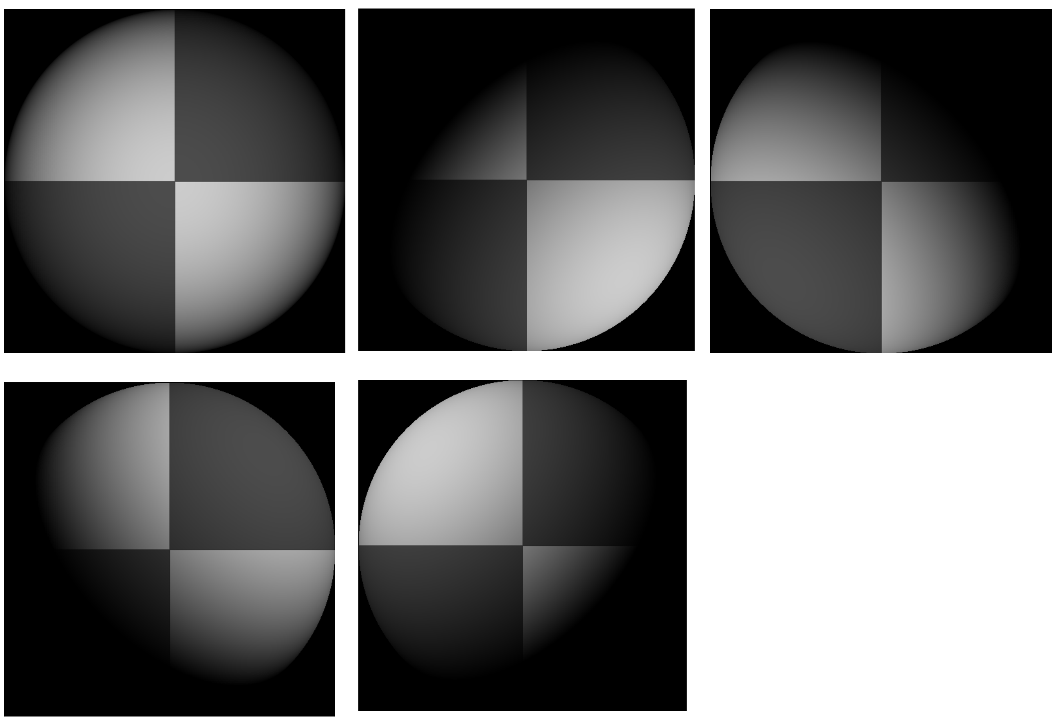
Shape from shading

- Can we find the normals at every point?
  - Under-constrained! (Only have one piece of data per pixel, but we need 2 or 3 (if we need to estimate albedo as well)).
  - We can can impose regularization (smoothness) and consider boundary conditions

- Do normals give us shape?
  - Normals are not shape, but they can be related to the partial derivatives of the shape as a function (x, y, f(x,y))
  - The partial derivatives **must satisfy integrability constraints**. Random normals do not come from a continuous surface!

Photometric Stereo

- Shape from shading is hard! Consider an easier problem.
- Suppose that we have a number of known point sources, and we have successive pictures taken with each one used in turn.



Photometric Stereo

- Shape from shading is hard! Consider an easier problem.
- Suppose that we have a number of known point sources, and we have successive pictures taken with each one used in turn.
- Let **g**(x,y) be the **unknown** surface normal times the albedo (for the point in the world corresponding to image point (x,y).
- Let $\mathbf{V}_i$ be the light source direction, i, times a scalar embodying the light source magnitude and camera sensitivity (**known**).
- Let I_i(x,y) be image intensity (**measured**).

Then $$I_i(x,y) = \mathbf{V}_i \bullet \mathbf{g}(x,y)$$ (Lambert's law)

Photometric Stereo

$$I_i(x,y) = \mathbf{V}_i \bullet \mathbf{g}(x,y) \qquad \text{(Lambert's law)}$$

So how to solve for the surface?

Simpler---how to solve for **g**(x,y)?

How many lights do we need
(assuming that albedo is not known) ?

Photometric Stereo

$$I_i(x,y) = \mathbf{V}_i \bullet \mathbf{g}(x,y) \qquad \text{(Lambert's law)}$$

Hopefully this looks like the i'th row of matrix, V, multiplied by a vector, **g**(x,y).

