[SQUEAKING] [RUSTLING] [CLICKING]

PROFESSOR: End of the photogrammetry section. We'll just briefly talk about exterior orientation, which is the fourth of the photogrammetric subjects we are talking about.

So what's this about? This is best illustrated by thinking about a drone flying above some terrain of which we have a detailed model. So we know where points are in some global coordinate system. And we have a camera perspective projection. We get images of those three points. And the question is, where are we?

So p1, p2, p3 are known. Let's assume that the center of projection is p0. That's what we want to find. And we also would like to find the attitude of a plane in the world. So it's the same old thing, rotation plus translation, except in this case, we have a mix of 2D and 3D information.

The coordinates of the points in the world are given in 3D. So we have a terrain model. The points corresponding in the image are in 2D. So let's see. One question right away is, how many correspondences do we need? So we know that we're looking for six degrees of freedom.

And so one question is, how much does each image point contribute? How much constraint does it provide? So we've got images of-- so here's our image. We've got three points. And naively, we can just say that well, every time we measure where something is in the image, we've got x and y, so two numbers, two constraints.

And so with three of them, we should have enough constraints to solve the problem. So we need three or more. And in this case, that naive argument actually is correct. We only need three. And this problem, I guess, I don't know if it was Church to first solve it. Or in any case, he wrote it up in a textbook. I don't know, 1950s. So it's an old photogrammatic problem well known.

Of course, machine vision people didn't read that stuff. And they reinvented all of photogrammetry. And they actually did kind of a poor job of it, came up with something called projective geometry, for example. So anyway, we go back to the roots, which is photogrammetry.

OK, now one thing you might say is, suppose I don't care about the attitude. Then I only have three degrees of freedom. Can I solve that problem? Well, unfortunately, these are coupled. So you can't cheat and just solve for the variables you want.

OK, so what do we know? Well, let's assume that interior orientation is known. So that means x0, y0, f are known. And so then when we have a given point in the image, we can just connect image point to center of projection. And we have a ray in space. And we know that the object is along that ray in space.

And we have that ray in the camera coordinate system, right? So with the origin at the center of projection, and so on. OK, so if we have these three rays, it's like having three sticks. And you're trying to arrange for the sticks to go through three points in the 3D world. So you move around the position of the center of projection until that happens. That's what we're doing.

So if we have the rays, we can calculate these angles, so the three angles. OK, so those are known. Once we've got our image points, we can construct those rays. And so we can just take the dot products to get the cosine. And we can take the across product and take the magnitude of that to get the sine. And that gives us a-- 8 and 2 will allow us to calculate the angles.

OK, so we know those. So what don't we know? Well, what we don't know is the length of these legs of the tripod. So one way we can think about this is that our task here is to find r1, r2, r3. And we're not quite done once we've got r1, r2, r3. But we're just about done. Because then we can construct p0 by intersecting three spheres, right?

So if we know r1, we know that the plane is on a sphere with radius r1 about the point p1. We know r2. We know it's on a sphere with radius r2 about point p2. And two spheres intersect in a circle. So we're not done. We haven't quite solved the problem.

So then we take the third one. There's a sphere of radius r3 about p3. We intersect that circle with that sphere. And we get two solutions typically. So there's potential ambiguity. But basically, we then have solutions. And if we have more than three points, we can resolve the ambiguity.

You can also see there's going to be some ambiguity just by thinking about, suppose we have the length of those three tripods. Is there some other position than the one I've shown, where those lengths would be exactly the same? So think about moving that airplane somewhere else. We're claiming that with three, we've got a solution possibly however, more than one solution. So work with the plane B. And those lengths would still be the same as they are there.

Well, if I move it a little bit, that's not going to work. Because then I'm going to screw up one or the other of these lengths. If I move it to the left, I increase r3 and decrease r1. But imagine that we're flying on the ground. Then we can have a mirror image of the position of the plane. And we get exactly the same three lengths for the tripod.

So if we draw the plane that contains p1, p2, p3 and think of that as a mirror, and then we mirror image that plane, then that has the same length. Now, of course, that's one way to disambiguate it. Because typically, planes don't fly on the ground. So we can resolve it that way.

Also, there's an issue about the cyclical order of the images that would be different if we were looking at it from underneath. It's like looking at some writing from the wrong side. It's mirror imaged. And similarly here, if we look at it from the mirror image position of the plane upwards, everything is a mirror image. So we can resolve it that way.

So if we only have two solutions, then we can easily get rid of the problematic one using some argument like that. OK, so how do we find r1, r2, r3? Well, there used to be books full of formulas of triangle solutions. So if you know one side and two of the angles, or if you know two of the sides and one angle, all sorts of combinations. And why was that done?

Well, because it was important for navigation. And it was important for surveying. So people used to know these things, but not so much right now. So they are in the appendix of the book. And the appendix is on Stellar. For example, there's a rule of sines, which is just that a over sine a is b over sine b.

And there's the cosine rule. Basically, those are the only two you need. You can solve all these problems using those two rules. Sometimes it's convenient to have some of the other rules. Because they make the job shorter. OK, so our problem is we have this angle. We know that.

What else do we know? Well, if we the digital terrain model, then we know this distance. We can calculate that similarly for that one, and so on. So in this triangle, we know that angle. And we know this distance. And the question is, what's r1 and r2?

