[SQUEAKING]

[RUSTLING]

[CLICKING]

PROFESSOR: Recovering the shapes of objects using image brightness measurements. And we talked at some length about photometric stereo, which was a method that gets us the result. But it's kind of unnatural in that it requires multiple exposures.

And we're about to transition into shape from shading, which is a method for doing this with a single image. And we've already talked about a few different types of surface materials and their reflecting properties. In particular, we talked about the Lambertian surfaces, and we started to talk a little bit about Hapke, which is a model for the reflection from rocky planets.

And I want to introduce a third one, which has to do with microscopy. So that's a scanning electron microscope. But first, for comparison, here's some pictures of transmission electron microscopes.

As you probably know, they allow you to achieve a huge magnification, much larger than with visible light, because you're limited by the wavelength of light. And the wavelength of electrons at kilovolt energies is very much shorter, potentially allowing you to image large molecules.

And they haven't yet achieved their limit. They theoretically would be able to resolve individual molecules and atoms, and in some cases, they do. Anyway, if you do a search, electron transmission, electron microscope, you click on Images, you get all of these pictures of these types of microscopes.

And then eventually, you get some pictures of things that might have been imaged with an electron microscope, like this thing on the right. And this is a cell with a nucleus and some vacuoles and so on. And it's an extremely useful technique for seeing fine detail, but it has its limitations in terms of interpretation.

So you look at-- well, what we will do in the moment is look at the comparison with other kinds of microscopy. So let's see. I was going to click on this thing. So I guess you can make sense of it, but it's a very thin slice of something, and you'd need to know the details of what that is.

Now if in comparison, we do a search on the scanning electron microscopes and click on Images, we get this. And you have to scroll down before you get a picture of a microscope. Why is that?

Well, because scanning electron microscopes make these great pictures that people love. The transmission electron microscope make great pictures that scientists involved in the research love, but they're not easily interpretable whereas these scanning electron microscope pictures, they're very popular. They're just fabulous.

I mean, look at that. I mean, you can immediately see the head of this jumping spider, and it's six of its eight eyes and so on. And why is that?

Beautiful pictures that we find very easy to interpret, know exactly what that is. And this is a surface of who knows what. But you can immediately tell the shape and where the crevices are and so on.

So the point is that these scanning electron microscopes produce images that we find easy to interpret. So I don't know. Can you think of a reason why we don't find the transmission electron microscope images so easy to interpret, but these we can given where we are in the course?

Well, if you look at these pictures, you see that there's a variation in brightness. So that's pretty obvious. But it's a variation that's sort of special. As you approach the edges, things get brighter whereas if you're looking at the frontal surface, they're darker.

So there's a variation in brightness depending on surface orientation. Ta-da, that's what we're talking about. So the reason we find these images so appealing and easy to interpret is because they have shading.

They Have a dependence of brightness on surface orientation. And so this is obviously the head of some moth. You can see the compound eyes. You can see the schnozzle there all rolled up.

So it's so easy to see what's going on. There's an alien landing on-- and so the only thing that's a little bit odd is if you look at these shapes, so this is obviously some ovoid football-like shape, it's darkest in the middle, and it gets bright towards the edges. So that's sort of anti-Lambertian. With Lambertian, we get the brightest surface reflection where the surface is perpendicular to the incident light.

And so we expect that as you approach the occluding contour, it gets darker. So if we look at the isophotes on, say, the image of a sphere, it'll be concentric curves that sort of drop off towards the edge. And these go the other way.

And so that's kind of interesting because if we grew up in a world where most things were sort of Lambertian, we'd get very good at interpreting shape of things that have that type of surface. And yet we're able to interpret the shape of this very well. And at one point, I had a UROP take a bunch of these pictures and reverse the contrast because they should be much easier to interpret. Why didn't anyone think of that?

And well, it was not a very successful UROP assignment because with the reverse contrast, the pictures didn't look any better. In fact, they were marginally harder to interpret, and we had some explanations for that. So it shows a couple of things.

First of all, this modality of microscopic imaging is very nice from our point of view because we can understand these images. We don't need some complicated calculation. And the other one is that the human visual system apparently can do this as well, and it's not hardwired to have any particular idea about what the surface material is. It can adapt and deal-- I mean, we can see that this is not a Lambertian surface.

At the same time, we can have a pretty good idea of what the shape is. I mean, maybe not metrically accurate, but that's another story. OK, so that's what we're going to talk about next.

So let me get rid of that. How do we do this? How do we implement this?

Yes, I want to shut down. So first, a little note on how these scanning electron microscopes works. You're all probably too young to remember cathode ray tubes, but people used to create images by deflecting electron beams and having them impinge on phosphorus, which would then glow. And those devices had a source of electrons.

Basically, you're sort of boiling electrons off some surface. So that's a little heated coil. So that's the source of electrons.

And they just sort of bubble off the surface. And then you accelerate them by having some electrode, which we'll call the anode. And, well, a lot of the electrons end up there.

But other electrons are accelerated through. And then we have lenses. And these typically are magnetic.

And they can also be electrostatic lenses. And the idea is that we're going to focus this beam onto some object. So make some space for some object down here.

So with suitable cleverness and enough money, you can focus that beam down to a very small point. And then what do you do? Well, the electrons hit that object, and some of them bounce off.

