CS 280 Computer Vision — Spring 2018 — Efros, Malik, Yu — Lecture 6

1 Lecture

Radiometry

We're going to cover three aspects of image formation. The first is pure physics (the sensor elements on the camera are capturing... what?), the second is the reason for light coming into the camera, and the third is the corresponding sources/causes of light in the physical world. Essentially, we're going from the image to the physical 3D scene.

Are pixel values indicative of some physical quantity in the world? In an idealistic pinhole camera, each element traps light from a particular beam of light from the world. If we're tracing the beam back, the first surface we hit will be the thing we see on the image at that element.

To measure intensity, we have two main terms: irradiance and radiance.

Radiance

Radiance is a directional quantity. We have a infinitesimal surface area dA receiving a beam of light, where the size of the beam of light is measured by the solid angle $d\Omega$. Also for the surface area, we have a surface normal. The angle between the surface normal and the incoming beam of light is θ. Radiance is then defined as

$$L = \frac{\text{Power}}{(dA\cos\theta)\cdot(d\Omega)}$$

This describes the radiant power traveling in a given direction per unit area (measured perpendicular to the direction of travel) per unit solid angle. Note that $dA\cos\theta$ measures the effective receiving area, while $d\Omega$ measures the size of the bundle of light.

In this way, radiance ends up with the units

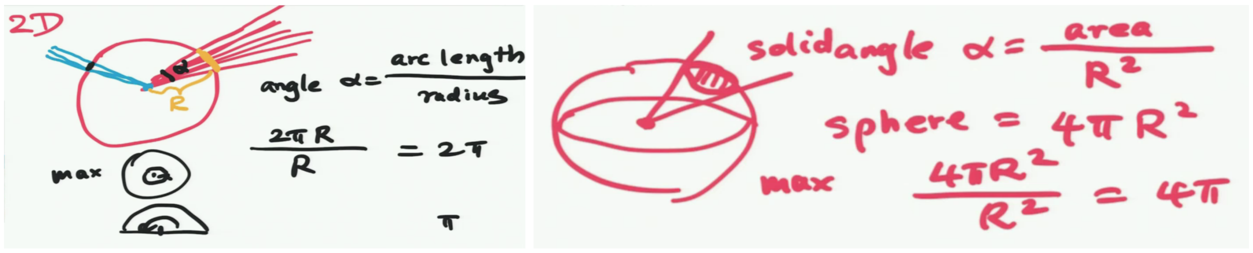
$$L = \frac{W}{m^2\cdot(sr)}$$

We can make a few observations. If the intensity of light is stronger, the radiance will be larger. If the normal points to the light surface (such that $\theta = 0$ and $\cos\theta = 1$), we will maximize the amount of light we can receive. As we tilt the normal away, the effective receiving area gets smaller and smaller. At 90°, the light will all be tangent, and won't be trapped by the receiving area; hence the effective area will be 0.

Solid angles allow us to get a sense of "narrow beam" versus "wide beam." For a ray of light, it's the angle that matters (not the distance). In 2D, we can imagine light rays coming into the center of a circle. The arc length of the bundle of rays measures the width of the beam. Thus the larger the angle, the fatter the beam. Our angles are defined (arc length)/radius, and are measured in radians.

In 3D, we have a sphere instead a circle. Therefore we now have an area instead of an arc length. Our solid angles are defined area/radius^2 (the area subtended by the object; the surface area projection onto the unit sphere), and will be measured in steradians ("ste" means "solid").

The inverse square law (if you move a candle closer to you, the effect is 4 times bigger) is automatically captured by a solid angle. If we move a receiving element closer to the light source such that R is halved, the solid angle will be four times larger.



angles and solid angles / source: Stella Yu

Irradiance

Irradiance is *not* a directional quantity; it is simply the amount of heat or intensity or photons we actually collected. More generally, it describes the light (technically radiant power, i.e. energy over time) falling upon an area.

Image irradiance is proportional to scene radiance in the direction of the camera. If radiance is "outgoing" power, irradiance is "incoming" radiance. Note that irradiance is in units of watts/m^2, while scene radiance is in units of watts/($m^2 \cdot$ solid angle).