Well, that's not enough information to solve using the sine or the cosine rule. But we can write an equation involving the unknowns, r1 and r2, and all of these known quantities. And I won't be doing it. Because it might come up in the quiz.

So the result is going to be that we have three of these triangles. We get one equation out of each of them. So we're going to have three nonlinear equations in the three unknowns, r1, r2, r3. And then we can talk about solving. It might not be easy to find a closed form solution. But at least we can talk about how many solutions there might be.

And numerically we can always solve that problem. OK, that's r1, r2, r3. And then, as I said, we still need to intersect those spheres, so a little bit of algebra there. And then we have the position of the plane. If we have more sightings, more correspondences, that's always better. Then we can formulate the least squares problem.

And if we're worried about outliers, we can use ransack. Suppose we have 10 correspondences. Then we might pick three correspondences to get a solution, take a different set of three correspondences to get a solution, and so on. And so we've done all of that in other contexts. So I won't want belabor that.

So what's left? Well, what's left is finding the attitude. What is attitude? Well, it means it's the orientation relative to the ground coordinate system. So there's a rotation we need to find. Well, the thing is that once we know where the plane is, the center of projection p0, then we can construct these vectors in 3D in the ground coordinate system. Just subtract p0 minus p1 or p1 minus p0 and p2 minus p0 and p3 minus p0.

So we can construct three vectors. For example, we could say-- And this gets rid of the translation. We're only interested in the directions. But we also know these vectors in the camera coordinate system. So based on the image positions here, we connect those up to the center of projection, which is the origin in the case of the camera. And we get three vectors in that coordinate system.

So they correspond-- three vectors in the camera coordinate system to those three vectors in the world coordinate system. So that's pretty heavy constraint. That means that we should be able to relate those two coordinate systems. And we know that we've taken out the translation.

So all that's left is a rotation. And so depending on which way we go, are we interested in a transformation from the camera coordinate system to the world coordinate system, or from the world coordinate system to the camera coordinate system? But we're going to end up with something like this.

Now, of course, these vectors may have different lengths, if we just do these subtractions. We're only interested in the direction, so unit vectors. And so-- we expect that. And as I said, it could go the other way around from the world coordinate system to the camera's.

So we have three equations like this. And what we're looking for is r. And let's represent it as an orthonal matrix, in this case. Well, we can stick these three equations together into one matrix equation, right? So we have-- now, all of these things are now known. We have the interior orientation of the camera. And so we know how to construct the three vectors from the center of projection to the three image positions.

And we've calculated where p0 is. So we've got those three vectors. And so what is this? Well, this is a three by three matrix. And this is a three by three matrix, the first column of which is the vector A1. Second column is A2. Third column is A3. So we have a product and all three by three matrices.

And we can just solve for r by inverting one of them, so very straightforward. Interesting question is, is the result also normal? So I'll leave that an unanswered question. OK, yeah.

Is in the camera coordinate system. So it's p1 prime, p2 prime, and P3 prime. So A1 is p1 prime. Because in this coordinate system, the other hand's the center projection of this origin, 000.

Prime subtract. So that's the ray to that point in the environment as seen in the camera coordinate system. And as I said, you know, this is the minimal case. We just have three correspondences. In practice, we would like to have more, get better accuracy. And then we use least squares. And there's no longer a closed form solution.

But we can use this to get started. Just pick three of the answers, three of the correspondences, to get an initial guess, and then have some iteration that minimizes the least squares errors. Just make sure that what you minimize is in the image plane. Because this is where the measurement error lies, not some other arbitrary.

So there's that. Now, suppose that I mean, this doesn't need to be a plane. It could be some tourist camera in-- I don't know-- some famous square somewhere in the world. And maybe there's another tourist with a camera. And so there's a related problem.

So here is one camera position. And here's some famous sculpture, cathedral, whatever. And here is another camera position. Here's another camera. And there could be hundreds of these. And you may have seen the results of this. And in this case, we do something called a bundle adjustment.

And again, this is an old photogrammetry problem, which machine vision people have rediscovered and made a hash of. But they did finally get it right. And so what's the problem? Well, it's a nonlinear optimization. And the method we proposed, we talked about, Levenberg-Marquardt, is a good one to solve nonlinear optimization problems.

So what are the unknowns? Well, the unknowns are a set of points in the environment of which you may have more than one view. And we don't know where they are to start off. And so part of the problem is finding where they are in some world coordinate system.

What else is unknown? Well, the other thing that's unknown is we don't know where the cameras are. So we need to allow for the cameras to move around. Those are unknowns. And then, well, the cameras have an attitude in space. So there's a rotation. I'll just write it that way.

So we tweak things to make the errors as small as possible. And again, the errors are the errors in the image, not out in 3D. So what else? Well, in most cases, we don't know what the camera properties are. So we need to also do interior orientation, and maybe, if you want to be accurate, some allowance for radial distortion.

So assuming we have some initial guess, then it's just a matter of pumping it into this black box, which minimizes the error by tweaking all of these parameters, lots of parameters. But presumably, you've got lots of constraints, lots of pictures. And so people have made incredible reconstructions of all sorts of things, and not necessarily from multiple cameras, but for example, one camera flying on a drone.

So there's some volcanoes in the African rift zone that are very rarely visited. And somebody flew a drone over one of them and made a very detailed complete 3D reconstruction of the current shape of the caldera on that volcano. And this is the method. So when you're flying the drone, you don't know exactly where you are. But you do know approximately. And that helps start the solution.