Photometric Stereo

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

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

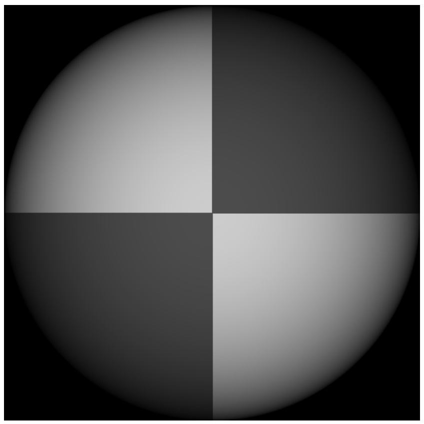
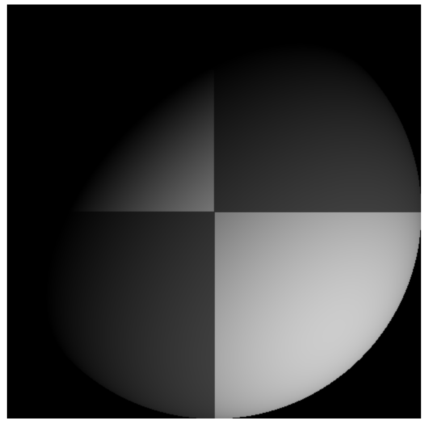
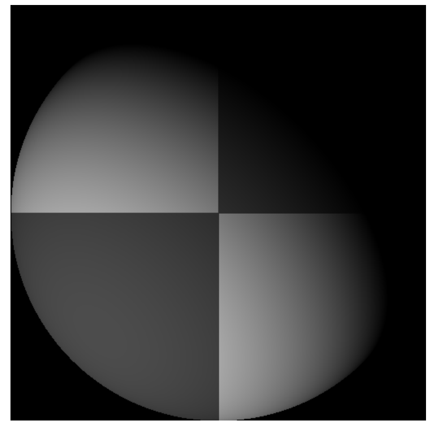
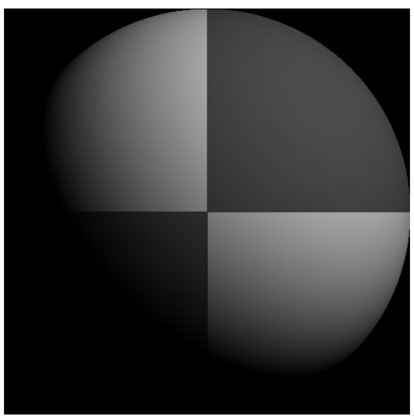
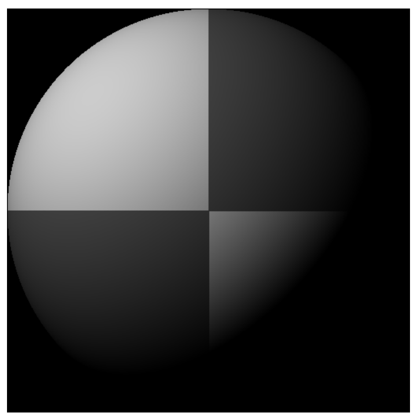
Where the i^{th} element of $\mathbf{i}$ is $I_i(x,y)$ and the i^{th} row of V is V_i

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.

Example figures

Dealing with shadows

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

If $I_i(x,y)$ is in shadow (too dark), 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.

This approach 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 (possibly for the better, depending on your error model).

Dealing with shadows

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

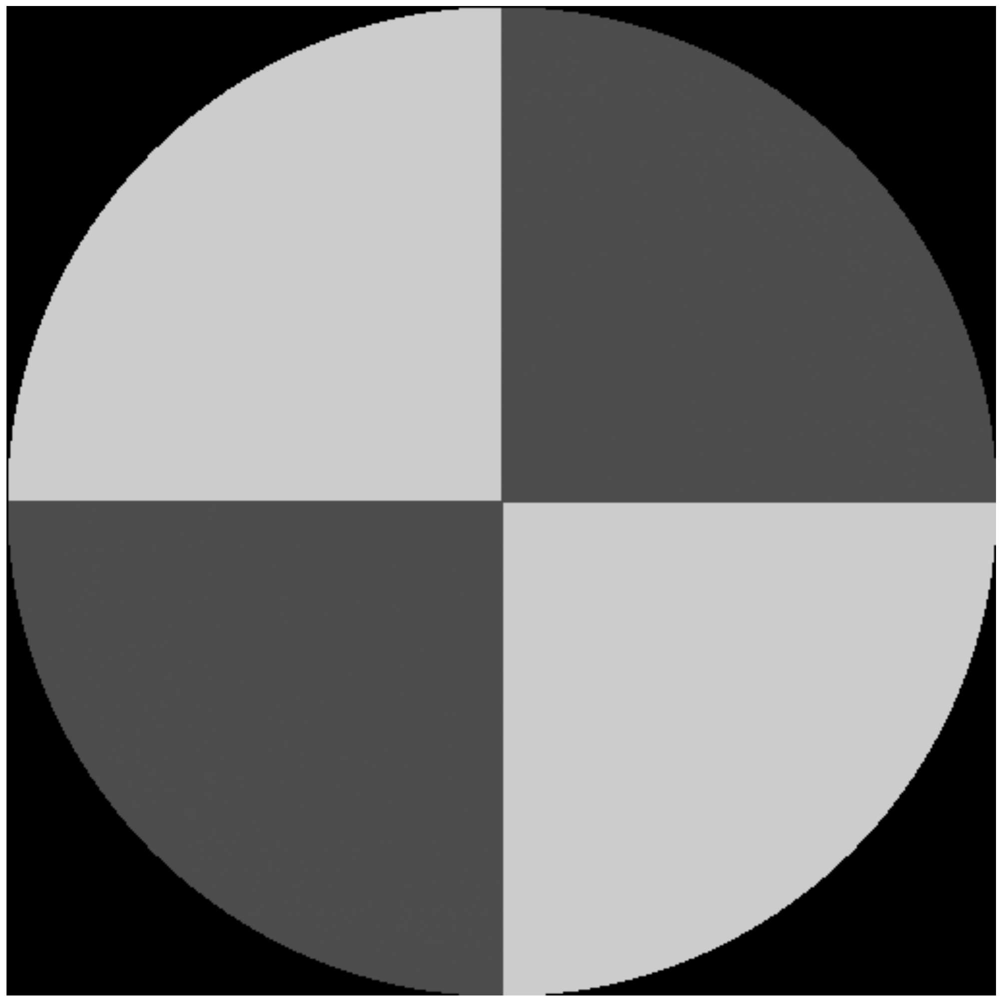
image intensity becomes squared

shadow => 0

known light vectors

unknown

Recovered reflectance



Recovered normal field