The Cause of Outgoing Radiance

We now know that a pixel in an image captures the radiant power in a particular direction. That direction is determined by the location of the pixel in the image.

However, what causes the outgoing radiance at a scene patch? If it's not a light source, where does it get its light? This depends on the original incoming radiance from the light sources, the angle between the scene patch's normal and its incoming radiance, and the reflectance properties of the patch. (It's just ray tracing.)

When light is absorbed (i.e. not reflected, e.g. with a black surface), the surface is heated. Meanwhile, white surfaces bounce all the light back. This quality is described by the albedo ρ of a surface. Albedo 0 corresponds to black, while albedo 1 corresponds to white.

The Lambertian model is defined $L = \rho\lambda(n \cdot s)$. λ depends on the radiance of the light source, and $n \cdot s = \cos\theta$ is the foreshortening term. *Note: we often model reflectance via a combination of a Lambertian term and a specular term. More precisely we use a BRDF, a 4D function corresponding to the ratio of outgoing radiance in a particular direction to the incoming irradiance in some other direction. This originates from empirical measurements.*

Note: *edges* are areas in which the intensity changes a lot. These are important. They arise because of discontinuities in reflectance (e.g. albedo), illumination, or surface geometry (e.g. difference in normal).

Given an image, it is very difficult to produce the physics! This is the shape-from-shading (SFS) problem, where we try to go from the measured irradiance values in an image to the scene geometry, reflectances, and illumination that caused it. It is the inverse of the computer graphics rendering problem (where we want to produce the image given the scene).

Image Processing

An image (the thing that gets recorded when the light hits our retinas) can be thought of as a function f from $\mathbb{R}^2$ to $\mathbb{R}$ (mapping (x, y) positions to intensity). $f(x, y)$ should give the intensity at position (x, y). Realistically, of course, we'd expect an image to be defined over a rectangle with a finite range.

A color image is just three functions pasted together. We can write this as the vector-valued function

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

It's often useful to think of our images as surfaces. In other words, we can have x and y as normal, and then $f(x, y)$ (the intensity values) coming up out of the plane. It makes it more natural to take gradients and such.

The pixel intensity that's being stored at every point is

$$f(x, y) = \text{illumination}(x, y) * \text{reflectance}(x, y)$$

Illumination $[0, \infty]$ refers to the photons coming in from light sources. Reflectance $[0, 1]$ describes the photons that get bounced from an illuminated object. Note that there are many orders of magnitude here, since illumination is not really bounded. Unfortunately, that means we have to map a very high dynamic range world to a very low dynamic range image. If we take a long exposure, pixel value 255 really means 255 or greater; it's everything that went beyond our limit and got washed out. If we take a short exposure, we'll map the higher end of our dynamic range to $[0, 255]$ – but some of the very dim things just won't be recorded at all.

Because of the lengthy image acquisition pipeline, a camera is *not* a photometer. It's definitely not just registering photon quantities; there is a lot of processing that goes on after we get our photons. The pixel values have only a passing resemblance to the actual scene invariance; they longer represent a physical quantity. They are only some bizarre shadow of the physical quantity we really want.

However, since these are actually weird remapped pixel values (and not the ground truth values), we can play around with the values and make them more suitable for human consumption. So there is stuff to do! Broadly, this is known as **point processing**: a family of transformations that go from f to g in a global fashion, performed at every intensity independently.

$$g = T(f)$$

For example, we have the gain and bias transform

$$g(x, y) = a \cdot f(x, y) + b$$

A final note: we can **histogram** pixel intensity values over an image. For every bin (one for each possible pixel intensity), we'll count how many pixels have that value. When we plot this histogram, it can give us a sense of how much contrast the image has (how well it covers the range of pixels). The images with the nicest contrast will have a histogram that is more or less equal across the pixel intensities. *The act of enforcing such equality is known as **histogram equalization**. We can perform equalization by using the image's cumulative histogram.*