So any time you have this nonlinear optimization, you want to be near the solution or you might get sucked into a local minima that's not the global solution. OK, there's one thing we haven't really talked about a lot, which is how do you find these interesting points?

We talked about in detail how to find edges. We haven't said much about interesting points. If you want to, there's an online resource on Stellar that describes one of several methods that do that. And it's by Lowe, who was the guy who patented the original method. And since then, there have been lots of alternative methods that do not violate the patent and are faster, and maybe not as accurate, or maybe more accurate.

There's a whole industry coming up with identifying areas that are likely to be easy to find again in another image and describing those areas. So that you can do a good match and find them again in another image. OK, so that's bundle adjustment briefly. I mean, there's a whole industry on that.

But we have all the basics. By going through all of the other photogrammetry problems, we've developed all of the tools you need to implement something like this. OK, switch the topic. So we worked our way up real low level stuff, filtering, aliasing, subsampling, edge detection, and so on. And then we did the photogrammetry. We also did some work in 2D on recognition and determining position and attitude.

So let's try and do that in 3D. And it's obviously, not going to be as good. If you have *Robot Vision,* this is chapter 16. But you don't need *Robot Vision.* Because there's an online resource specifically on this topic called *Extended Gaussian Images.*

So what are you trying to do? Well, we're trying to describe 3D objects. If the 3D object is polyhedral, that's not that hard, lots of interesting representations. We can get the coordinates of the vertices and then construct a graph showing which vertices are connected to what, and then maybe talk about the faces. And each face is connected to the vertices on that face in the graph and to the edges of the face. And each edge is connected to the two vertices and to the two faces that come together.

So you can imagine a nice, typical, computer science solution involving some linked data structure. So polyhedra aren't that difficult. So we'll not say a whole lot about them. And in a derogatory way, they've been called blocks world problems, like children's blocks. So that's where we started in the 1960s. And hopefully, we've progressed away from that.

So what other representations can we have? Well, we look at graphics. And they typically will use measures, which are just perfect for that application. So we can approximate any curve. So we're interested in curved surfaces, now that we've given up on talking about blocks world.

And we can represent any curved surface with whatever precision we want by approximating it by a polyhedral surface with lots of facets. And that works great for rendering. Because for each of those facets, we can determine the surface normal easily by just taking a cross product of two of the edges. And we can get the edges by subtracting vertices. So it's a very straightforward calculation. And then we use a reflectance map, or something like that, to figure out how bright to paint that little facet, and so on.

So it's very convenient for output of pictures. But what about the things we want to do? So what is it we want to do? Well, a couple of things. We'd like to find where things are and how they rotated in space. And I guess we call that pose. So that's position and orientation.

And the other thing we might want is to do some recognition. And so let's see how well this representation works. Well, we could try and do alignment by taking two meshes and trying to bring them into alignment, which would mean we'd have to sort assign a vertex in one alignment in one of the objects with a vertex and the other one, and maybe minimize the square of the distance between them.

But it's not very meaningful. Because these vertices don't have any particular meaning. It's not like they have a label on them that's meaningful. And if you digitize the surface again, what are the chances of getting that particular mesh? Zero, or close to zero. And so for alignment, this isn't particularly useful. I mean, you can do something. You can say, OK, I'll do an iterative thing, where I have approximate alignment. And so for each vertex, I can find what's currently the nearest vertex, and try and reduce that distance, and hope that it'll converge, that process.

But it's not great. And for recognition, you can't even say, OK, this has 320 facets. And the other one has 360 facets. So it's probably the same object. It's not. So you need to do more. And there are ways of progressing from that representation and helping deal with alignment and recognition.

But we'll look at a more elegant method, which has some limitations. So this is not a problem that has been cleanly solved for all possible situations. So what are we looking for in a representation? So one thing is this is like physics. We'd like to understand invariance and symmetries. So what kind of invariance?

Well, what I'd like is that if the object moves translation, then the representation doesn't change, well, in a significant way. For example, if it means that all the x-coordinates are incremented by some fixed number, then that's not invariance. But it means that I'm keeping a representation that is changed in a very simple, understandable way.

So that's translation. And one way I can deal with that is say, well, just reference everything to the centroid of the object. And that gets rid of the translation component. And I've solved the invariance problem.

Then rotation, OK? So what I'd like is that-- now of course, if I rotate, it's likely the representation will change. But I would like it to change in some understandable, systematic, simple way. If I consider, for example, perspective projection images, they don't change in a simple, understandable way.

As we know, if we take a 3D object and we rotate it in front of a camera, we get images that are not simple changes from a previous image. The perspective projection induces a complex, messy transformation. So if I then want to recognize the object, or I want to align it, that's not a good representation. Because the transformation is very complex.

So I can't have rotational invariance. Or rather actually, I don't want it. Because I'm going to try and recover the rotation. But what I'd like is that when you rotate something, the representation changes in a very understandable, simple way that I can exploit to both handle the recognition and handle the alignment problem.

OK, so now there are lots of attempts at doing this, finding a representation that satisfies those criteria. Let's just look at one. So this is generalized cylinders. So a cylinder, when someone says cylinder, you tend to think of a right circular cylinder. That is one that has a circular cross-section. And it's obtained by sweeping a generator along a straight line.

