[SQUEAKING]

[RUSTLING]

[CLICKING]

BERTHOLD HORN: Briefly last time about tesselating spheres in various dimensions. We found that representing rotation as unit quaternions was useful. And so in exploring that space of rotation, we're dealing with a sphere in 4D, and it'd be good to divide it up into equal areas.

And in the process, we started talking about 3D, which is slightly easier, but which is also going to be important for us. And tesselations of the surface of the sphere can be based on the platonic solids, which there are five of. And they have equal area projections on the sphere. So they're very nice from that point of view, but the division's kind of coarse.

And so typically, we'll divide it more finely. So one way is to go to the Archimedean solids. And so here, we've got all of those.

I guess in this view, there are 13 of them. As I mentioned last time, there are potentially 14 because this one is not equal to its mirror image. So you could have two of those with different orientations.

And they have, again, regular polygons as facets, but this time, you're allowed to use more than one flavor of polygon. So here, we even use triangles, squares, and pentagons, and as a result, the areas aren't equal. So this is a pretty extreme example where we have triangles and dodecagons. So those triangles obviously have a much smaller area than the dodecagons, and so on.

Anyway, we can base a tessellation on that, dividing up the sphere. Also, this is relevant to-- when we're talking about rotations, we're interested in the rotation groups of these objects. And let's see, is there another one somewhere hidden behind here? No? Well.

And so there are 12 elements of the rotation group for the tetrahedron. There are 24 for the hexahedron. And there are 60 for the dodecahedron.

So what about the octahedron and the icosahedron? Well the octahedron is the dual of the cube, so it has the same rotation group. The icosahedron is the dual of the dodecahedron, so it has the same rotation group.

And we'll define more precisely what "dual" means, but roughly speaking, replace a face with a vertex, replace a vertex with a face, replace an edge with an edge at right angles. So you can see how you can construct one of these out of the other. So that's not much-- we got groups of size 12, 24, and 60.

And we're talking about the surface of a sphere in 4D, so that's a three-dimensional thing. And we're only dividing it up into 60 regularly spaced points. So we'll need more. Oh, by the way, one of these Archimedean solids is one that's commonly used and kicked around.

Now, if we divide them up, we'll often start off with one of these solids, either platonic or Archimedean. And here, for example, we've taken an icosahedron and divided it up into lots of triangular areas. And that's a popular thing.

So here is a geodesic dome. And you'll notice that many of the vertices have six edges coming in, but there are a few that have five. And the precise rules affecting that-- there will be 12 of those with five, and everything else will be hexagonal.

And it looks very regular, but actually, the facets aren't all the same size. Here's another one. So this one, actually in a way, is better for our purposes.

When we divide the surface up, it's kind of making histograms on the sphere. And if we make a histogram in the plane, we'll often just use square tessellation, which isn't the best, but pretty straightforward.

Over here, it's not so obvious what to use. But one of the things that's convenient about this type of tessellation is that it's not very pointed. So you don't have parts of a cell that are far away from the center compared to other parts.

So actually, the tessellation, the triangular ones aren't that great, the ones I've shown you. This is a dual of a triangular tessellation. We replace the center of each triangle by a vertex, and so on.

And so in that respect, this one is a better tessellation. And of course, it gets bigger. So this was, I guess, the 1968 Expo in Montreal.

They built this geodesic dome. And what was interesting about it was that it was union built. And the architect had made sure that all of the links were labeled because they weren't all the same.

But the difference was small. So the people working on it said, that's stupid. Let's just work with it. And it started not becoming a sphere because they hadn't followed the labeling. And so they had to actually take it apart again and start over again.

I think this still exists. I've been there a few years ago and it was there. And that, of course, is Buckminster Fuller. Here is, actually, a page of out of Buckminster Fuller's original patent application of 1965, where he describes how to build these geodesic domes.

And again, the geodesic dome for us has a number of features. One of them is that the facets aren't all exactly the same size, which is undesirable, and then they're triangular, which is undesirable. So we're likely to instead use the dual, which would have a large number of hexagonal or approximately hexagonal facets, and some number-- well, 12-- pentagonal facets. Enough of that.

Then right now, we'll be talking about-- since I have the projector here, I thought I'd do this. We're going to run across certain types of surfaces that make some machine vision problems hard, called "critical surfaces." And they are hyperboloids of one sheet. So I need to talk a little bit about-- it's not very well-- I'll blow it up.

So quadrics are surfaces defined by second-order equations. So we've got x squared over a squared plus y squared over b squared plus z squared over c squared is 1. That's an ellipsoid, shown over here.

Special cases where a, b, and c are the same-- you get a sphere. And we're all familiar with that American football shape. We're going to be interested in this surface, which is a hyperboloid of one sheet.

And it's x squared over a squared plus y squared over b squared minus z squared over c squared equals 1. And this is a shape you may be familiar with from certain kinds of furniture made out of rods. And it has the feature that it's ruled-- that is, we can embed straight lines in the surface, which obviously, you can't do here.

And actually there are two sets of rules that are at an angle to each other. And so you can make this beautiful, smooth, curved surface out of straight sticks, which seems pretty strange. But there are parts of the world where this is used widely to make chairs, and tables, and so on.

And then we have hyperboloid- of two sheets where there are two minus signs. So basically, you can classify them based on the signs. These are all positive. Here, there's one negative, and here, there are two negatives.

