[SQUEAKING]

[RUSTLING]

[CLICKING]

BERTHOLD HORN: We'll start off today by talking a little bit about noise gain. In other words, the relationship between errors in measurement and errors in estimation of quantities you're interested in about the environment. And I just happen to have this example on my computer. It's not vision-related but illustrates some of the points.

So what is this? This is a indoor equivalent of GPS where instead of using the timing of signals from satellites, we use the timing of signals from Wi-Fi access points. And it's been in the works for several years, but it doesn't really exist yet.

For example, at this point, you need Android 9. And the only phone that runs Android 9 is Pixel. And most of the Wi-Fi access points don't-- well, all of the access points don't support it. I've been driving around looking for them. I've registered thousands, and I found one.

Anyway, the idea is that in the future, we will be able to measure distances to access points. And of course, you can imagine that the accuracy is somewhat limited, because electromagnetic radiation travels at a nanosecond per foot. So you have to really measure these round-trip times from phone to access point and back with very high precision.

But suppose you do. So you have a bunch of access points. I mean, we've got four right here in this room. And then, your job is to determine the accuracy of your location.

And just as in GPS, you may be able to measure the distance to the satellite to, let's say, 10 meters. That doesn't mean that the accuracy you can expect of your x, y, z coordinates-- latitude, longitude, and altitude-- is 10 meters. In fact, it's typically a lot worse. And in GPS terminology, that's called dilution of precision.

And in GPS, it's different for horizontal than for vertical. That's another interesting point, that you can determine your horizontal position with higher accuracy than your vertical position. So like this morning, my phone's GPS said I was at minus 36 meters. And I know Boston is at sea level and the water level is rising, but that's probably not accurate.

So point of that is that when we do this noise gain analysis of some machine vision process, it's probably going to be different in different directions. So it's not just a single number that says your accuracy is 1 meter. So back to this one here. So the green points are for Wi-Fi access points-- nicely, symmetrically placed. The red points are possible places where your cell phone can be.

And the circles, or ellipses, are areas of-- or contours of constant error. So in other words, if you move out to that curve, you will be in error with respect to the distances you can measure. And so turning that around, it tells you the accuracy you can expect.

And it looks very nice in the middle there. They're small circles, meaning, A, that you will be able to determine your location pretty accurately and B, that there isn't a difference in different directions. It's not like you can determine x better than y.

But then, when you go outside the convex hole of the responders, you'll see that things become elongated. We have these ellipses, and they get larger down here. What does that mean? Well, it's that you won't be able to measure your accuracy very well in this direction. But you can still determine it pretty well in that direction.

And why is that? Well, if you think about the distances to the responders and how it changes as you move around, if you move in this direction, you're moving at right angles, pretty much, to those vectors. And so you're not changing the length of those vectors by much. And so correspondingly, you won't see a big change in the signal. And so correspondingly, you won't be able to accurately determine the position.

Whereas if you move radially, if I move in this direction, I directly have an impact on the distance to that one and the one up there. And with this one, I'm at an angle. But it's like 45 degrees, so that's square root of 2-- 1 over square root of 2, so that's 70%. Yeah, right. It's the latter.

So the idea is that the red is a possible position for my sensing equipment and the circle around the red is how far I have to go before the error is a certain threshold value. So in other words, if I'm here and I measure these distances and there is no measurement error, then the sum of squares of errors will be 0.

If I move a little bit away, my distance from the responders is wrong. And I can-- in this case, I have four of those numbers. And I look at how big they are, and I add up the sum of squares as a overall measure of error.

So when I'm drawing the circle, that's the locus of all the points that have the same size error. And out here, you can see that I have to go further away before I see that error. And conversely, I'm less sensitive to the error and I will be able to not determine the position as accurately. So that one's kind of pretty clean.

But let's suppose that we have three transponders. Now, you can see it gets a bit messier and slightly surprising results. One is that out here, these things that look like ellipses become non-elliptical, because they are ellipses in the limit of very small displacements. But for larger displacements, the curve can have any shape. But in most cases, we'll be focusing on the infinitesimal case-- make life simple.

So what's interesting about this? Well, one of the things is that it's pretty accurate down here, which is away from the responders and it's away from the centroid. You'd think, well, maybe it should be pretty good at the centroid, or at the intersection of the bisectors of the angle, or the intersection of the perpendiculars drop on the other side of the triangle-- something like that. But that's not the case.

And so this is interesting, because it means that you may be able to do things away from where the responders are. So the responders could be like down a hallway, and it'll still give you good accuracy away from that. At the same time, we can show that if you put them in a line, that's a really bad idea. And of course, that's what they are here-- everywhere. They're just-- anyway.

And again, the idea here is that if I'm over here and I move a little bit to the side, I'm not changing the distance to those three responders much, because I'm moving pretty much at right angles to those three vectors. And therefore, I won't really notice that I've moved. And therefore, conversely I can't determine my accuracy in that direction, as well.

So that's with three. And let's go one more. So that's with two responders. And you can see that there are some areas where it's working just fine over here, but in between them, not so good. Why?

Well, because if I move horizontally, I'm not really changing the distance very much. It would be second-- it would be quadratic. It will be moving, changing as x squared. And so for small x, it's a very small change. And even worse out here.

And then, you start to see something. Maybe you can see a locus of places where it's working pretty well. And their property is that from each of those positions where it's working well, the two responders seem to be at right angles to each other.

And that makes sense, because basically you're saying, OK, I have a constraint in a direction and a direction at right angles. If I rotate the coordinate system, I can make that x and y. I have a constraint in x, and I have a constraint in y. That's like the ideal condition.

