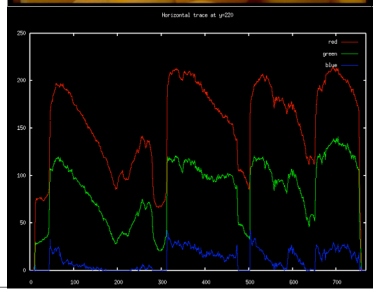
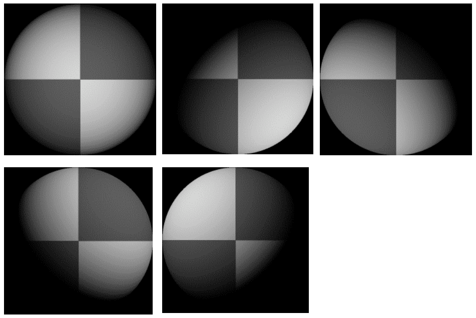
Shape from shading



Photometric stereo example



Photometric Stereo

Thus combining the conditions given by each light, i, we get

$\mathbf{i} = V\mathbf{g}$

Where the i^{th} element of **i** is $I_i(x,y)$ and the i^{th} row of V is V_i

Since g has three elements, we need at least 3 lights.

If the number of lights is more than than 3, then use least squares!

You should understand the construction of this problem.

Dealing with shadows

Each point is in K images (one for each light)

If $I_i(x,y)$ is in shadow, then ignore it.

As in the book, we can simplify this in a program by multiplying both sides by a diagonal matrix with the image intensities on the diagonal.

Dealing with shadows

$$\begin{pmatrix} I_1^2(x,y)\\ I_2^2(x,y)\\ ..\\ I_n^2(x,y)\end{pmatrix} = \begin{pmatrix} I_1(x,y) & 0 & .. & 0\\ 0 & I_2(x,y) & .. & ..\\ .. & .. & .. & 0\\ 0 & .. & 0 & I_n(x,y)\end{pmatrix}\begin{pmatrix}\mathbf{V}_1^T\\ \mathbf{V}_2^T\\ ..\\ \mathbf{V}_n^T\end{pmatrix}\mathbf{g}(x,y)$$

unknown

image intensity becomes squared

shadow => 0

known light vectors

Dealing with shadows

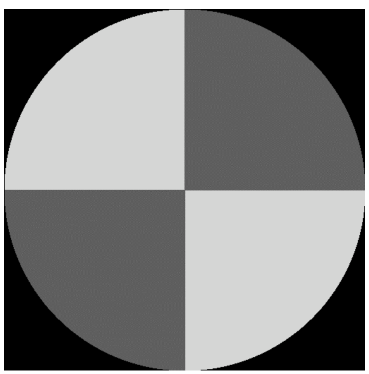
The approach on the previous slide weights the equations according to image intensity and so pixels in shadow are ignored (weight is zero).

This changes the impact of the non-zero ones also (not necessarily for the better, depending on your error model).

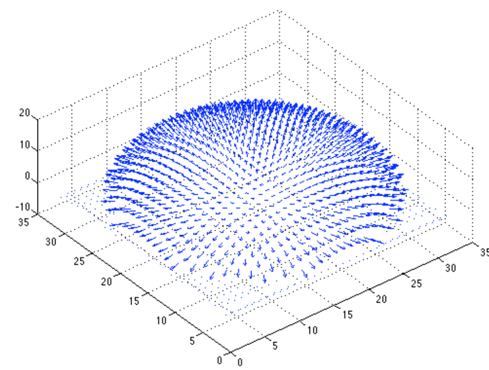
This has the advantage that you do not need a threshold for deciding how dark a pixel has to be before it is ignored. Instead, the darker they are, the less important they are.

Note that weighting rows is a good general trick for getting least squares to do what you want.

Recovered reflectance



Recovered normal field



From Normals to Shape

From **g** we can get the normal $\hat{\mathbf{n}} = \frac{\mathbf{g}}{|\mathbf{g}|}$

It is natural to represent surface as a depth map (x,y,f(x,y))

But what is the relationship between that and the normals?

From Normals to Shape

Given (*x, y, f(x,y)*), what is the surface normal direction?