So we have backscatter, backscatter the electrons. But most of them penetrate. And in the process, they lose energy and create secondary electrons. So you can sort of think of them as bumping into molecules and losing some energy and creating-- bumping electrons off ionizing things.

And some of those secondary electrons come out of the object, and they're gathered by some kind of electrode, which is also-- there are other effects. For example, there will be some fluorescence at X-ray energies. So some people measure the X-rays coming off.

And there may also be visible light and so on. But we're going to focus on the secondary electrons because that's what's used in imaging and creating those images that I showed you. So why does the secondary electron current, which we measure here, vary with the surface? Well, first of all, the way I've drawn it, it's not a whole lot of use because we just get one measurement.

So what we really want to do is scan this. So some of these electrodes-- well, typically, it's done magnetically. As you know, electron in a magnetic field will turn. And so using two orthogonal magnetic fields that you can control from the computer, you can direct that beam anywhere you want, and you can scan the object in a raster-like fashion, for example.

So this always reminds me of Mr. Farnborough, who has-- his claim to fame is that he has a dirt road named after him a few miles from where I live. He's the guy who invented television. Nobody's giving him any credit for it, and that's because he was a very clever person but not a good businessman, and he got shafted by the people who ran RCA, et cetera.

Anyway, probably never heard of him. But when he was asked by a reporter, you know, how did you come up with this idea-- I mean, to us now, it all seems pretty obvious. But at the time, what was available?

Well, there was telegraph, the beginnings of telephone, and the big question was audio is a one dimensional signal. So we understand how to measure that, how to transmit it, how to reproduce it. But this image is 2D. How do you do that?

And there were all kinds of crazy, harebrained schemes. And supposedly, his inspiration were the furrows in his father's farm because I guess he had to walk behind the oxen and hold the plow. And he decided that, well, just as I can plow this whole area by turning it into a 1D problem, I can do the same with images.

To be honest, I think that's an explanatio he came up with when a reporter asked because a lot of times, people need some way of understanding how on Earth you ever got to the solution. And then you have to dream up some reason. So anyway, that's what he did.

And that's applied here. So we scan in a raster fashion. And then what do we do?

We use the current measured here to modulate a light beam in display, either another CRT or read it into a computer and display it on an LCD. And then you can control the magnification. How do you do that?

Well, you just control the deflection. So if you deflect the beam a lot, you don't get much magnification.

And as you reduce the deflection, you get more and more magnification. And so you can get thousands, tens of thousands compared to optical microscopy, which maxes out below 1,000 pretty much despite all sorts of incredibly clever tricks. OK, why do we create shaded images?

So we have to look at how the surface interacts with the beam. So here's the beam coming in, electrons at several electron volts. And they hit this material, and they start interacting. They bump into things, and they create secondary electrons. And they lose energy as they go.

So the first few rather high energy, and then they get lower energy, and those guys bump into things. And most of those electrons just disappear in this object, but some of them that are near the surface escape and are measured by our secondary electron collector. And so basically, what we're displaying is what fraction of the incoming beam actually makes it back out again.

By the way, one of the limitations is this thing better be conductive because you're pumping all these electrons. It's not a huge current, micro amp or less. But still, this insect is in there, and you're putting a micro amp of current, and pretty soon, it'll charge up to be some significant fraction of a Coulomb, and it'll explode.

So in order to do this, you typically gold plate these objects so that they're conductive. Why gold? Well, because it doesn't outgas. This whole thing has to be done in vacuum.

OK, so that's what we have in the vertical situation. Now imagine that we have a highly inclined beam, surface element. Well, now many more of the electrons are going to escape because they're generated closer to the surface.

So that means the currently we measure will be higher when we're dealing with a surface that's inclined. And so that's it. That's the mechanism for turning surface orientation into brightness. And our job is to invert that.

So one way we can understand this is to plot the reflectance map, which we've already done for Lambertian surfaces and Hapke surfaces. Brightness versus orientation-- so in this case, coming straight down means that we get a relatively dark image because a lot of the secondary electrons are just never going to make it out of the surface. Then as we go to higher surface inclination, it gets brighter because more of the secondary electrons escape.

And in typical arrangements, the thing is axially symmetric. So whether these electrons come out in this direction or that direction, if they get out of the surface, they're going to be counted. And so that means that this is going to be symmetric.

And so if we plot the isophotes, we're going to get something like that. Put numbers on it. Where it gets bigger, the further out we get, the more inclined the surfaces.

And yes, you can modify this. You can remove some of the collectors or wire this collector to a different sensor and so on and make this asymmetrical. But the standard way of using it is just to measure the total secondary electron current. And that gives us that type of reflectance map.

OK, so now if I measure the brightness at a particular point in the image, I get the slope of the surface. So I can directly translate that. If it's 0.7, then the slope is whatever that distance from the origin is.

But I don't get the gradient. Right, what I'd like to know is what's the surface orientation. And as usual, we got two unknowns, p and q. We only have one constraint. We measure the brightness.

And so yes, we've reduced the number of possibilities, but we still have a one dimensional, infinite set of possibilities. So the slope typically is thought of as a scalar whereas this is a vector. So it's a little bit like speed versus velocity.

I mean, not everyone agrees on those definitions. But to me, speed is a scalar quantity going 50 miles an hour whereas velocity is a directed vector. So OK.

So what to do? Well, it's just another example of this shape from shading problem. We're going to have some distribution of some pattern of brightness.