So in this case, we can think of this object as created by taking a circle and moving it along a line. And perhaps, even more constricted, that circle is perpendicular to the line. And so we generate that cylinder. So that's the most strict definition of a cylinder.

And we can generalize it a little bit by changing the shape of our generator. And now we can generate more complicated shapes. But they still have the property that the cross-section anywhere is the same, if I cut it perpendicular to the axis anywhere along the length. and I guess the mathematical definition of a cylinder allows for that version.

Now I can introduce a couple of other things. One of them is I can tilt the generator relative to the axis. Well, that doesn't do a whole lot. That's just a for shortening transformation. But suppose I allow the size of the generator to vary as I go along. Well, then I can generate cones, for example.

So again, we're sweeping along a line. And now we're allowing the size to change. Then we can allow the line along which we sweep to be curved. So far, we've had a straight line. So I might have a curve like this. And then let's take a circular cross-section, but of varying radius. So I can generate a shape like that.

And we can combine these. And we could even allow the generator to change as it goes along the shape. But now it's getting out of hand. Then you can do anything. And it's not unique anymore.

So this was an idea that was pursued for a while as a representation for objects, so that we can determine the alignment and recognize them. And it was somewhat of interest when people were trying to represent human bodies. So you can imagine that you can represent arms, parts of limbs, as generalized cylinders and then kinematically link those together, and build a 3D model that was kind of like an artist's wooden puppet that would have parts that were each individually generalized cylinders.

There are some problems with this. So one is that in order to do recognition, you would like the representation to be unique. It's going to be harder if there's an infinite number of different ways of describing the same object. And here's an example. Here's a sphere.

And I could represent it as a generalized cylinder by having that axis and then circles that grow in size and shrink in size, a perfectly good representation of a sphere. Unfortunately, there's an infinite number of those. Because it could be this axis and those circles.

So the sphere is kind of a tough case in particular, because of the symmetries. But the same problem shows up elsewhere and in particular, when we allow for inaccuracies in the data. Sometimes it's hard to tell the difference between objects that do have a unique generalized cylinder representation and objects that don't.

So this was used a little bit. It's not been overly successful because of the reasons we described. And there was a tension between allowing more freedom in the generation of these generalized cylinders versus assuring that there was some semblance of singularity, uniqueness. So that you could solve the problems that the whole thing is designed for.

OK, so instead, we're going to look at this representation. And again, keep in mind that this is an active area of research, unlike some of the 2D problems, which people kind of agree on solutions. Here, each proposed solution has some limitations.

And so we'll look at the limitations of this representation as well. OK, so let's go back for a moment to polyhedra We said we didn't want to do polyhedra as a good starting point. So as I mentioned, one way to describe the polyhedron is to give the vertices and then the graph coordinates in 3D. I mean, the other ways.

But one way you could have a list of vertices with 3D coordinates and then a graph structure that tells you which vertex is connected to which vertex, which face has what edges, and so on, that graph structure. But another way is to look at the faces and draw unit vectors perpendicular to them, and then multiply those by the areas. And then throw away that whole graph structure and just remember those quantities.

So we'd have a vector n1, which is that, and then a vector n2, which is this. And that's, interestingly enough, under certain circumstances is a unique representation of that object in the sense that there will be only one object that has that representation. And that's something we want. Because when we do recognition, we would like uniqueness. We would like it not to match some other object.

And it's kind of surprising. Because we've thrown away a lot of information. We've thrown away the relationship between the faces. We've thrown away actual coordinates, corners and stuff. And yet, here's a representation. And Minkowski has a non-constructive proof long ago, that this is unique for a convex polyhedra.

You know, it's interesting that oftentimes a theorem will be ascribed to a person that varies with geographies. They go to another country. And oh, this is Green's theorem or actually, no, it's Stokes, or it's-- well, in this case, Minkowski got to have his name on this theorem. Because there wasn't competing theorem invented by someone in the English speaking world.

So he actually got to be the guy with the name on the theorem. Now interestingly enough, the proof is not constructive. Meaning, if you give me these three quantities, I can tell you there's only one convex polyhedron that corresponds to those. But I don't have an algorithm to construct that thing.

And so there was some effort for a while in the machine vision world to come up with an algorithm. And I guess, Katsu Ikeuchi came up with an iterative algorithm that would solve that problem in a very slow fashion. But I guess people pretty quickly realized who cares? Because what's our job? Our job is recognition and alignment. It's not reconstruction.

So if we describe the object using these quantities, we want to compare those against the model library, and match them up, and figure out, how do we have to rotate this object we're seeing so that it lines up with the model in our library? We're not in the business of saying, OK, we need to reconstruct this in 3D.

I mean, we may have already done that. But the fact that it's a non-constructive proof isn't the deterrent. It's not important, not relevant. OK, so how do I use this in a more interesting case? So this is just a polyhedra. Oh, and by the way, it's not too hard to prove that-- when you stack these vectors tail to tail, they form a closed loop. That is the sum of these vectors, n1, n2, n3, is zero. And we'll prove something similar in a moment.

So that's constraint on what would constitute a valid representation. If you get a bunch of vectors that don't add up to zero, then you know it's not a closed convex object. And that can happen if, for example, you've left out one of the facets. Then yeah, they won't add up to zero.

But other than that, any combination of vectors does represent some convex polyhedron, as long as it satisfies that constraint. OK, so let's take a more complex object, like-- I don't know-- a ICBM re-entry vehicle. Somewhat simplified. There's a cylindrical part. And maybe there's a conical part. And I guess there's a flat part that we don't see here that's on the back.