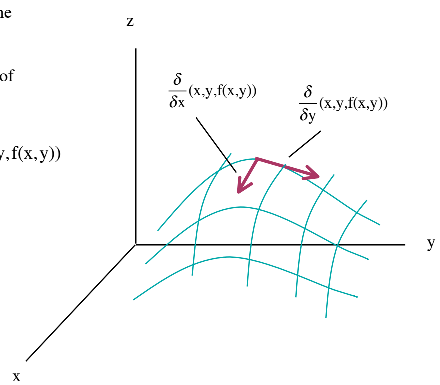
Method one (cross product)

$$\hat{\mathbf{n}} \propto \frac{\delta}{\delta x}(x,y,f(x,y)) \times \frac{\delta}{\delta y}(x,y,f(x,y))$$

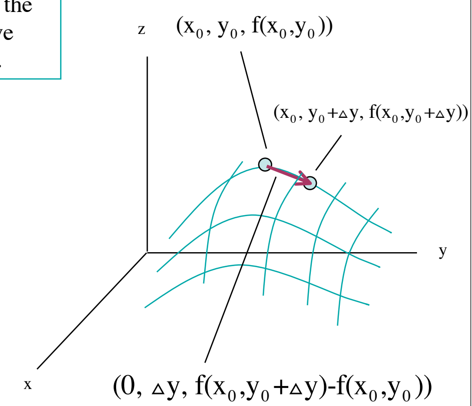
The partials in x and y give us two tangents which are vectors in the plane touching the surface.

To get a vector normal to both of them, take their cross product.

$$\hat{\mathbf{n}} \propto \frac{\delta}{\delta x}(x,y,f(x,y)) \times \frac{\delta}{\delta y}(x,y,f(x,y))$$



In case the claim that the partial derivatives give tangents is confusing.



From Normals to Shape

$$\hat{\mathbf{n}} \propto \frac{\delta}{\delta \mathrm{x}}(\mathrm{x,y,f(x,y)}) \times \frac{\delta}{\delta \mathrm{y}}(\mathrm{x,y,f(x,y)})$$

$$= (1, 0, \mathrm{f_x}) \times (0, 1, \mathrm{f_y})$$

$$= (-\mathrm{f_x}\ , -\mathrm{f_y}\ ,\ 1)$$

From Normals to Shape

Given (*x, y, f(x,y)*), what is the surface normal direction?

Method two (level curves)

Given a surface, S, specified by $g(x,y,z) = 0$
$\nabla g(x,y,z)$ is normal to S

So, find $g(x,y,z)$ such that $g(x,y,z) = 0$ is our surface

From Normals to Shape

Given (*x, y, f(x,y)*), what is the surface normal direction?

Method two (level curves)

Given a surface, S, specified by $g(x,y,z) = 0$
$\nabla g(x,y,z)$ is normal to S

So, find $g(x,y,z)$ such that $g(x,y,z) = 0$ is our surface

$$g(x,y,z) = z - f(x,y)$$

From Normals to Shape

Given (*x, y, f(x,y)*), what is the surface normal direction?

Method two (level curves)

$$g(x,y,z) = z - f(x,y)$$

$$\nabla g(x,y,z) = \quad \mathbf{?}$$

From Normals to Shape

Given (*x, y, f(x,y)*), what is the surface normal direction?

Method two (level curves)

$$g(x,y,z) = z - f(x,y)$$

$$\nabla g(x,y,z) = (-f_x,\ -f_y,\ 1)$$

From Normals to Shape

Either way, $\hat{\mathbf{n}} \propto (-f_x,\ -f_y,\ 1)$

It should be clear that $f_x = -\frac{n_x}{n_z}$ and $f_y = -\frac{n_y}{n_z}$

(In general, **n** will embody the albedo, so we must be prepared for an arbitrary scale factor in **n**---most easily dealt with using the above ratio if we want f_x and f_y).

From Normals to Shape

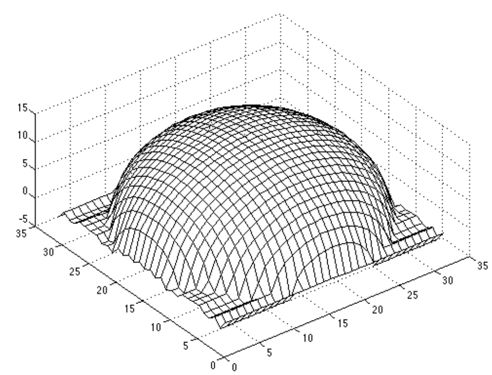
So, if have the normals, we can estimate the derivatives of f(x,y)

Minor point for those who have vector calculus: If we assume that f_x and f_y are the derivatives of a differentiable function, f(x,y) we can further check (or constrain) that $f_{xy}=f_{yx}$.