If the two directions are very similar, then I can move at right angles to the average and not change things very much. So it turns out that points where the responders appear 90 degrees apart are particularly good. And what is the locus of all such points?

Geometry, theorems about circles-- so the points on the periphery of a circle have the diameter, the endpoints of the diameter, at a right angle apart. That's not the way to say the theorem, but I think what I mean.

So if I draw a circle with the line connecting the two responders as diameter, points on that circle will have that property. And so not surprisingly, those are the points where I'm getting good accuracy, even in this simple case. OK, enough of that. Let's go back to--

So the idea really is very simple. It's that we have some sort of forward transformation that takes us from a quantity and environment we want to measure-- I don't know, distance, velocity, whatever-- to something that we can observe in our instrument, the camera, say. And so the forward problem is that we-- let's just take the simple case of a scalar.

There's an x. And we don't observe x directly. We can measure f of x. And of course, in vision, our problem is to go the other way. If I measure f of x, what is x? And if f is inevitable-- nice, smooth, monotonic function, we can do that.

But then the second question is, OK, my measurements aren't perfect, so I don't really know f of x perfectly well. What does that mean about x? And OK, that means x won't be accurate, which isn't catastrophic, but how much bigger is the error? And of course, the answer is very simple.

So here's our function. And let's say we have a certain x. Then the forward direction is we take the x, we go up to the graph, we read off a certain value, y. And what we're doing is we're getting y from our camera or whatever, and we go in the opposite direction.

So we invert that function. Of course, there'll be problems if it's multi-valued. Then there won't be a unique answer. But for the moment, let's assume that we can invert it.

OK, so then the next question is, what if there is some error? Well, in the forward direction, it's pretty straightforward. Suppose that I take a second value here, then there will be some error up here-- delta y. And what's the relationship between delta x and delta y?

Calculus, derivatives-- OK, you know what the answer is. It's delta y over delta x is basically the derivative of f of x, or at least in the limit as we make those very small. So the relationship between the noise in the quantity we're interested in and the noise in the measurement is just the derivative of that transfer function.

And so conversely, we're trying to go the other way around. We'll just turn this on its head. And now we see that the noise in our result is related to the noise in the measurement by this 1 over f prime.

So clearly, f prime equals 0 is bad. And that's not too surprising. If this continues and becomes flat, that means that the thing we're measuring is not responding to the quantity we're interested in. So no surprise we can't recover it.

But also, if f prime of x is small, that's not so good. Because if we're up here, and we make some change in x, that's going to be an almost imperceptibly small change in y. And conversely, if there's any noise in the measurement, I don't really know where I am. Because suppose the noise in measurement is that, well, that means they have an uncertainty of that size in the quantity I'm trying to estimate.

And that's it. It's just-- this is the noise gain. In a lot of situations, we don't worry about it too much. We have to worry about it in machine vision, as I mentioned, because of the noise in the measurements.

OK, so that's real simple case. We just got scalar quantity. Let's extend this a little bit. We're recovering a vector. So let's suppose-- so this is the forward direction.

There's something we're trying to measure, like the location of a robot manipulator arm endpoint in space-- x, y, z. And we're using a camera or a couple of cameras to image it. And so we're measuring something in the image that we'll call x, which is a transformation of the quantity we want. Now, in that case, it won't often be that linear, but let's suppose we'd have a simple case where there's a linear transformation.

Well, then of course, we just do that. If we can invert that matrix, that's our answer. We've estimated where the endpoint of the robot manipulator is.

But of course, we are also interested in, how good is that answer? And so we'd want to know if I change x a little bit, how much does b change? And so a crude way of talking about gain would be just to take the magnitude of the change in the result and divide it by the magnitude of the change in the measurement.

But that's not-- it doesn't take into account anisotropy. Like we saw in those diagrams, the error may be very low if you go in a certain direction. But it might be quite large in another direction. So the answer is a little bit more nuanced. We should be a little bit more careful. So let's say a little bit more about this.

How do we solve linear equations? We use Gaussian elimination. And if you do that, you come up with some formula, which includes the inverse of the determinant.

And so the conclusion is that determinant equals 0 is really bad, because then you can't do this. And actually, the magnitude of the determinant being small is also not that good. Why?

Well, because you're going to take 1 over some small number, gives you a large number. And then multiply your experimental measurements by that large number. And so any tiny, little deviation in that is going to be magnified by the inverse of the determinant.

And let's just go back to an example that we did do, which was 2D. So our matrix in this case is 2 by 2. So as a start, just to refresh your memory-- and I did this last time, but I think I got a lot of blank looks, and so I'll do it again.

So what's the inverse of this 2 by 2 matrix? It's that. And how do I know that, or how can I verify that? Well you just need to multiply. And what do you get? ad minus bc, and then ab minus ab plus ab. And here we get cd minus cd. And then finally, we get minus bc plus ad.

So these, of course, are 0. And these are the same as that. And so for 2 by 2 matrices, we think it can explicitly get an inverse. And of course, we've got a lot of interest in that quantity. Because if that quantity is small, than our calculation is subject to amplification of error.

And this came up because we were looking at computing image motion. So we had-- at two pixels-- we had our constraint equation, which looked like that. So he had two of these.

And now, we're solving for the motion u and v. And so of course, we just get-- just using the formula up here and so on. So we just plug that in.

So there's an example where we can apply this idea of noise gain directly. And we know that if this quantity is small, we're going to be amplifying noise a lot. And we already went through the argument that means that the brightness gradients at those two pixels are similar.

They are oriented in the same or almost the same direction. And so that means they're not providing very different information. It's almost as if you'd only made one measurement. So no wonder the answer's flaky.

And we can go into the velocity space diagram. Each of these constraints-- this is a straight line. Why? Well, because it's a linear function of u and v equal to 0. So that's going to give us a straight line.