And our job is to estimate what the surface shape is, and we have a reflectance map to help us with that because it allows us to look up for any particular brightness we measure what the slope is, unfortunately not the gradient. If it was the gradient, then we would be in better shape. And we're going to do this for various special cases and then for the general case, where we can use any reflectance map we like. And that's important because we don't want this to only work for say, Lambertian, surfaces or the lunar surface or whatever. So OK.

So when you get a chance, look up scanning electron microscope images and just Marvel at the wonderful world that exists there and how we're able to understand it so well. And, you know, puzzles like why is it that if you reverse the contrast, it's not easier to interpret the image, which is what you'd expect. You'd expect that a surface that's brightest where it's facing you would be easier to understand that than one that's darkest.

OK, so in preparation for this, I want to talk a little bit about how to go from a needle diagram to shape. Now where are we going with this is we're going directly to a method that takes us from a single image to a shape. But meantime, we have some tools, and I want to exploit those tools. So for example, what's a needle diagram?

Well, it's just surface orientation at every pixel. And what are the needles? Well, you can think of each facet of the surface as having a normal that's sticking out. And you're looking down at that normal. That's the needle.

And the view of it, you may remember from the very first class, if you're looking straight down perpendicular to the surface, then that needle is just a point. And if it's not, then it'll be pointing in a direction. that tells you what the surface gradient direction is. And the length of it will tell you how steep the surface is.

So where do we get these? Well, we're not going to get them from there. So there's more work to be done. But for example, we get this from photometric stereo.

So in photometric stereo, we're not computing z as a function of x and y. What we're getting instead is p and q as a function of x and y, of course remembering that p is dzdx and q is dzdy. Well, it's our estimate based on our image brightness measurements.

OK, so that seems actually like a very simple problem. Why? Well, in the discrete case, we have a number of unknowns equal to the number of pixels. We're trying to recover z at every pixel, let's say. So there are millions of unknowns.

But at every pixel, we also have two pieces of information-- p and q. So if we have a million pixel image, we have 2 million pieces of information. So it's overdetermined. We actually have more information than we need.

And as we saw, that's always handy because that means that we would be able to reduce noise and get a better result than if it wasn't overdetermined. So in fact, we have twice as many constraints as there are known. OK, so what can we do?

Well, one thing we can do is suppose we start here and just integrate out p, which is dz dx along this axis. And what we'll get is the change in height. So z of x is z of, let's say, 0. Suppose we start at the origin plus from 0 to x.

Right, because dz dx is the derivative. You integrate the derivative here. You know all that.

So OK, so I can get the height anywhere along the x-axis. And similarly, I can get the height anywhere along the y-axis by integrating the other thing I know. So then from there, I can combine them, and I can go partway up on the y-axis and across on the x-axis. So I can get partway on the x-- so I can fill in the whole area in various ways like this.

And actually, I could take a curve. So let's take that curve. And so along that curve, what we're looking at is a combination. Right, so what is this? Well, that's just the dot product of the gradient pq dot dx dy.

So that just tells me the change in height. So this is actually delta z, If you like. That's a change in height if I take a small step in the x-direction and then a small step in the y-direction.

Pretty obvious. And so I can do that for all these points. I just pick contours and integrate up.

Well, that's OK, but what if somebody else comes along and says, oh, I don't like the contour you chose? I'm going to go this way. And what you'd hope, of course, is they get the same answer. But there's no guarantee of that because these p and q's are determined experimentally.

They're subject to measurement noise. So they're not perfect. They're not actually the derivatives of z with respect to x and the derivative of z with respect to y.

They're our estimates which include some noise. So OK, well, what we'd hope is that they come out the same. And what that means is that actually, what we're hoping is that if you go around the loop, if we go this way, da da da, up there, and then we come back, what should this equal?

STUDENT: Zero.

PROFESSOR: So that should be zero. Yes, thank you. And every jogger knows that that's not true.

If you jog in a loop, you seem to be going uphill all the way. OK, I guess we don't have many joggers here. So-- oh, it's like-- well, OK.

So that should be true. So there's no guarantee that when we measure p and q experimentally, we estimate them, that this is going to be true. So what do we do with that?

Well, one thing we can think about is turning this into some sort of condition on p and q. That is, p and q have to satisfy some constraint for this to be true. And this has to be true for any loop, not just that one.

And I can decompose a large loop like that into small loops. So suppose it's true for that loop. Well, then I can put another loop next to it and another loop.

And now rather than go around these four loops, I can eliminate the inner parts, and they cancel each other. See, if I am going from here to there and then I'm going from there to here, I'm back at the same place. So this is a crude way of proving that if it's true for a small loop, for any small loop, I can decompose any large loop into lots of small loops.

And then it's going to be true for the large loop as well. So OK, so let's see what-- so let's have a small group of size delta x in this direction and delta y in that direction. And let's suppose that this is the point xy.

And let's see how the height changes as we go around this loop. Start down here, say. Well, then we need to know what the slope is in the x-direction.

So that's p. And let's take the slope at the center of this stretch, which is going to be p of x and y minus delta y over 2. And that's the slope, and we're going delta x with that slope.

So we make this loop small enough so that we can use this linear approximation, that the slope is pretty much constant over that stretch. Then we go up. OK, so that's going to be-- we need the slope in the y-direction, which is q. And we'll take the slope and estimate it based on the center of this line.