So what I can do is approximate this using our mesh polyhedral representation. For example, I can cut it up into slices, such that the normal for all the points on these slices, well, they're not exactly the same. But they have very little variation. And with a conical section, I can do that. OK, so the idea is, I mean, I can make a much finer mesh. But I'm actually combining things that have similar surface normals.

And then what do I do? Well, then I compute all of these quantities, the area ij times the unit vector nij. And I keep those. Now, of course, I've got to be careful. Because I just said that the mesh representation isn't good. Because I might draw a different mesh. So now, I go to unit sphere. And I plot these vectors. So each of them has a direction in space that then corresponds to a point on the sphere. And I put down a mass at that point corresponding to the area of that patch.

For example, if I divided the patches up more finely, I would have had two areas half the size. But they both contribute to the same point on the sphere. That's why I could go with them this way. So I could cut it up into tiny little pieces. It doesn't matter. What's important is how much mass ends up on the sphere here.

OK, now let me do this for the whole cylindrical surface. Well, the surface normals for this cylindrical surface keep on turning. But they're all in a plane that is perpendicular to the axis of the cylinder. So then imagine cutting that sphere with that plane. I'm going to get a great circle.

So I'm going to put down masses all along this great circle. And that's the part of the representation that corresponds to the cylindrical surface. So for example, over here, there's a unit vector that points out that way. That's going to be on the sphere somewhere here. And I put down a mass. And in this case, the way I've cut it up, all of these masses are the same.

OK, what about the conical part? Well, same thing. I construct a unit vector. I take into account the area. Let's call this bi and-- I don't know-- mi, just for variety. And that ends up on the unit here somewhere. And I put down a mass there. And now, if I consider all of the facets on the cone, the surface normal will change. And they're not in a plane.

But if you think about the surface normals, they form a cone themselves with the complementary angle of the cone. So then I cut the sphere with that cone. And I'm going to get a small circle. So if you think about all of the facets of this cone, they will all contribute to points on the sphere on that small circle.

OK, and then, well, there's a piece missing. If I want to describe the whole surface of this object, which is the plate at the end. And the plate at the end is going to end up behind. But somewhere on the sphere, there's got to be a big mass that corresponds to that area. Why is it big? Well, because everything in that area points the same way.

So that whole large back plate area contributes to mass at a single point. So it's like an impulse. And there's my representation for a non-polyhedral object. And you can see now how we could use this in various ways for the task we've set ourselves, alignment and recognition. So we could have a library of objects. And for each of them we pre-compute this representation.

And then we can do a comparison. Now, the comparison is not least squares in the plane. It's on the sphere. So it's going to require a little bit of thought about how to implement that. But basically, we want to get them lined up, so that where this one has a lot of mass, the other one has a lot of mass.

And so you could imagine inventing some measure of how they correlate, how well they match up. And that can then be used in two ways. The one is orientation. So I'd have to take one of the two representations and rotate the sphere until things line up as best as they can.

And for recognition, I then do the subtraction to see how well they match up. And so this representation does provide for the two tasks that we described, lots of details to fill in. Then we said that we wanted certain properties. One of them was quote "invariance" or simple transformations resulting from translation and rotation.

Well first, translation doesn't get into it. Because we're only looking at surface normals. So if you take this object then you move it, we get exactly the same representation. So it's invariant to translation. Rotation, what about rotation? Well, rotation has a very simple effect on this representation.

If I rotate this object and just rotating the normal vectors, and that means that I'm rotating where they end up on the unit sphere. So it's just like rotating the unit sphere in an equivalent way. So the change in the representation resulting from rotation is very simple, very intuitive, easy to understand, easy to implement. And so it satisfies that constraint.

So in general, what we're going to be dealing with is kind of a density. So crudely speaking, if I have a certain mass here, that means there's an area on the object equal to that mass that has that orientation. So it's like the mass at any one point here tells me the area that has that orientation.

In the case of discrete facets, in the case where I'm taking a limit and taking a continuously curved surface, what I'm dealing with is density. So the density of points on the surface tells me something about the curvature. So if the object is highly curved, the neighboring surface normals will be pointing in very different directions. And that means that they'll be spread out on the sphere. And we get a low density.

So low density corresponds to high curvature. And conversely, high density corresponds to low curvature. And we can see this right here, where the thing with the lowest curvature is the plate, the end plate. And it gives a huge contribution to the sphere. Because it has very low curvature.

OK, and of course, we'll have to say what we mean by curvature. Because this is 3D. So it's a little different. Now, one way to get started on this is to look at a 2D version of this first. And so first of all, what's the idea of the extended Gaussian? What's the Gaussian image? Well, the important thing to keep in mind is the relationship between points on the object and points on the sphere. What do they have in common? It's the surface normal.

So if I want to find the part of the sphere that corresponds to this particular patch of the surface, I just find the point on the sphere that has the same surface normal. So go back to 3D for a moment. Suppose we have an Earth that is not a sphere. And we'd like to draw a map that's based on spheres. Well, then we have some way of mapping between them.

And there are lots of conceivable ways. Well, Gauss came up with one, which is basically to say, I'm going to map this point to the point on the sphere that has the same direction of the normal. And that's the one we actually use. Because if you are saying we're here at MIT at 42 and half degrees latitude, that is not this angle from the center of the Earth, which we sometimes mistakenly say.