This is going to give us a straight line. We're basically looking for a point in the plane that's on both lines. So we're looking for the intersection of the two lines. If the two lines are at right angles, that's a very well-defined point.

If the two lines are almost parallel, then you can imagine that any small shift can move the point of intersection a lot. So this is the case that's not so good. And if I move this one a little bit, say I'll move it over here, there's a huge change in the intersection point.

So that corresponds to the case where the two gradients are almost parallel, and that makes this quantity small. Notice that not all hope is lost. It's true that the component in this direction-- we don't really know very well. But we've got very good constraint in that direction.

So that's another lesson, which is we may have a situation where the noise gain is high, but it may not be equally high in all directions, as we saw in the diagrams I showed you. And it's good to know which component can you trust. If I'm going to have my robot move over and pick up a part, that's a useful thing to know. OK.

So let's review a little bit, and I want to go on to something called time to contact. We're kind of cutting across the material in a diagonal way. I mean, it would be nice to present all of this stuff first and all of that stuff second. But you'd probably fall asleep.

I want to motivate you by showing you that if you put these pieces together, you can get some pretty powerful results. So let's go there. Let's see if we can, first of all, review what we've done so far.

So we have this idea-- the constant brightness assumption. And we have that image solid, which was a function of x, y, and t. And we followed some curve through here, which is perhaps the image of some particular thing in the environment that moves or the camera moves.

But we're assuming that as it moves, its physical properties don't change, so it's still going to be imaged with the same brightness. So along this curve, this holds true-- the total derivative. And from that, we got our constraint equation. You're going to get probably pretty tired of seeing that one.

And so this is the constant brightness assumption. And from it, by just chain rule, we get this constraint, which is also called the brightness change constraint equation. And we use this in many different ways.

I mean, it's fundamental to what's happening in an image when there's motion. And then, we looked in particular at the optical mouse problem, which is a simplified version of things we'll be looking at later.

It's simplified in the sense that we're assuming that the optical mouse is working on a flat surface and that the whole image is moving as one. In other words, u and v is the same for the whole image. And that's obviously a very nice and simple case.

And we said that one way of dealing with that is to turn it into a least squares problem. So we're going to add up over the whole image in x and y direction, we're going to integrate-- or in the discrete case, take some of the rows and columns-- of this quantity, which supposedly is 0. So we should get 0.

Why might it not be 0? Well, if you plug in the wrong values for the velocity, u and v, you won't get 0. And so one way of trying to find the right velocity is to find the minimum of that integral.

Now, in practice, your measurements-- Ex, Ey, and Et-- will be corrupted by noise. And so you'll never actually get this integral to be 0. So the answer isn't we're going to find the place where it's 0. The answer is, we're going to make it as small as possible. And that's our estimate of the correct value for u and v.

And we could justify that by some hairy, probabilistic statistical argument, but I think it probably is not beneficial to go there. We'll just-- it seems like an intuitively right thing to do. OK, so that's what we're going to minimize.

And we only have two parameters, so clearly this is a calculus problem. We just take the derivative and set the result equal to 0. So we're assuming that this is varying smoothly with u and v, as it obviously is. Yeah?

Well, OK, so let's put it this way. So suppose that I have made it Ex, Ey, Et, and now you tell me u is 1 and v is 2. And I can plug it in to the equation, and I get some large number. I'm likely to say, I don't think you're right.

And then he says, well u is 1 and v is 3. And I plug it into this equation, I get a smaller number. Which one would you trust? So it's-- ideally, this should be a very small, or even 0. And--

AUDIENCE: And why is that the case?

BERTHOLD HORN: Sorry?

AUDIENCE: Why do you want a smaller number for that?

BERTHOLD HORN: Oh, well, because if there was no measurement error-- if there was no error either in the measurement or in your knowledge of u and v, then it would be 0 at every pixel and so then integrating over the whole image. OK. All right, and I'm not going to do that, because that's too close to what you did in the homework problem.

So we'll stop there. OK, what are we going to do? Well, a very simple case is where the velocity is the same everywhere-- optical flow case. And we did that.

The other other simple cases where u and v are not constant, but they vary in a very predictable way in the image. And we saw that last time when we were talking about the focus of expansion. And so let's, again, review and go back to all the way to perspective projection.

Right, so we had a relationship between world coordinates and image coordinates. So the capital letters are coordinates in the world, and the small ones are corresponding in the image. So that was perspective projection, again, in a camera-centric coordinate system.

And then we said, well, now suppose things are moving-- big X, big Y, bz are changing. And how are little x and little y changing? Well, we just differentiate with respect to time. And we get 1 over f, dx dt is 1 over z big X dt.

And then, unfortunately, we have this one as well. Or written in a slightly nicer form-- right?

Because dx dt is the velocity in the x direction. And that's what we've been using the symbol u for. And similarly in 3D, we'll use the big U stand for the velocity in the x direction in the world. And similarly, so we have that.

And then, we looked at the situation where u is 0 and v is 0, and we called that the focus of expansion for that particular motion. And so if we plug in little u is 0, we get a relationship here that is-- oh, where's my x0? So let's do it.

So we got 0 is 1 over zu minus-- and now the big X over big Z we know is x over F. so And this is x0. This is where the velocity is 0. And similarly, for the y direction, so we get x0 is f over z, u over w.

So we have relationship between the velocities, the distances, and where the focus of expansion is. And then, we talked about that quantity that occurs there, which is w over z, and what is that?

Well, easier maybe if we turn it on its head, as we did last time. And of course, big W is the component of motion in the z direction. So it's dz over here. So this is a distance over speed.