So that's at x plus delta x over 2 and delta y. Sorry. And and then we go across the top, minus p xy plus delta y over 2. Oh, this is times delta y.

And then we go down. So this is minus q. And that should be 0. We should be back at ground zero.

So what we get here is if we do the Taylor series expansion of that, we get-- and down here, we get p of x comma y plus higher order terms. And then we expand this.

We get q of x comma y plus delta x over 2qx-- let's see-- plus. OK, and of course, these terms cancel.

So we end up with py delta x delta y minus qx delta x delta y is 0. Or in other words, these two are the same and cancel out the the sides of this little box. And so we get pi is qx.

So if this is true of very small areas, then that condition has to be true. And that makes sense because p was supposed to be our measurement of dz dx and q was supposed to be our measurement of dz dy. And so that's z sub x y, and that's z sub y x.

Now it's slightly confusing because that's obviously true of the original surface as long as it's smooth enough that the next partial derivatives don't depend on the order. OK, so if p and q really were the derivatives of the surface z of xy, then this would be true. But we're measuring them from experimental data. So this typically won't be true, and if we take this integral, we will get different results depending on different directions.

And so this is like overdetermined linear equations. There is no solution. All we can do is find some least squares approximation. So same here-- there is no surface z of x,y that will give us exactly the p and q that we measured from photometric stereo or some other visual method.

OK, now actually, we can get there a more elegant way, which is handy because we'll use this again later. So like many important theorems, this one has many names. I guess it's attributed to Gauss, and it's a special case of more general theorems used in fluid dynamics. But what it does is it relates contour integral to an area integral where-- let's draw some sort of shape.

So l is this contour and D is that area. So why do we care? Well, in machine vision, often we have computations that take us over all pixels. And there are a lot of pixels.

Any time you can reduce that computation to a computation about the boundary of a region alone, you've got a huge win. You're down from millions of calculations to thousands of calculations. So there are several places in machine vision where we use a trick like this to turn an integral over an area into an integral over just a curve with a huge benefit.

Of course, it's a very specialized thing. There are only that many things that have the special property, namely things that can be written in this form. But you know, suppose, for example, you want to compute the area of a blob.

Well, you can count the pixels. You can just go row by row by row, see, OK, this pixel's in the blob, that one's not, and so on. Or you can just trace the outline. Turns out you can compute the area just by tracing the outline.

In fact, there used to be a wonderful instrument which is fiendishly difficult to understand. But it allowed people to compute integrals in an analog fashion. You should look it up. It's an instrument that often was made out of precious metals because it was only for special people like surveyors.

And you would basically hold one part of it and trace the outline and compute the area. Once you complete the loop, you've computed the area. You can read it off the instrument.

And that's an example where we're turning an area integral, the integral of 1 over that area, into a contour integral. As you go around the edge, there's a little wheel that slips on the paper in just the right way and so on. So this isn't just something useful in digital domain. It's been used in the analog world as well.

So area is an example of something where this is a useful formula because we can compute area by doing something on the boundary instead of doing something in the interior. But it turns out that there are lots of other things we want to do. For example, in vision, often we want to know where something is.

And in the simple case, it's a blob, and where is it? Well, the centroid is a very good way of talking about the position of a blob in an image. And so how do you compute the centroid?

Well, again, you can go pixel by pixel and cumulate x times something, throw it in accumulator, and when you're done, you divide, and you get the centroid. Or you can go around the boundary, which is much faster. And continuing along that line of reasoning, there are things called moments of which the centroid computation is the first and the area computation is the zeroth. But generally, you can compute moments by just going around the boundary.

And moments are useful in describing shapes. So so far, we talked about area and position, centroid. But you might also want to be able to say something like, oh, it's elongated. And so these moments are useful for that, and they can all be computed in this clever way.

So let's apply Green's theorem. There's a 3D version of that, turning 3D integral, volume integrals, into surface area integrals. But since our images are 2D, we typically don't need that.

OK, so we apply this to our problem. So we're trying to match something in this formula with what we were doing, and I guess we had pdx plus qdy up here. So let's try that.

Let's try using Green's theorem. So m is q dq dx minus l is p. And we're saying that that is zero, right? We're going around the loop.

OK, and this has to be true for all, not just particular loop, but any loop-- small loop, large loop, whatever. And that can only be true if the qdx equals dpdy because suppose that quantity, that interbrand, was non-zero at some point. All I need to do is construct the loop that includes that point, and I'd have a contradiction.

So that again confirms the conclusion we arrived at painfully here by making a small square and then using Taylor series. This is a slightly more elegant way of getting to the same thing. OK, so what to do?

p and q do not satisfy that constraint. So we could try to introduce that as a constraint. You could say, OK, find the z of x and y that has these p's and q's as derivatives approximately and also satisfy that constraint.

Well, we haven't talked about how to solve constraint optimization problems. That's a little harder than the least squares we've done so far. So let's try a different tack.

So let's basically do just brute force least squares. So we try and look for a surface z such that that is the smallest possible. Right, so ideally, if there was no measurement error and we had the correct surface, then there was a z such that its x derivative matched the p we computed from the image and its y derivative matched the-- sorry, the p that we got from the image and this one matched the q that we got from the image.

But because of measurement noise, we expect that will not be possible. So let's at least try and make it as small as possible. OK, so that's the least squares problem in a nutshell.