What it is it's this angle. And so when we're dealing with a circle, all of these are the same. But when we have some other shape, we have to be clear about which directions we're talking about. And in this case, we find that the one that-- why do we use this? Well, because it's easy to determine.

You've got gravity pointing perpendicular to the surface. And then you have rotation about the celestial sphere. So you can determine the north celestial pole. Or you can look at where the sun is and what time of the year it is. And that that's the angle you're going to measure. Now, there are subtleties there, like the centrifugal force, the uneven distribution of masses in the Earth, and so on.

OK, so Gauss basically said one way of mapping between the convex object of arbitrary shape and the unit sphere is to just identify points that have the same orientation. And that's point to point. Well, we can generalize that to shapes. So suppose that we have Africa here, then we can map it onto the sphere. And why is that convenient? Well, because lots and lots of clever methods exist for mapping spheres onto planar surfaces for map making.

But the first step is you need to convert it from the ellipsoid to a sphere. OK, so how do we do it? Well, for every point here, we look at the surface normal and we find the point over here that has the same surface normal. So that's the basic idea of the mapping.

We make correspondences between points that have the same surface normal. And you can see that this mapping is actually invertible. So if I'm on the sphere and I want to know what point this part of Madagascar corresponds to over here, I just find the point that has the same surface orientation on this other convex shape, as long as it's convex.

Now there's a problem when we have non-convex shapes. Because there might be more than one point that has the same surface orientation. And so that's the limitation of this method, that it has very nice properties for convex objects, has some issues with non-convex objects.

But for the moment, let's focus on convex objects. So in the case of convex objects, this mapping is reversible. OK, back to 2D. So the idea is that, again, we now from some shape to a circle and maybe I should make this shape less circle-like, given that my circles aren't that great anyway.

OK, so here's a convex object. And now what we want to do is take a patch, well, in this case, a short line segment, on that surface and map it onto the sphere. And how do we do that? Well, we look at the surface normals. So there's a surface normal at the beginning. And that'll correspond to some surface normal here. And there's a surface normal at the end. And because it's convex, it's changing monotonically in between. So that whole range of surface normals in between maps into the whole range of surface normals here.

And we'll just parameterize that unit circle in the plane by the angle, eta. OK, and so what do we want to do? Well, we want to put down a density, which is inversely proportional to the curvature. So the mass, delta s, that's proportional to delta s, is going to end up being spread out over this part of the unit circle.

So if we have high curvature, it's going to turn rapidly. And whatever the mass is gets spread out at a large angle. And conversely, if we're in the flat part, the surface normals to turn very slowly. And all of that's going to end up in a very small segment over here. And so the density will be high.

And so we're going to end up with a continuous quantity of that angle. And that's the thing we're interested in. So first, let's pick some arbitrary point. So s is the arc length along the curve from here to there. And then again, these normals are parallel. So this must be angled eta. And so what we're interested in is curvature. Let's start with that.

It's the turning rate, right? So for example, if you're not turning, then k is zero. So it's the rate of change of direction. Or it's the inverse of one over radius of curve. OK, and then the density is going to be the inverse of that. So the density-- and that's it.

So that's our representation for a convex closed curve in 2D. We just map onto a unit circle the inverse of the curvature. And that representation is unique. There is no other closed convex object that will have that same distribution. Now in the 2D case, it's actually invertible.

Now, you can see how you could make a transition from a discrete case to continuous case. You can just divide this up into lots of little facets that are straight lines. And each of those facets will contribute point mass on the circle. And then as you produce the size of the facets, they become smaller and smaller and closer and closer together. And all that matters is the density. How much mass is there per unit area?

OK, we call this thing G for Gauss. OK, now I'll show the inversion, even though we'll find that, as Minkowski found, there's no inversion in 3D. But just to illustrate some of these ideas some more. OK, so we're at an angle eta. And delta s is going to be perpendicular to that.

And so when we make a small change, we get delta s is minus sine eta. Delta s and delta y equals cosine eta delta s, right? Because we're going to move back by an amount, delta x. If I blow this up-- OK, this is eta by something that's proportional to sine eta. And then move in 1. And so all I need to do is integrate that equation. Because it tells me as I move along how far I move.

Now, I may not know delta s. I'm probably integrating in eta. So let's see. We can say x is x0 plus-- and then change variables. [INAUDIBLE] And of course, this is--

And similarly, there'll be an equation for y. Put it under there. So y is [INAUDIBLE]. So in the 2D case, I can invert it. I can actually obtain the convex object that corresponds to that circular image. I mentioned that's not the case in 3D.

While we're there, I've been a little bit sloppy about limits. I haven't put them in. But we can do that. And one interesting question is, what is this? So those are the quantities that appear there. I started x0. And I drew this integration. And I construct this whole supposedly closed convex object. And so I should get back the same point, right?

So therefore, that integral better be 0 when I go all the way around the loop. So again, because I started at x0, I integrate over the whole curve, I should be back assuming it's a closed curve. So we're assuming that it's a closed convex curve. Then those integrals better be zero.

And so that means that the centroid of that mass distribution-- I keep on coming back to thinking of it as a mass distribution. There's a density on the circle, or the sphere, is at origin. Because these are really-- the integral of x, g, blah, blah, blah. And sorry, yeah, x. And this would be y. So the integral of x weighted by this thing is zero. And the integral of y weighted by that thing is zero.

