came up with involved computing this visibility field in the free space above the surface, using the local visibility constraints.

The algorithm can be explained intuitively as follows. Imagine a surface with height function $Z(X,Y)$. Suppose we were to flood the surface with water so that all points on the surface were covered with water. Then, we drop down on the surface of the water a square grid of water spiders at locations (X,Y), which correspond to the image pixels (under weak perspective). Using the approximate model that irradiance is proportional to the solid angle of the visible sky, we can tell each water spider the following: *when the water is drained away and you land on the surface below, you will see a certain total solid angle of the sky.*

The algorithm proceeds depth by depth, analogous to draining away the water. For each depth $Z=k$ and for each point (X,Y,k) at that depth, we use the local visibility constraints to compute the set of directions in which the sky is visible. If for any X,Y the solid angle of the sky decreases to the given solid angle (what the water spider is told at the beginning), then the water spider knows that it has reached the ground and it stops. After it stops, it is able to block the sky from other water spiders that have not yet stopped.

How does the water spider compute how much of the sky is visible? The idea is to use the local visibility constraint mentioned earlier. Suppose the water spider at (X,Y,Z) wishes to know if the sky is visible in direction $\mathbf{l}=(l_X,l_Y,l_Z)$. It can then ask the nearest neigbhoring water spider in direction (l_X,l_Y) whether the sky was visible in direction $\mathbf{l}$ earlier when that water spider passed through a point on the ray $(X,Y,Z)+r\mathbf{l}$. This way, *local visibilities are used to solve a global problem.*

There are various technical details required to get this algorithm to work, e.g. I assume that outside the (X,Y) range of the image, the surface is flat, and so the surface $Z(X,Y)$ has been excavated from the ground – it lies no higher than the (assumed) flat ground outside the field of view.

[4] In fact, we are given the image irradiance, not surface irradiance. But these two quantities are closely related for a Lambertian surface, seen under weak perspective – recall lecture 15, page 1.

```
\\ Algorithm (Shape from shading on a cloudy day)
\\
\\ Given an estimate Omega*(X,Y) of the solid angle
\\ of the sky that seen from each unknown surface
\\ point  (X,Y, Z(X,Y)), compute the depth Z(X,Y).

for all (X,Y),
   Z(x,y) := 0
   for all k <= 0, V_k(X,Y) := hemisphere of directions
k := 0
repeat
  forall (X,Y)
     if Z(X,Y) = k
        Compute V(X,Y,k) using the Local Visibility Constraints
        Compute Omega(X,Y,k) from V_k(X,Y,k)
        if Omega(x,y,k) > Omega*(X,Y,k)
          Z(X,Y)++
  k++
until Omega(X,Y,k) <= Omega*(X,Y,k)  for all (X,Y),
```

Subsequent work...

In class I discuss a few other shape from shading papers that I wrote after the original one. You will not be held responsible for these subsequent works. If you are interested, though, then you can have a look at at the following papers which are (also) available on my home page:

- Towards Accurate Recovery of Shape from Shading Under Diffuse Lighting, A.J. Stewart and M.S. Langer, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 411-418. San Fran., CA. (1996).
- A prior for global convexity in local shape from shading, M.S. Langer and H. H. Buelthoff, Perception. 30(4):403-410, 2001.
- Depth discrimination from shading under diffuse lighting, M.S. Langer and H. H. Buelthoff, Perception.29 (6) 649-660, 2000.