And so we last time said the units would be meter over meter per second or second. So this is actually the time to contact-- how long it's going to take before we crash into that object if nothing changes. And that's obviously a useful quantity to try and find. And so we'll try and find it from a sequence of image measurements.

Now, actually, we're going to call this-- we're going to give this a name c. And c is actually 1 over the time to contact. And the reason for that is mostly to make the algebra simpler. After we find c, of course, we can just find its inverse.

But it has the advantage that if the motion is very slight, then the time to contact can be huge, where c just becomes close to 0. Yeah, sorry. No, I hesitated, because it didn't look right. But thank you. OK.

OK, so let's start real simple. Let's suppose that there's only motion in the z direction. So I'm moving towards the wall. And of course, the image is expanding as I approach the wall.

And what is the focus of expansion? Well, if I'm moving straight to the wall, the focus of the expansion will be right down the barrel. It'll be at 0, 0. So let's take a special case.

And then, according to the formulas for x0 and y0-- and if I draw the diagram for the motion field, it's going to look like that.

And so a couple of things. One is, well, if I can measure those vectors, then I'm done. Why do I need to do all this other stuff? Well, the thing is, we don't know those vectors. All we have are images.

Images are brightness patterns. Images are brightness as a function of x and y. There are no vectors in there. So one approach might be we'll find the vectors, and then we can intersect them to find where that point is.

So for us, this vector field is a useful tool to visualize what's going on. But the actual experimental data is there's an image, and then we take another image. And the other image is either expanded or shrunk. And our job is to solve this problem.

Then, I had the arrows pointing inwards, which means that that's actually a focus of compression, rather than the focus of expansion. But of course, they're just depending on the direction of motion. If I'm standing here and the wall is receding with a positive velocity, then it's going to compress in the image, and I get this.

On the other hand-- and there's no danger. The time to contact is negative. I left the surface a while ago.

But the case we're going to be more interested in is where the velocity of the wall relative to me is negative. And then this diagram is reversed, and there is a focus of expansion. OK. So what can we do with this?

So in this case, we go back to the same old equation. And now, in this particular case, U and V-- we've made we've made the capital U and the capital V 0. So all that's left is that second term. And so U and V take this form.

OK, so stick that in there. We get, let's see, U-- we get cx, Ex plus y, Ey plus Et is 0. So I've just combined these two terms. And so if I wanted to--

Just as what we did before, we can measure one-dimensional motion from a pixel. Similarly, here we can measure the inverse of the time to contact just from brightness derivatives at one point in the image. Of course, it's not likely to be very good, because all of these quantities are subject to noise. And in fact, of course, estimating derivatives makes things worse.

So we've got two numbers that are subject to noise. We subtract them. If they're similar in magnitude, about all you got left is the noise. So this is not going to be a very accurate method.

Before we go on, though, let's look at this term over here. And I'm going to call this the radial gradient. And I put it in quotation marks, because it's not the greatest notation. But we use it so much, we need some notation.

And we can think of it as that dot product. We've got two vectors. And what are those?

Well, this vector is the brightness gradient. And we keep on seeing the brightness gradient. And it's just pointing in the direction of the most rapid change of brightness in the image.

One way to think about images, E of x and y, is as a topographic map. So if you think of translating brightness into height, then you can visualize the three-dimensional thing. And the image is some surface in the three-dimensional thing.

And the brightness gradient is just the gradient of the surface. It's a vector in the direction of steepest ascent. If I want to get to this mountain as fast as possible, I go up the gradient. And if I want to get down as fast as possible, I go in the opposite direction.

So that's Ex, Ey. It's the gradient of the brightness. And then what's x and y? Well, that's radial vector. It's like in the polar coordinate system. I could imagine erecting a polar coordinate system in the image with the origin at the center, and that's that vector.

And so what this is doing is it's taking the dot product of those two. So for example, if they're at right angles, then this is going to be 0. And if they're parallel, then they will be as large as possible.

Now, we could-- let's see, which way do I want to go? Let's try this. We could normalize this, and turn this one into a unit vector.

So first of all, you recognize that by dividing by the square root of x squared plus y squared, I turn it into a unit vector. Because now, if you take a dot product with itself, you get 1. OK, that clear? So there's a unit vector.

Unit vectors are useful for indicating direction. So this doesn't have-- well, magnitude's 1, so the magnitude doesn't tell me anything useful. But it's a way of talking about different directions. And then, taking the dot product with another vector does what? Well, it gives you the component of that vector in that direction.

So here's a vector. And I'm interested in how much of it is going in this direction. I take the dot product, and I get that component.

So what this is really computing is how much of the gradient is in this radial direction. And that's why we use that term, radial gradient. And the reason I am putting it in quotation marks is because it isn't quite right, because there's this factor that I've left out-- that it's not just the radial gradient, but it's multiplied by the radius. But it's basically measuring how much of the brightness variation is in the output direction from the center of the image. OK.

And now I'm ready to solve the problem. Because I have this equation here, and I've solved it for a single point, except I don't really trust that. So what I'm going to do is as before, use least squares.

So we're going to minimize-- and this is over the whole image. And again, keep in mind that we're dealing with very simple cases where the motion in the image is not some arbitrary vector field, but it's defined by a small number of parameters. Yeah?

This here? Oh, so that comes out of the equation up there. So there's an equation for 1 over f u is 1 over z big U minus wz x over y-- over z. And first of all, [AUDIO OUT] is now assumed to be 0.

We're assuming that there's no motion in x or y direction, so those terms drop out. And the other term-- we've defined c to b w over z. And then, the big X over z little x over F, because of perspective projection equation.