And well, we solve it as usual. We just differentiate with respect to z. Yes, can we do that?

What is z? Is z a parameter? Right, so now unfortunately, we can't do that.

If we had a finite number of knobs, we could turn-- parameters, variables. Then we could just differentiate with respect to each of them and set the result equal to 0 and we're done. That's what we've been doing.

Unfortunately, z here is a function. And so there's no sort of sense of a way of talking about the derivative with respect to a function. So it has not a finite number of degrees of freedom. It has an infinite number of degrees of freedom.

And there's a subject in mathematics that deals with that-- calculus of variations. And it's called that because basically, you say something like suppose that z is the solution. Then if I make any small variation on z, that integral should go up. And based on that very sensible idea, you can come up with some equations for solving that problem.

But we're going to remain in the trenches and work with discrete version for a moment because-- and this particular problem is pretty simple. The reason to work through it in detail is because it hides a lot of the pitfalls that you might run into when you solve a more complicated problem. OK, in a discrete case, we don't have a function of x and y.

What we have is a bunch of discrete values on a grid, so a finite number of unknowns-- maybe huge, maybe a million. But we can use calculus, ordinary calculus. Just differentiate with respect to each of them. Set the result equal to 0.

So true, we're going to get a lot of equations, but our methods apply. We don't need to invent something new. So this is what we're looking for, and what we're given is again a set of gradients on a grid with measurement noise in them.

And we're trying to find a value of z at every grid point that will make the error as small as possible where the error is the discrete equivalent of this thing. So let's look at that. So i and j are row and column numbers.

So slight annoyance here for computer scientists and mathematicians, which is that you would like the first discrete index to correspond to x and the second discrete index to correspond to y. And as a student, I had a real hard time with that. But of course, in mathematics, we count rows down and columns across, and we write them ij in the other order and also with I reverse.

OK, so this is-- what is this? We're trying to estimate the derivative in the x direction. And that's why we were varying y rather than i, right?

And so this is our estimate of the derivative in the x direction. And that should be small. It should match what we observed from the image.

And we also need to add in the other term. We also want to match the derivative in the y direction. And now typically, there aren't the set of values of z that will make both of those terms zero.

So we kind of compromise. We just want to make them as small as possible. So we're going to minimize this.

And what can we minimize? Well we have the set of zij's. OK, so we then differentiate and set the result equal to 0 for all possible values of ij.

So if the image has a million pixels, there be a million equations coming from that. Fortunately, because we picked least squares, the equations are all going to be linear. So as much as the disadvantages to using least squares, such as not being robust against outliers, we can solve these equations because they're just going to be linear equations. OK, so here, pay attention.

Big problem-- if you differentiate with respect to zij, you get the wrong answer. Why is that? Well, ij up here, these are dummy variables.

If I replace ij with alpha and beta, it's the same sum. But then if you differentiate with respect to zij, so another way of thinking about it in programming language terms-- you know, you've got an identifier collision. You're using i and j to mean one thing in here, and now you're trying to make it mean something else.

So here we have a sum of all possible ij's, and now you're going to differentiate. So long story short, pick some other identifier names. And you might think, OK, that's kind of obvious.

Yes, but there have been papers published that got this wrong. So it's possible. It can happen.

OK, so, well, this sounds kind of messy because we have a sum over a million terms, and now we're going to differentiate. But the good thing is that, yes, we have a lot of equations, but they are all very sparse because if you think about, OK anything here having to do with row 1,000, when you differentiate with respect to z in row 990, it's not going to have any effect.

So all of those derivatives are 0 except a small number, the small number where kl matches up with an index in here, ij, or kl matches ij plus 1 or kl matches plus 11j. That's it, three terms. So we're going to have a lot of equations each of which has three terms in this case.

OK, so let's try that. So let's first try where we match kl is i comma j. So there's a square. So differentiate that, we get 2 times that times the derivative of the thing inside with respect to zkl, which is going to be minus 1 over epsilon.

Right, so it's a square. We get twice that term. And then we have to differentiate what's inside with respect to zkl. And we said that it matches this term.

So it's going to be minus 1 over epsilon. And sorry if I'm going over this in a very pedestrian way, but you'll need to do this in a more complicated example, and it's easy to not do it right. So OK, so that's that one.

But there's also this match. So we have to add to this 2 times this term. So we get twice this term and then differentiate this with respect to zkl, which is matching that.

So that gives me minus 1 over epsilon again. OK, so that's-- there are two terms. Then we also have to consider where kl is ij plus 1.

In other words, k is i. l is j plus 1, or conversely, j is l minus 1. Right, so we differentiated with respect to that.

We get 2 times this OK, OK. So we get twice that term, and we have to differentiate what's inside with respect to zkl, which matches that one. So it's going to be plus 1 over epsilon.

And we're almost done. Just one more term. This one here, now we have k comma l is i plus 1j. Or in other words, k matches i plus 1, meaning i is k minus 1.

And l is just j. So we get that one. That's going to be zkl minus z.

OK, i is k minus 1, l over epsilon minus-- and plus 1 over epsilon. And all of that's supposed to be 0, right? That's the least squares method.

And this is going to be true for all points in the image. So fortunately, this simplifies. So first of all, of course, I can ignore the two because if twice something is 0, then something is 0.

So forget the 2. I could also cancel out one of the epsilon, but I'll keep them just for reasons that will become apparent. OK, so now I'm going to gather up the terms.