We can recover the surface height at any point by integration along some path. For example, if we declare the origin to be at height C, and go along the x axis, then parallel to the y axis:

$$f(x,y) = \int_0^x f_x(x',0)dx' + \int_0^y f_y(x,y')dy' + C$$

Surface recovered by integration



Color (very briefly)

Color is a sensation

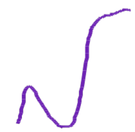
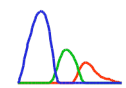
Usually there is light involved, and usually there is a relationship between the world and the colors you see

Your brain has a big effect on the colors you see

We will focus on what colors mean to a camera which is **much simpler**

Color for a camera (R,G,B) is a very limited sampling of spectral light energy (why three values?)

Recall Image Formation (Spectral)

$(\mathbf{R},\mathbf{G},\mathbf{B}) = \int_{380}^{780}$  $*$  $d\lambda$

Recall Discrete Version

Represent the light by a vector, **L**

Consider a matrix R whose rows are the discretized version of the response functions.

Let **C** be a vector of camera responses (i.e., $(R,G,B)^T$)

Then

$$\mathbf{C} = R * \mathbf{L}$$

It is common to use ρ for both camera responses (as we did earlier in the course) and albedo as we have just done so. To reduce confusion, we will switch to C for "camera".

From previous slide

$$\mathbf{C} = R * \mathbf{L}$$

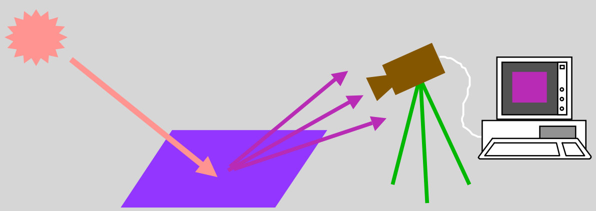
R is **not** full rank (typical values are 3 by 101 or 3 by 31)

First key observation is that you cannot recover **L** from **C** (**L** is spectra**, C** is RGB)

Second observation---many spectra can have the same RGB. These are metamers (the spectra are a metameric match).

(This is the essence of color reproduction)

(R,G,B) depends on the light, the surface, and the camera

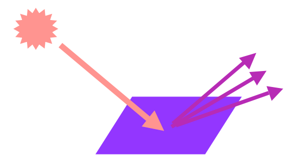


Spectral reflectance

By definition, the spectral reflectance, satisfies

$S(\lambda)=\frac{L(\lambda)}{E(\lambda)}$ where is $E(\lambda)$ incoming and $L(\lambda)$ is outgoing

So we get $L(\lambda)$ from before by : $L(\lambda)=E(\lambda)S(\lambda)$

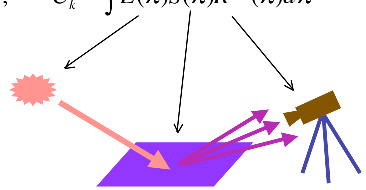


Spectral reflectance

So we get $L(\lambda)$ from before by : $L(\lambda)=E(\lambda)S(\lambda)$

Recall $C_k = \int L(\lambda)R^{(k)}(\lambda)d\lambda$

Now, $C_k = \int E(\lambda)S(\lambda)R^{(k)}(\lambda)d\lambda$



Naive Color Model

Now consider "white" light (255, 255, 255)

- This is **relative** to the camera!
- By definition, this is the color of perfect diffuse, uniform, reflector

Suppose that a surface has color (R_S, G_S, B_S) under white light

- Naively, this is the "color of the surface"
- (Naïve, because surfaces don't have color until you turn on the light, and it matters what the color of the light is!)
- The albedo in each channel is $\rho_R=\frac{R_S}{255}$ $\rho_G=\frac{G_S}{255}$ $\rho_B=\frac{B_S}{255}$

Naive Color Model (2)

Naive value for the color of the surface under a **different** light, (R_L, G_L, B_L) is given by:

$$(R,G,B) = (\rho_R R_L,\ \ \rho_G G_L,\ \ \rho_B B_L)$$

This is naïve because we assume that the part of the light that stimulates one channel, does **not** interact with the albedo of any other channel.