OK, so yeah, there's probably more than one step going from up there to down there. But it's applying the perspective projection of the equation in the special case that there's only motion along the optical axis. There's no motion in other directions.

OK, so again, what's going on here is if we had the correct value of c and there was no measurement noise, this would be 0 at every pixel. And so if we add it up over all the pixels, it should still be 0. If we plug in the wrong value of c, it ought to be non-zero-- grow with our error in c.

And so finding the c that makes this as small as possible is our way of estimating what it is. And yeah, it can be justified in terms of-- if you assume the noise is Gaussian and do all that statistical, probabilistic stuff. But I don't want to do that. I think it's intuitive that if this is supposed to be 0 when I have the correct information, then making it as small as possible in the presence of noise is a sensible thing to do.

Now, I just-- calculus. There's only one unknown in there, only one knob I can tweak. So I'm just going to see where that has a 0 derivative.

And so well, what do I get? I get 2 times-- there's a square in there. So I'm going to get twice whatever is under the square. And then I have to differentiate that term, as well.

So multiplied by-- and now, the derivative of this with respect to c is, of course, just that part. So multiply by-- and if it's equal to 0, I don't really care about the 2, so get rid of that. And now, I can split this up into two integrals.

And I'm going to leave out the dx dy pretty soon, because it's implied if I have the double integral over the image. OK, so I can solve for c is the integral of-- let's call this G squared. And this is G-- oh, sorry, minus G Et G squared, where G is this radial gradient, which will occur so often that I get tired of writing it out.

So there's a way of estimating time to contact, because c is the inverse of the time to contact. And it's kind of interesting, because there's no high-level stuff in here. We're not detecting edges, or tracking points, or doing anything sophisticated-- recognizing poodles behind palm trees.

We're just doing this brute force number crunching. And it's very effective. We got a million or 10 million points, each of which is lousy. But we combine them this way, and the good to one part in 1,000.

So what when you implement this, what do you actually do? Well, you have to take the image and estimate the brightness gradient, which is trivial. We just take neighboring pixels in x direction, subtract them. There's our Ex.

And then take neighboring pixels in the y direction, subtract them. That's Ey. And we need Et, so we take two frames, one after the other, and we take corresponding pixels and subtract them. That's our Et. Or there may be some scaling, but that's the basic idea.

From those, we then compute this radial gradient, which is easy to do. x and y are the position in the image relative to the center of the image. We'll talk about that some more later.

And then, we just compute these to sums. So these double integrals are just sums over all of the pixels. We just run through all of the pixels adding up these things. And so really, well, if you want to do it sequentially, you need two accumulators.

You set them to 0, then you run through the image row by row, pixel by pixel. And at each pixel, you compute G. You're computing x Ey from it. You compute G.

And then one accumulator-- you multiply G by et. add that to that accumulator. The other accumulator, you take G squared, and you dump it into that one. And then, when you've come all the way through the image, you've got these two sums. You divide them, and you're done.

So it's totally brainless. There's no intelligence there, artificial or otherwise. And I call this a quote, "direct computation." Because there are lots of other ways of approaching this problem.

For example, you might measure the size of-- like you're coming out of a parking lot and there's an MIT bus in front of you. Then you can measure in the image, how many pixels is the image of the bus? And how many pixels per frame in time is it changing, and so on? You can do that.

But it means you have to detect the bus. You have to find the edges. You have to estimate the beginning and end of the bus. And you better measure it with very high precision, like a hundredth of a pixel.

And it's not too hard to find out where the edge of the bus is in terms of pixels, but hundredth of-- and why do you have to measure that accurately? Because it doesn't change much from one frame to the next. So there are other ways of doing this. But this is-- it's brute force, mindless, and elegant. That's my view of it.

OK, now of course, this is very specialized. We're only dealing with a case where we are running straight into the wall. And so we'll look at more interesting cases in a minute. But what I'm hoping to do, if I can get the projector going again, is to demonstrate some of this.

So what I'll show you is a little program called MontyVision, which allows you to create image processing applications by a graphical process of basically plugging together quote, "filters." So let me bring this over here and show that to you for a second.

So this is what it looks like. And plug together the left most thing is a file source, which, in this case, is .avi file, video. Then it goes into a splitter that cuts it up into streams, including going from color to gray level. There's a decompressor, because all the video is compressed, otherwise it would be ridiculously large.

And then it goes into the time to contact box, and it comes out the video renderer over there. And this time to contact box has a set of parameters that show up somewhere off screen right now. OK, so I'll put this back where I can see it on my screen. And let's hope this works. Oh.

OK, so I'll run this several times, because this goes by pretty fast. So here we are running into a truck that's used in the mining industry. It's kind of a large truck. And what we're seeing are a whole number of different things. I'll also try and get rid of this.

Well, let me just run it again. So you be seeing a circle-- red circle-- that denotes the focus of expansion, the estimate of the focus of expansion. And you can see that we're going to hit that tire or that wheel.

And then, you will see the three bars on the right. So the third one is c, the thing we just calculated. And you can see that it's red meaning bad, and it's growing upwards as we approach. It's the inverse of the time to contact.

And the other things, like the time to contact is over there in frames. And we can plot to see how accurate it is compared to when we actually hit the target. So this is slightly different from what we did, because this allows some motion in the x and the y direction, and we'll do that in a second.

And as a result, it has three qualities its computing, which we'll call a, b, and c. And those are the three bars you see on the right. And if we were going straight down the barrel, the first two would be 0. They wouldn't be either up or down, but they're not.

There's an x component and a small y component that-- and of course, this is noisy. So it's not perfect. OK, let's look at a different one then. Newman--

Well, I guess I'm obsessed with running into trucks-- one of my nightmares. So here's another truck we're running into. The circle, again, indicates the focus of expansion. The three bars are a, b, and c. And the rightmost one is the one that we now know how to calculate.