Let me first gather up all the terms in p. Well, there are only two of them. So that's pretty straightforward.

So I get pkl minus pkl minus 1 divided by epsilon-- so those two terms. And let me gather up all the terms in q.

So I get qkl minus qk minus 1l over epsilon. And then let me gather up all the terms in z. Well, there are a bunch more of those.

And they are 1 over epsilon squared. So let me write that out front. So we're going to have minus zkl plus 1 minus zk plus 1l minus zkl minus 1 minus zk minus 1l.

And then zkl occurs four times. Well, let me just keep the epsilon squared. And all of that's supposed to be 0.

So what are those terms? Well, this looks awfully like the x derivative of p, right? Because we take the value of p at positions, separate it in x, and we divide by epsilon.

So we can think of this as p sub x. And this looks awfully like y derivative of q. And then I claim that this thing here corresponds to the Laplacian of z.

Let me see why. OK, so I can draw a little diagram here and gather up all these terms. So I get 1 over epsilon squared, and I have four for kl and then minus 1 for all of the neighbors.

And now z sub xx OK, so those things I'm drawing down there are variously called computational molecules or stencils or something like that, and they're a graphic way of showing how to compute an estimated derivative. Now I didn't bother for these, but I could. So I could, for example-- that's the computational molecule for the x derivative.

And then here's the computational molecule for the y derivative. It's so obvious, we don't even bother with that. But when we get to higher derivatives, it's handy to draw these diagrams. So if we could convolve this with itself-- remember 603-- then you get that.

And if you convolve this with itself, you get this second derivative. So just how do you convolve? You flip and shift.

Anyway, you could also get this from Taylor series or any number of other methods for estimating derivatives. OK, so then if we combine these two, if we add these two, we get that diagram-- well, except for the sign.

So this is actually minus the Laplacian but it occurs on the same side of the equal sign. So when we move it to the other side, it becomes plus the Laplacian. So the Laplacian is actually heavily used in machine vision, particularly pre-processing. So we should become familiar with it.

By the way, another way of writing it is that. And engineers have a different point of view on that as mathematicians. I gather mathematicians prefer the first way of writing the Laplacian, and fluid dynamics people like the other one. But whatever. It's a Laplacian.

So the Laplacian is an interesting operator from many points of view. It's a derivative operator. We've already seen the brightness gradient plays an important role.

First derivatives-- for some operations on images, we would like things to be rotationally invariant. In other words, if you are performing some image operation enhancing edge detection, you don't want to depend on your choice of xy-coordinate system so that when you turn the coordinates, you would expect things to work out pretty much the same.

Of course, they'll have different coordinates, but they'll be in the corresponding place. So one of the big questions is, are there derivative operators, which are very useful in edge detection, that are rotationally symmetric so that when you rotate the coordinate system, they do the same thing? Well, the Laplacian is the lowest order linear derivative combination that does that.

It doesn't look like it, does it, because we're adding second partial derivative with respect to x and the second partial derivative with respect to y. But if you go through the pain and agony of changing coordinate systems and computing the first derivative and then computing the second derivatives and you add them up, and magically, in the new coordinate system, you get exactly the same. So suppose we have a coordinate system like this, and then we go to a rotated coordinate system.

It turns out that zx prime-- that in that rotated coordinate system, the Laplacian is the same. I'll spare you the details. It's pretty boring.

But this makes the Laplacian a lot of interest in all sorts of image processing operations because it is rotationally symmetric. Of course, on a discreet grid, we are going to be somewhat biased anyway. Like, does this look rotationally asymmetric? Well, not terribly.

So there are actually better approximations if you're on a square grid, and we'll talk about them. And if you're not on a square grid, you can do even better. For example, if you're on a hexagonal grid, you can get a better approximation. Anyway, back to this-- so what we've done over here is we've bypassed the calculus of variation for the moment, and we somewhat painfully went into the discrete world of discrete pixels. And we found an equation that has to be true at every pixel in order for this to be a least squares solution.

That is, this is a solution that minimizes that error. It's the best fit. It's as good as we can get. And then sort of looking at this formula here, we decided oh, maybe that's the discrete version of that continuous equation.

And that's exactly right because if we use the calculus of variation, we go directly to that in one step. So why didn't we do that? Well, because there's this huge learning curve. So it's one step after you do a lot of stuff, and I didn't want to do that right now.

So anyway, what do we do with these equations? How do we solve them? Of course, we can-- they're linear equations. So we can use Gaussian elimination or some perhaps slightly more sophisticated algorithm, all of which take a long time for lots of equations.

Right, typically, they go something like n cubed or, if you're a complexity theory person, n to the 2.76 or whatever with a very large multiplier in front. But whatever. Once n is a million or 10 million, that's a large number, and it's not likely you're actually going to be doing that.

So what to do instead-- well, the great thing is that we have a sparse set of equations. So we got a million equations, but each of them only has a handful of terms in it. So they only involve-- each equation only involves five values of z.

So we got-- you think of writing them out. There's a million rows of these, but each row only has five non-zero elements. Well, it turns out we can then solve this iteratively pretty simply.

And of course, it's easy to propose an iterative solution. It's hard to show that it converges. So we leave out that part, and we'll just appeal to the textbooks that say that it does converge. So what can we do here?