And those are the moments, the first moments, you use in calculating the centroid. So this means that, OK, this mass distribution on the circle has to have a special property, which is that they may be more here or less somewhere else, and so on. All of that's fine. But it better have the property that the centroid is at the origin. So that's one limitation, which is exactly the same as what we had in the polyhedral case, that some of those vectors were zero. It's the same statement really.

So that's a limitation on what distributions are legitimate. But that's it. Other than that, you can have your masses arranged any way you want. OK, so let's look at an example. It's all very well in theory. Let's think of a circle of radius r. Always good to start with something really simple.

Well, in that case, the curvature, what's the curvature? So the curvature is k is d theta over ds. So how do I find out? Well, one way I can think about it is to relate the surface normal direction to the arc length along the circumference of the circle. So I can say that s is our theta, assuming that theta is measured in radians. OK, so that's very simple for a circle.

And then I need the a to the s. Well, I can get the a to the s is then 1 over r. Because eta is 1 over r times s. OK, so that means that the curvature is just the inverse of the radius of curvature for a circle. Now, in a more general case, we can still talk about the radius of curvature. Suppose the curve is not a circle. Suppose that elliptical shape, or something. We can still talk about radius of curvature.

Because we can fit a circle locally to that part of the curve and ask the question, what is the radius of the best fit circle at that position? OK, so circle is very easy. And it's not particularly interesting. Because it's the same all the way around. It has the same-- g is constant. So g of eta is 1 over k. And that's r.

And so it's constant all the way around. And by the way, this shows that we have this not very correct interpretation, which is that the value for any particular angle eta is how much of the object's surface has that as a surface normal? So this is saying that, first of all, in the case of a circle, that quantity is constant.

It doesn't matter which direction we're looking at. And then also, that goes up as the radius. Because as we make the circle larger, it gets flatter and flatter. So more and more of it have approximately the same orientation. So that's a useful way of thinking about that. OK, so we won't be able to use this for determining orientation. Because the orientation is ambiguous with that much symmetry.

So we need to come up with a better, more complicated example. And I'm doing this in 2D now. Because for 3D, I'm just going to write down the result. It's too boring to work out. So by torturing you with a 2D version, I am saving you the pain of looking at the 3D version.

So let's look at an ellipse. And we'll line it up nicely, so the equations come out easily. The center of the ellipse is at the origin. And the main axes are lined up with the x and y-axes. And of course, we know that one way we can-- and that's a so-called implicit form of the equation for an ellipse.

There's a wonderful book that's gone out of print many times and then got reprinted, which talks about different ways of representing curves. And you think, well, there's one and there's perhaps another. No, there's a dozen that are commonly used and dozen more that are less commonly used, so loads of representations. And we don't teach much about them these days.

But here's another, which is more useful for our purposes. OK, so what is this? Well, we can think of this as a squashed circle. So we basically multiplied-- imagine the circle of radius a and we've squashed the vertical dimension by this factor. And we get that. And that theta is the angle in the original circle. So there was some original circle. And we somehow squashed it to produce that.

So the theta is not an angle in that diagram. It's an angle in the diagram of the non-squashed version. OK, but you can see that if you take x over a squared and y over b squared you'll get 1. And that's kind of in many ways a more convenient representation. Because you could use it to generate the circle. This one here, how do you generate the circle? Well, I guess you could try all possible x and y's. And some of them will produce 1. And some won't. And you can put down a point there.

But if you want to draw it, this is much more convenient. Because you just step through theta and compute a polyhedral approximation of however fine a detail you want. OK, so that parametric representation is great. And that relates to the Earth as well. The Earth can be thought of as a sphere that's squashed in the vertical direction. And just remember that those angles aren't the same, just as we talked about that's not latitude. And it's not geocentric latitude either.

Oh, and by the way, if we needed the area, it's pi times ab. And you can check that works for the limiting case, where we are dealing with a circle. So what do we need to know? Well, we need to map it to a circle based on the surface normals, or in this case, the normal to the curve. So we need to compute the normal to the curve.

Well, we can start by computing a tangent. So how do we do that? Well, we just differentiate with the parameter. And that gives us a vector that goes along the curve. So we're going to look for something like that by differentiating this with respect to theta. And so we get minus a sine theta b cosine theta. Then you first define this vector r, which is this thing.

So these are now two vectors. Because we're in 2D. OK, so that's the tangent. And the normal, of course, is just perpendicular to that. And so how do I do that? Well, I flip x and y and change the sine. And that's not a unit vector. But it's a vector in the direction of perpendicular to the curve. And that tells me where I am on the sphere.

Now, on the sphere, on the circle, I have that. And so these directions have to match. So the direction, that's not a unit vector. But if I normalize it, then that should match that. If I match those up, I get-- let's see. a sine theta. where n is the length of that vector. So let's say n squared is b squared cos squared theta plus a squared sine squared theta.

OK, well, let me just define another vector. In analogy with the vector we have over there. And then we find that-- And the details of this aren't terribly important, just algebra. The important thing is that this is what we end up with. And the quantity we're interested in is just the inverse of that, G is 1 over k. So this is the curvature. And the quantity we want is the inverse of the curvature.