And I'll show there's a couple more times. And it's growing as we go, because the time to contact is getting shorter so 1 over the time to contact is getting larger. The time to contact computed is down on the right-hand side.

Now, it turns out that if we were to do this frame-by-frame, you would notice that near the end it just is wrong. And so why is that? Well, there's a whole bunch of different reasons. One of them is that the image becomes out of focus and so the information is changing.

We're kind of assuming that the image is just zooming, but in the system that has a lens, unless we adjust the lens to stay in focus, we're going to go out of focus. Another point is that initially, the image motion is very small. It's a fraction of a pixel between frames.

But then, of course, as we get close to the object, things are really exploding. And so our whole assumption about estimating derivatives by taking neighboring pixels and subtracting them and all of that stuff is not working very well.

And so how do you deal with that? Well, one way is to combine pixels so that on the super pixels, the motion is a small number of pixels. And we know that's when this works. So what we could do is run this at multiple scales.

So you have the full resolution of the image. And when things are far away, that's the one that's going to give us the best information. Then we, for example, average 2-by-2 blocks of pixels. Now we have an image which has a quarter as many pixels. And it can cope with twice the image motion, because a certain amount of image motion in the original image now corresponds to half as many pixels in this reduced one.

And then, we do that again. We take that reduced picture, we again-- 2-by-2, average and now we're down to a 16th. So this gets very cheap to compute compared to the original image. And so there's no real cost-- additional cost to this. Well, some.

So what is 1 plus a quarter, plus a sixteenth, plus a sixty-fourth plus-- OK, well, you know how to sum geometric series. And the answer is-- I don't know-- 4/3 or something. So yes, it costs more, but it's not a huge cost. You can afford to do this at multiple scales.

And that allows you, then, to cope with all of the velocities, starting off with a very slow sub-pixel motion to where things are really blowing up. OK, well let me do one more thing and hook this up to the camera. Hopefully that'll work.

Oh, don't do that to me. OK, so here's a web camera. And I guess it's relatively dark in here. Let me try the paper. Out the focus.

Trying to find something that will give it a nice texture to work with. I guess there's a chair which has lots of holes in it. Let's try that. OK, so I'll try and hold it as still as possible.

So again, the three bars are a, b, and c. And they're all over the show, but relatively small. Now, if I move the camera up, that third bar should go down and be green for safe, meaning the time to contact is negative.

If I move it down, that third bar will go up and be red, meaning I'm about to crash into something. And I think the reason it's got this jerky motion is because it doesn't like this projector. So it's got a very slow cycle time. It should be running at 28 frames per second, and it's obviously not. OK.

Now, if I try to move in x-- let me see if I can move in x-- that first bar should go up. Oh, I'm not holding it straight, so I'm getting the crosstalk between x and y. Right now I'm moving in y. And I'm moving in the opposite direction in y.

So that second bar goes up, and the second ball goes down. And here, I can move in the x direction-- the first bar goes up, and the first bar goes down. And I mean, to me, this is fascinating, because it's such a neat instrument. And it doesn't have any magic in it, and we can understand its limitations.

We can calculate what the error is and so on. It's not like something that's very elaborate and has some hidden code in it. So let me show you that version. Let's see. Oh. Oh, OK. Oh, I see. It's closed.

So one of the features of this code is it can show you some intermediate results. I need to look at the control panel for this to see what it's showing right now. So right now, it's sub-sampling 2 by 2. And let me show Ex. Let's get rid of this.

OK, so Ex is just the derivative in the x direction. And positive is shown green, negative in black. And it's fairly weak in most parts of the image, except where the black line is.

There, on one side of the black line, there's a rapid change in brightness up. And on the other side of the black line, there's a rapid change in brightness down. So that gives you the green and the red fringes in the x direction.

Now, if I turn it so that that black line runs horizontally, there'll be less, because the x derivative now is very small. I see that, because it's more or less constant in the x direction. Whereas, if I look at the y derivative-- oh, I can actually show x and y together in a slightly different presentation.

So here, the x derivative is controlling red, and the y derivative is controlling green. And you get this funny kind of almost three-dimensional feeling about the result. But the gradient is showing the direction of most rapid variation. So think again of what we said about the analogy with a surface-- a topographic map. And that's why this looks a little bit to us like it's three-dimensional.

OK, so those are the basic things we compute. I can also show, instead, Et. Now, Et, if I was solid as a rock, would be 0 everywhere. And obviously, I'm not. Let's see if I can hold on to something to make it more constant.

So you can see it decreasing in magnitude, although I can't seem-- OK. So that's a time direction. We're subtracting successive images. And let's see, let's look at G.

So G is that radial derivative, x. And what can you say? Well, it goes outwards. You see that square box? It's green all the way around, which is not what Ex would do, because Ex would be opposite on the two sides. But because we're multiplying by the vector from the center of the image, it actually ends up being green all the way around.

Now, we can do one more thing, which is to multiply it by the time derivative Et. So in that formula, you may remember that there's a product of ET in this brightness gradient. And obviously, if that is 0, it will say there is no motion.

If there is motion, then it'll be non-zero. And in particular-- hmm. It's unfortunate that it's not-- oh, OK. So here you can see that in most places-- I wish it was running faster, because it's very hard to-- in most places it's now green, because I'm moving away from the surface. When I do this, in most places, it'll be red.

And the reason the other areas is because it's jiggling back and forth. I'm not a very good manipulator. If I had a robotic manipulator, we could do this better. OK, so anyway, anything else we want to see on this? So before I put that away, are there any questions? Yeah.