Well, we can pull out one of these terms and pretend that we know-- we have an initial guess. So we know what these values are at the moment.

And we're going to compute a new value for this one. So let's see if I got it written down or not. So I'm going to use subscript in parenthesis as a way of indicating the iteration step.

So this is step n plus 1. And that's step n.

OK, I almost wrote superscript parenthesis n here for the n step. But no, these are from the image. They're fixed.

Once we've processed the images, that's given. The only thing that might change is this lot. And what is that? Well, that's local average. And in our case, with that computational molecule, that, looks like that.

So the iterative step is very simple. We go to a particular pixel, and we get a new value by taking the average of the neighbors from the previous step. And then we add in this correction here, which is based on the image information.

And what does this do? Well, it brings us closer to satisfying the Laplacian condition because this is the estimate of p sub x and this is the estimate of q sub y. and the difference between this and that is our estimate of a Laplacian. So as we go along, we adjust that pixel value using that very simple equation.

So here's where the sparseness comes in. This wouldn't be very useful if we had a million terms in this. But we've only got four neighbors to consider.

And you can show that this converges. We won't do that. And it's going to be very much faster than trying to solve it with Gaussian elimination, particularly since you don't need infinite precision. You can stop after a certain point and relax.

So a couple of things to point out-- first of all, this is closely related to solving the heat equation, this type of iteration, because the heat equation is the second order PDE just like the equation we have. It's also the diffusion equation. And so there are loads of techniques available from other fields that tell us how to do this efficiently, how to do it in parallel, and whatever.

So again, the idea is you step through the pixels. And at each pixel, you compute the average of the neighbors and then add in a correction to them. And that's the new value. And you just keep on doing that.

Now for a start, you don't have to do this sequentially as Farnborough did the rows of his field. You can do this in any order you want. And in fact, there are advantages in terms of convergence if you do that.

Also, you can parallelize this. You can do as many of these as you like in parallel as long as they don't touch each other, right? So if you are making a change to a pixel based on some neighbor, you do not want the computation that changes the neighbor at the same timestep.

But that's huge because we can divide the image into, I don't know, nine sub-images, little 3 by 3 blocks. And while we're operating on this one. We're not allowed to touch these others. But that's fine.

That means we can have huge parallelism, a million pixels divided by 9. Well, we probably don't have 110,000 processes. So we can do something slightly more intelligent. Anyway, methods for numerically solving these equations are well--known, and it's a very stable problem in the sense that if there's noise in your measurements, it'll still converge. It'll still work, and everything will be cool.

OK, so-- OK, we have a bit of time. So what have we done? Well, what we've done right now is learned some techniques because in a way, this isn't directly useful except for the case of photometric stereo. And we now want to move on from photometric stereo.

But the kind of tools like this discretization and avoiding the conflict of identifiers and then looking at the terms and seeing, oh, this is an approximation of the derivative-- those will reoccur. So it was good that we did this. But beyond that, this shows you if you have photometric stereo data, you can reconstruct the surface in a least squares way and get a reasonable solution that matches the experimental data.

OK, now we're going to go back to the somewhat more challenging problem of single image reconstructions. And we solved one case already, which is if the surface had these very special properties. Right, so let's just go over that a little bit.

So this was where we had-- brightness was a function of that, which was incident. And in this case, the reflectance map was very simple. We had these parallel lines. They won't be equally spaced because of the square root, but they're parallel lines.

And each of them has the 1 plus psp plus qsq equals a constant. So the isophotes in the reflectance map are these parallel straight lines. And that's actually what allowed us to solve the problem because then we said, well, suppose I rotate to a somewhat more useful coordinate system.

In that coordinate system, I can just determine the slope in that special direction. So if I read off brightness value like this, in the original coordinate system, it's a linear relationship between p and q. But in the new coordinate system, it's conveniently giving me an exact value of p-prime. And it tells me nothing about the q prime.

So I've sorted of maximized the information in one direction and made it worse in another direction. And I'm just going to write this down. I'm not going to do this carefully, but it turns out that we will need to deal with these coordinated transformations at some point. So-- oops.

So-- oh, and what is this angle? Let's call it theta s. Since that ray, that access p prime, is perpendicular to all of these lines-- right, I'm going to have that p comma q dot ps qs. It's going to allow me to determine that angle is given by this triangle.

qs-- so I can-- well, might as well write it down. So whatever the light source position is, I can figure out what that angle is. So actually, the light source is somewhere along this line.

Let's say it's up here. OK, now what I want to do is rotate the image ahead of time so to make this unnecessary later. So what's the relationship?

Well, I'm sure you know this. And we can call it theta s or just theta for simplicity. OK, but what we need is a relationship between p and q and p prime and q prime

So we need dz dx prime is dz dx dx dx prime plus dz dy dy dx prime. And of course, this is p prime that I want to go for. This is p. This is q.

And so I'm left with that. Well, annoyingly, that's going the wrong way. So I actually need the inverse transform of this. So what's the inverse of rotating through theta? Rotating through minus theta, right?

So x is x prime cos theta minus y prime sine theta, and y is x prime sine theta plus. OK, so then this is dx. dx prime is cosine theta. And this is dy dx prime. So that's sine theta.

So actually, I got p prime is p cosine theta plus q sine theta. And similarly, I'm going to get q prime is minus p sine. So this is nice.