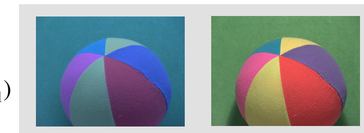
Alternatively, everything about the surface color can be captured in these 3 numbers.

This is the "diagonal model" for illumination change.

Diagonal Model for Color

(Same scene, but different illuminant)

Light color (R_{L1}, G_{L1}, B_{L1})

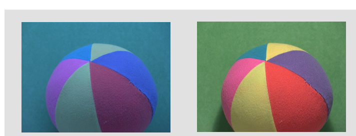


Light color (R_{L2}, G_{L2}, B_{L2})

Diagonal Model for Color

(Same scene, but different illuminant)

Light color (R_{L1}, G_{L1}, B_{L1})

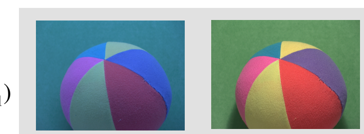


Light color (R_{L2}, G_{L2}, B_{L2})

Diagonal model assumes that all the (R,G,B) in the left image change by the ratio of the lights

$$R_2 = \frac{R_{L2}}{R_{L1}} * R_1 \qquad G_2 = \frac{G_{L2}}{G_{L1}} * G_1 \qquad B_2 = \frac{B_{L2}}{B_{L1}} * B_1$$

Light color (R_{L1}, G_{L1}, B_{L1})



Light color (R_{L2}, G_{L2}, B_{L2})

Diagonal model assumes that all the (R,G,B) in the left image change by the ratio of the lights

$$R_2 = \frac{R_{L2}}{R_{L1}} * R_1 \qquad G_2 = \frac{G_{L2}}{G_{L1}} * G_1 \qquad B_2 = \frac{B_{L2}}{B_{L1}} * B_1$$

One way to understand the above equations

Estimates of the albedos for each channel

Diagonal Model for Color

- In matrix form

$$\begin{pmatrix} R_2 \\ G_2 \\ B_2 \end{pmatrix} = \begin{vmatrix} \frac{R_{L2}}{R_{L1}} & & \\ & \frac{G_{L2}}{G_{L1}} & \\ & & \frac{B_{L2}}{B_{L1}} \end{vmatrix} \begin{pmatrix} R_1 \\ G_1 \\ B_1 \end{pmatrix}$$

- Note that this says $\frac{R_2}{R_{L2}} = \frac{R_1}{R_{L1}}$ (etc, for G, B)

(albedo estimate for the channel)

Diagonal Model for Color

- Note that this says $\frac{R_2}{R_{L2}} = \frac{R_1}{R_{L1}}$ (etc, for G, B)

- This would mean $\frac{\int E_2(\lambda)S(\lambda)R^{(k)}(\lambda)d\lambda}{\int E_2(\lambda)R^{(k)}(\lambda)d\lambda} = \frac{\int E_1(\lambda)S(\lambda)R^{(k)}(\lambda)d\lambda}{\int E_1(\lambda)R^{(k)}(\lambda)d\lambda}$!

- Or equivalently $\frac{\int E_2(\lambda)S(\lambda)R^{(k)}(\lambda)d\lambda}{\int E_1(\lambda)S(\lambda)R^{(k)}(\lambda)d\lambda} = \frac{\int E_2(\lambda)R^{(k)}(\lambda)d\lambda}{\int E_1(\lambda)R^{(k)}(\lambda)d\lambda}$! !

- But this is not generally true!

Diagonal Model for Color

In general, $\frac{\int E_2(\lambda)S(\lambda)R^{(k)}(\lambda)d\lambda}{\int E_2(\lambda)R^{(k)}(\lambda)d\lambda} \neq \frac{\int E_1(\lambda)S(\lambda)R^{(k)}(\lambda)d\lambda}{\int E_1(\lambda)R^{(k)}(\lambda)d\lambda}$

- But expression holds when
  - Surface reflectance is uniform
  - Sensors are delta functions

- Naïve approximation is relatively good when the camera sensors are “sharp” with minimal overlap.