You've got a very limited, very constrained problem, which is the problem of the fly landing on the surface or someone running into a wall. If there are multiple motions, this isn't going to work. And we're going to do that.

So we're slowly expanding. We started off with one unknown, which was just the inverse of the time to contact. And now we're going to do a little bit more. But ultimately, what we'd like to do is deal with arbitrary motions, where u and v are not constrained by just a few parameters, but they could be different all over the image.

So u and v, right now, are some simple function of some constant. Where is the equation? Over here. And what we'd like to eventually get to, after a few more stages, is where u and v can vary arbitrarily over the whole image.

And you can see how that's going to be problematic, right? Because at every pixel, we have our magic equation up there, the brightness change constraint equation. At every pixel, we get one constraint. So if we have a million pixels, we have a million equations.

But we have 2 million unknowns, because at every pixel, we'd like to know u and v. So that doesn't sound very good. So we'll have to introduce some additional assumptions to solve that problem. Because right now, it looks like we have a million equations and two million unknowns. We know what the answer to that one is.

But fortunately, in most cases, there is additional information. For a start, when I'm moving around the room and you're moving in your seats, in most places in the image, neighboring points are moving not the same, but very similar. And so if they were moving the same, then we could easily integrate that with our approach. If they're similar, it's a little bit harder.

But that's what we're going to do. We're going to say that we're trying to recover this vector field in the situation where things can move independently but there is some-- it's not like you're looking at-- oh, you don't remember this. But it used to be that late at night TV stations would go off the air, and there was no signal. And then the radio receiver would be basically putting white noise on your screen.

So you'd just see this-- every pixel is totally unrelated to every other pixel. So that's a situation we're not going to be dealing with, because that's not a natural situation. As I'm doing my daily business of eating and whatever, I'm dealing with a situation where neighboring pixels typically have almost the same velocity.

OK, well let's go back to this and try and generalize it a bit. So the next most general thing to do is to say, OK, let's have motion in U and V in the x and y direction, as well. So no longer will U and V-- big U and big V up there-- be 0. But there'll be-- we'll allow them to vary, as well. So let's see.

Before I go there, let me just say something about-- I totally missed this. But everything we did so far, we got the answer, and then I said, oh, but how is this going to fail? We haven't done that here.

So you remember that this was c equals the ratio of two integrals. And of course, it's going to fail if the integral of G squared is 0, for example. That's one way it can fail.

So what does that mean? Well, that means that there's 0 radial gradient everywhere in the image. And one way for that to happen is you're in a coal mine without lights. E is 0. That's not a particularly interesting case.

The other one is that the brightness-- the radial derivative-- is 0 everywhere. That means that the component of the gradient in the radial direction is 0. So we have this expression over here. This is 0.

And that means that-- let's see. That means that the radial direction and the gradient are at right angles to one another. That's how we get the dot product to be 0.

So let's try and draw a picture like that. So here's an image. And say we're there. Then the radial direction is that. And we're saying that the gradient has to be at right angles to it so that the brightness can change in that direction, but it can't change in this direction.

So in this direction we can change. We can't change in that. And similarly for any other direction-- we can have the brightness change this way but not radially. So what sort of pattern would do that?

How about a bullseye? Is that-- it will have variation in the radial direction. So that doesn't work. So if it's not a bullseye, turn it 90 degrees everywhere. An x will do. Sorry, what did you say?

AUDIENCE: Rotate it.

BERTHOLD HORN: If we rotate it-- hmm. If we draw a slice and rotate it-- pie charts. So if we have a pie chart, then everywhere the gradient is in the rotating direction, there's no variation in brightness inwards and outwards. So this whole pie sector is one color. This is another color, and so on.

And so that's a case where this will fail. And does that make sense? I mean, would you expect it to fail? Why-- could you do better? So you're looking at-- you see nothing else in the world except this pie chart, and you're moving towards it. Well, the-- shaking of heads.

The image doesn't change. I mean, we're making assumptions, like you can't see fine, little specks on the surface that would give you a clue. Assuming it's perfectly smooth and you just have these, if you magnified by a factor of 2, it looks the same. So that's perfectly consistent.

This method isn't going to work. And it can't. It doesn't-- there's no information there for it to work on.

So we'll come back to this issue when we generalize it. Again, looking at the case where it'll fail, and is it reasonable for it to fail? Could we do better?

Now, in a lot of cases, we can do better, just because, if I look at a piece of paper, it's not perfectly smooth. They're tiny fluctuations, there are little fibers. And so this overall pattern might look like this, but there are tiny little specks in there, and I can pay attention to those.

And similarly, this algorithm could, if the contrast is high enough of those little specks, it would pick up a non-zero Ex, Ey and it would calculate the time to contact. OK, so more general-- slightly more general case.

OK, so now we're allowing u and v to be non-zero. And so we've got u of f is-- I'm just copying the equation from over there.

And for convenience, I'm going to multiply through by f. OK.

So I'm just defining two new variables. So we've already had c was w over z, which is the inverse of the time to contact. And now, we're defining two more, which are fu over z, which is f times x0. And fv over z, which is f over y0.

Why is that? What does that mean? Well, what we're going to find is time to contact and FOE. But it's more convenient to define these variables that are functions of time to contact and FOE, because then the equations are simple. We only have a few terms in them.

So this one here is just f times the x component of the focus of expansion, and this one is just f times the y component. So once we know a and b, we can calculate where the focus of expansion is, assuming that we know f.

One thing that I didn't mention is in our formula here for c, f doesn't show up. So that's kind of interesting, because it means that we don't need to certain properties of the camera in order to do that computation, which is surprising, because for many purposes, you do need to know.