And you might say, well, I could have guessed that, right, because it has the same form. Well, think again because when you get to the second derivative, this doesn't work. You can't just guess the answer.

And that may be relevant because we will be talking about-- we're talking about the Laplacian. So right there, if you were trying to prove that theorem I said about rotational symmetry, you'd need to get second derivatives, and then this wouldn't work. Anyway, so I'm going to do that.

I'm going to rotate these coordinate systems, and then I'm going to end up with something like-- which relates the image measurements to the slope. And the details of the equation isn't so important. What's important is that I can be at a particular point in the image.

I read off the brightness, and I compute this quantity. The other things are all known. I know where the light source is. And I've got the slope, which is amazing.

I have the slope of the surface in a particular direction, and that's enough. We can now reconstruct the surface. But this did require that we rotate the coordinate system, that we have a particular direction. And that's going to be true in the general case.

So this is only for Hapke, where we have nice, straight line isophotes in the reflections map. But we'll generalize that. And OK, so what do we do with that? Well, we went over this a little bit last time because we can just integrate out z of x is-- right, so we have the slope.

We just integrate it out. In terms of discrete implementation, we're here. We go a small distance delta x-- like, I don't know, a pixel. We know the slope.

So we know how much z is changing. Right, so we have a new value of z And then we look at the image and say, what's the brightness here? And using that formula, we find a new value for p prime, and we continue.

We take another little step delta x, and we get there and so on. And so we can build up a shape in the discrete world. And we can go the other direction as well. We can go minus delta x and build up a profile this way.

So that's kind of amazing that you can do that. Well, that's a single profile. But now in the image, we have the y direction to deal with as well. But we can do this for any y, right?

But in each case, we're confined to running along a row. There's no interaction between the rows. We just treat it as a 1D problem along each row. We read off the brightness at each pixel. We convert it to the slope using that formula and then use that slope to see where we go.

So there are couple of things about this that are interesting. So one is that if I add a constant to z, does anything change? So in fact, let's go back to this one up here, this overdetermined problem which has now sort of disappeared. If you add a constant to z, does the Laplacian of z change? No, right?

So that's interesting because that means that, OK, you can recover z. But actually, the absolute height of z you don't know. And that's reasonable because if brightness depends on surface orientation, then moving the surface up and down shouldn't have any effect. And in particular, in the case of orthographic projection, it doesn't change in any way. It's not even magnifying or minifying or something.

And so similarly here, the shading information tells us nothing directly about depth, only about relative depth. You know, what's the slope of the surface? And so here, to actually get a reconstruction, I need an initial value for z.

And, well, that's a problem. Now if it was just-- like in the case of the Laplacian solution, if it's just an overall change in height, that would be easy to deal with. That's just one unknown.

But here, we potentially have a different initial value for every row that we're integrating along. So you can imagine that we're computing these solutions, and we have an initial value, say, here. And we integrate it out.

But we can pick those independently, or somebody could go there and measure it. But that defeats the purpose. We're trying to find the surface shape of the moon before anyone goes there. And so having someone go there and survey it kind of defeats the purpose, although it does mean that you only have to survey one curve, and then everything else sprouts off from that.

And we already talked about various heuristics to try and build something even though we don't know that. So we don't need to have these initial points along the y-axis, though. We could have initial points along any curve. So that means what we actually need is an initial curve. Doesn't really matter what the curve is as long as it's not parallel to the x-prime . Axis

OK, and then now, we can map this back into our original unrotated world. And these profiles now will run at an angle. And we have some sort of curve here.

Now let me introduce coordinates. Let's call the initial curve is some function of eta. So initial curve is x of eta, y of eta, z of eta. Right, so we'll assume that's given.

And then we go away from that using some other parameter, [INAUDIBLE]. And that's our x prime in this case, but we want to generalize that. So this is going to be very similar to the general method.

And so we take a step delta x along this curve, which would be a step in x prime. And because of the rotation-- so this is the scheme we're following. So we're we exploring the surface by moving along these curves, and from step to step, this is what we do.

We have delta x and delta y change in a way that's at an angle, theta s, in this original xy coordinate system. I've gone back to that system because in general, we won't be able to use this trick. It's only for Hapke surfaces that it works.

And so you can see the numerical calculation is very straightforward. We pick up the image brightness. We plug it into this formula, and we get a step in the z direction.

And so we just continue doing that and explore that curve. And then we do the same for the next curve, and they can be computed independently. So you can do them in parallel if you want. Just one little note-- you can change the speed that you explore the solution in because if this tells me to take a step delta x, delta y, delta z, maybe I'll take 2 delta x, 2 delta y, 2 delta z or a half.

And how would I decide? Well, suppose that you only move 1/100 of a pixel. That's probably a bad choice for increment.

Or suppose you move 10 pixels. That's a bad choice for increment. You want to reduce it.

And the convenient thing is we can easily change the speed. We just multiply all three of them by the same constant. And so if you want simple equations, we can just multiply by that constant, and you get a somewhat simpler looking expression.

Right, so we're moving in the x direction proportional to qs. We're moving in the y direction proportional to ps. And those are fixed. We're just driving along. And then in the z direction we have this simple formula.

And so what we'll do next time is generalize this to arbitrary reflectance map, not just Hapke-type surfaces. And please do have a look online for pictures of images taken by scanning electron microscope. They're really neat, and they provide nice examples of shading and our ability to interpret shading. So--