So one thing that's interesting is to ask, what are the extreme of this? And so you would imagine that the extreme are going to be at the ends of the semi axes. And so we would expect that the extreme will occur for theta equals 0 and eta equals pi over 2. Well, 0 and pi is the same thing. And pi over 2 and 3 pi over 2. And in that case, we end up with ab over a cubed, which is b over a squared and ab over b cubed, which is a over b squared.

So I'll draw that ellipse again. Let's see. a is the large one. OK, so details aren't that important. But what we've done is we've computed the extended circular image for an ellipse. And it's a continuous function of eta, the angle on the unit circle. And it varies, unlike the circle. And it has a maximum and a minimum and a maximum and a minimum, as you go around. And they depend on the semi-axes.

And as you can imagine, the a is the larger of those two semi-axes. So here we see the curvature is quite high. And b is the shorter axes. And the curvature is small. And so there's a continuous distribution on the circle, which we can now use to determine the orientation of an ellipse that's not lined up with a coordinate system. Because we'll have that same distribution of this angle. But it'll be rotated.

And so in order to get the match, we have to take one of the two and rotate it until it's a good match to the other one. And similarly, once we've done that, we can check how good a fit it is. And if it's a good fit, then we do, in fact, have an ellipse. If not, well then, the object's not an ellipse.

And so if we have a library of objects, what we would do is do this calculation for each object in the library and find the one that is the best match. I know all these different angles, just like the Earth image over there. So theta is a parameter. It doesn't actually show up in this diagram. Where it comes from is the theta in the circle, before the circle got squashed.

And so this used to be theta. But it's now gotten decreased. Whereas, eta is the position on the sphere. So we map from this space onto the unit sphere of directions. So there's a relationship between the two, which let's see, it's something like b10-- I've got it somewhere.

So tan theta is related to tan eta by something like this. And this is a, if I got it right, this is a important formula used in geodetics. Because it relates the geocentric angle, the angle we make at the center of the Earth to the angle of the latitude that we use for computing latitude. And in the case of the Earth, the difference is not very large. The flattening is only 1 over 292. So those angles are pretty close.

But it's important to keep them separate. Yeah, I got it right. OK, so that's a 2D version. And actually, there are applications of this in 2D. And you can do more interesting things in 2D also. For example, you can do some filtering operations. So you can do convolution on the circle, which is different from convolution along the line. Because things wrap around. And that's a whole another topic.

There's a paper on that on Stellar as well, in case you were interested. Let's go back to 3D. That's the problem we're really interested in. And so we start with Gauss mapping, which basically connects points on a surface to points on a unit sphere based on surface normal orientation. And that's four points. And then we extend that to shapes.

So we might have some object here. And there's a shape there. And then there's a corresponding shape over here. They're related in that every point here has a surface normal. And that gives me a point in that patch. So let me call this the object. And this is the area delta o for object. And this is the sphere. And this is an area delta s. And the curvature is just defined as, or the limit of--

So again, that intuition that if we have a very flat area, then almost all of that area is going to end up really close to the same place on the sphere. So that ratio is going to be very small, meaning the curvature is very low. If on the other hand, I'm looking at something like this, where it's very highly curved, well, those surface normals are going to be really spread all over the [INAUDIBLE]. They're going to correspond to a large area on the sphere.

And therefore, this ratio is going to be large, high curvature. So this is Gaussian curvature. Now curvature in 3D is more complicated than in 2D. So this isn't the whole story about curvature. This is just a convenient single scalar quantity that measures curvature.

By the way, if I doll around the circumference of this area in a certain direction, I'll go around the circumference here in the same direction, as long as this is convex. Now, what about non-convex surfaces? Well, if you think about what's a non-convex surface? Well, like a saddle point, or one of our hyperboloids of one sheet. So saddle point, think of a Pringle chip.

So here's a surface with negative curvature. And if we trace around the surface normals around the outside, and plot those on the sphere, they will actually travel in the direction opposite. So for non-convex object-- and in that case, we consider the curvature negative.

So that formula up there should take into account the sine of the area, so to speak, which is the direction. So if the two directions match, then it's positive. And if this one's going around in the other direction, it's negative. But that's not going to happen for convex objects. And we're mostly going to be talking about convex objects.

So take a very simple example. We take a sphere of radius r. And k is 1 over r squared. And therefore, g is r squared. So that's an analogy with our 2D case, where k was one of r and g was r. So what does that mean? And where does that come from?

Well, it's pretty simple. It's the ratio of these two areas. And in the case of a sphere, I can actually just take the whole thing. So this is a unit sphere. So its area is going to be 4 pi. This is a sphere of radius r. So it's areas 4 pi r squared. And so if I take their ratio, surface delta s over delta o, I get one over r squared.

And so again, that's consistent, that if you have a small sphere, it has high curvature. And conversely, for a large sphere, you want to have high curvature. OK, so this is kind of the key for us. And conversely, g is delta o over delta h. And by that I mean in the limit as we make those quantities smaller and smaller.

OK, so what we're doing is intimately tied up with Gaussian curvature. Because it's just the inverse of the Gaussian curvature. Oh, I guess we're out of time. But one of the interesting things we can do now is talk about integral curvature, which applies to surfaces that are not smooth. So suppose that we're looking at a brick. It has a rectangular corner. We can't really talk about its curvature. Because it's zero on the faces and infinite on the edges.

But we can actually talk about an integral of curvature over part of the brick. So we'll do that next time. And we'll talk about how to use this in recognition and alignment. And there will be a quiz out on Thursday.