For example, if you take the approach, OK, time to contact is distance divided by velocity. Well, to compute the distance, you need f. To compute the velocity, you need f. So it's a very pleasant surprise that to compute the time to contact, we don't need it.

Intuitively, what's going on is that if I approach a wall, it's going to loom outward. And if I approach it with a telephoto lens, it's still going to loom outward. And if you actually go through the perspective projection equation, it's going to increase in size at the same rate.

And you're right that that is big X over big Z. So I've applied the perspective projection equation here. OK, so now we have u and b as functional [INAUDIBLE] unknowns, and we plug them into our favorite equation.

And let's see. We get--

So that's our brightness change constraint equation. And I'm going to, again, call this G to cut down on the amount of writing the radial gradient. And I'll use the same method to try and find the answer.

OK, so we have this expression, which, if we had the correct values for a, b, and c, is 0. If we integrate over the image, it should still be 0. If we have the wrong values of a, b, and c, it won't be 0. And so we formulate the problem as making this as small as possible.

Now, in the absence of measurement noise, we could actually make it 0. In practice, Ex, Ey, and ET will be subject to measurement noise, so we won't be able to quite make it 0. But it's still a valid approach to pose it as this kind of least squares problem.

OK, and of course, only a finite number of parameters-- a, b, and c. So it's a calculus problem. I keep on saying that, because later on, we're going to be looking for functions, not parameters. And then we can't use calculus-- so can't differentiate with respect to a function-- is the short version of that story.

OK, so what happens if we differentiate this? Well, the derivative of that integral is the integral of the derivative of the integrand. The integrand is this square thing. So we get 2 times that. So we're going to get 2 into--

So we have 2 times that, and then we have to take the derivative of that term in here with respect to a. And the only thing here that depends on a is this part. So that's just Ex.

And then, we repeat that. And again, we take the derivative of the integrand. So we get twice this thing. And the derivative of the term here with respect to b is obviously just Ey. So and then the third step is-- I'm not going to write this out again.

And the only thing that depends on c is cG. And so we take the derivative and we just get G. So we get three equations, and magically, we can actually solve them.

So these simple cases have a wonderful feature that there's a closed-form solution. I can write out what the answer is. And that's invaluable, because yeah, you can always numerically calculate stuff. But it's very hard to say things about the solution.

Like, suppose I want to tell you that there's only one answer. Well, if you have some numerical way of finding the minimum, there could be another minimum somewhere else. And if I have an analytic solution, the closed-form solution, I can say things like, how sensitive is it to noise?

Well, I can just differentiate with respect to the thing that has noise in it. If I do it numerically, I'll have to recompute. And then you say, OK, that's fine. That answer is OK for this set of parameters.

What about different set of parameters? You have to recompute. Whereas if it's analytic form, there's the formula. There it is. So unfortunately that doesn't happen all the time. Once we make things complicated enough, we typically can't find a closed-form solution.

OK, so what do we do with this? Well, the integral of a sum is the sum of the integrals. So we can rewrite this like this. I'm leaving out the 2, because that doesn't make any difference to anything.

So I'm just multiplying out this term by Ex and then splitting it up into several integrals, because the integral of a sum is the sum of the integrals. So that's going to be GEx.

OK, so that's one equation. And you'll notice it's an equation that's linear in a, b, and c. So that right away gives us hope that we can solve this problem pretty easily. And if I repeat for the second equation--

So three linear equations and three unknowns. And before we go on, a couple of things. One of them is, you should recognize parts of this.

That's what we had before when we were going straight down the optical axis where a and b are 0. Then we're just left with this part. We just had one equation and one unknown, set that was nice and easy.

Another thing is that the coefficient matrix is symmetrical. So this coefficient is the same as that one. And that coefficient the same as that one. And this one is the same as that one. And often having a symmetrical matrix gives some advantage in terms of understanding the stability of the solution.

So what are these things? Well, how do we do this, before we even start to solve the equations? So let's look at the coefficient of a up here.

Well it's just the integral of Ex squared over the whole image. So we go through the image, and at every pixel, we look at the neighboring pixel. We subtract the gray levels to estimate E sub x. Square it and add it to an accumulator. And we do this for all the pixels.

Then we have another accumulator where we estimate Ex and Ey by looking at neighboring pixels in the y direction, multiplying those two differences. And then we have a third accumulator.

And G, we have to do x times Ex plus y times Ey. And then we take the result, multiply by x. We throw that into that accumulator.

Now, we don't need to do this one, because it's the same as that one because of the symmetry of the coefficient matrix. We do have to accumulate that one. We have to accumulate this one and that one. And again, these two, because of symmetry, we don't need to do those.

So we run through the image, and we have six accumulators that we just add to as we go. And of course, you could parallelize this, because they don't interact. I mean, the operation I do on this pixel, in this case, is completely independent of the operation I do on that pixel.

So if you have a GPU, you can do a whole bunch of pixels at once. You can dramatically speed this up. Although, you saw it running on a silly ThinkPad, and I was not doing anything fancy. It's using software that's five years old.

So even with that, if it's not hooked up to this projector, it gets 28 frames per second. So it's not critical that you paralyze it, but if you wanted to, you could. And then you could run it at thousands of frames a second, which, by the way, is what optical mice run at.

They're typically running at 1,800 frames per second, or in a gaming mouse, even higher. But the images tend to be smaller. The images are often only 32 by 32.

OK, so we accumulate those six. Notice that these do not depend on any changes in time. These only depend on the brightness pattern, the texture that's in the image. Then we need three more which do depend on changes in time. And that's it.

So we have a total of, let's see, nine accumulators we have to keep track of. And then we get to the end, and ther we have to solve three equations and three unknowns. And I guess I ran overtime. So are there any questions before we depart?