And obviously, it's called "two sheets" because there are two surfaces. You can't get from one to the other. And let's see-- signs, so what's the other possibility? We could have no negative signs, one negative sign, two negative signs.

How about three negative signs? What type of a surface do we get then? Well, we get an imaginary ellipsoid because it's all negative. How can it be equal to 1, unless you allow complex numbers-- hence the term "imaginary."

So these are the sort of generic ones. Now annoyingly, there are a lot of special cases. And we already talked about the sphere. That is a special case, and that's pretty obvious.

But then there's a whole slew of other special cases. So let's see-- so that's just the ellipsoid. We already saw that. Then there's a special case of that, where we're dealing with a cone.

And then, we have where that neck of the hyperboloid of one sheet becomes infinitesimally small. And then, we have elliptic paraboloid. And this one we've already seen, the hyperboloid of one sheet.

Then we have hyperbolic paraboloids. So these, you see, don't have a quadratic term in z. They just have a linear term in z.

And then, that's the other one. And there are more special cases. And most surprisingly, one special case is planar, which seems weird because you've got a second-order equation. How can that be? Hang on a second.

Well, imagine if you have two planes that intersect. How can you describe that surface? Well, one way, you could take the equations for one of them-- some linear thing equals 0-- and take the equation for the other one-- some linear thing equals 0-- multiply them, and that would describe that object. And obviously, when you multiply the two linear equations, you get quadratics.

So enough pretty pictures. Let's talk about relative orientation and binocular stereo. So the problem we're interested in is computing 3D from 2D using two cameras.

And we already said that if we know the geometry of these two cameras, it's relatively easy because if you have a point in the image plane where there's some feature, you can connect it to the center of projection. Gives you a line. You extend that line out into the environment, and the object is somewhere along that line.

And now you do that for the left camera, and you do that for the right camera. You have two rays. And ideally, they'll intersect, and that gives you the 3D position of the thing you're looking at.

So that's the easy part. In order to do that, you need to know the geometric relationships between the cameras. And that's what relative orientation is about.

And so that's something that you would do ahead of time before you install this thing in your car and use it for autonomous vehicle purposes. But you might have to do it again on the fly because if the baseline isn't something incredibly rigid, it's quite possible that things get misadjusted when you use that vehicle. So we may need to do this calibration again.

It's not just binocular stereo, though. The whole same machinery applies to structure from motion, where instead of a left and the right image, we have an image before and an image after. And again, there's a duality-- did the camera move or the world move? Doesn't matter-- the same math.

So let's look at the binocular stereo case, keeping in mind that it's just the same as the finite motion case. So infinitesimal motion is easier, but we're talking about a substantial amount of motion. So suppose that we have a left center of projection and the right center of projection, so those are the principal points of our cameras.

And then there's a baseline. And we'll have a vector B that describes that baseline. And now, we're looking out at a point in the world, and we can determine where it is in the two cameras.

From the individual camera images, we don't know how far along that ray is. But of course, if we have both, then we can find the intersection.

And here, I'm working in the right-hand coordinate system. I have to pick a coordinate system. And I have several choices-- I could pick left, I could pick right, or for symmetry, I might pick the center one.

But I do want to make sure, when I get the results, that they're not biased by my choice of coordinate system. So if instead I had picked the left coordinate system, I should get the same answer as if I picked the right coordinate system. So we'll check on that.

So in this case, then, the baseline is measured in the right coordinate system. Then our r, is, of course, measured in the right coordinate system. And this prime indicates that we've converted it from the left coordinate system to the right coordinate system.

Let's talk about the geometry of this thing. So if I have a point out in the world, and I connect it to the baseline, that defines a plane. And I can think about the image of that plane in both camera systems.

So suppose we have an image plane here and an image plane here. Then we expect to see something like this-- that this plane, which is defined by these three points, L, R, and P, projects into a line. And the point P is imaged somewhere along that line.

Or think of it another way-- suppose that I go into the left image, and I use my interest operator, like SIFT or SURF or something, and I define this point. What do I know? Well, I know that the world point has to be along that straight line.

And then I wonder, what does that mean for the right camera? Well, I project that straight line into the right camera. And of course, I just get a straight line in the right camera.

And so one conclusion is that when we search for correspondences, we only have to search along a line. So instead of trying to look for that interesting point all over the right-hand image, once we've figured out the geometry, we only have to search along that line. And that gives us some measure of disparity, which we can turn into a distance measurement.

So those lines are called epipolar lines, where there's a correspondence between two sets of lines. One way to think about this is that there's some line here between L and R. And now, imagine there's a plane through it. And now we draw other planes. So for example, there would be another plane here.

So think of all of the planes that have the property that they pass through that line. Or we could take one plane and just rotate it. Those are all the epipolar planes.

And when we look in the images, we're intersecting the image plane with this arrangement. And so we're going to get a set of lines, each of them intersected with the image plane gives us a line. But the lines won't be parallel.

So we're going to end up with-- if I look at the actual images, it'll be something like this. This is my left image. So if they're not parallel, they're going to intersect. And actually, they all intersect in the same place, if I've drawn it properly.

And so this is my left image, and that's my right image. So what is this point in the left image, if you think about this geometric arrangement of the sheaf of planes in 3D? So that must be where all these planes come together.

So that's the image of the right camera. So this is the image of R. And this is, correspondingly, the image of the left camera.

Now, they don't necessarily have to actually appear in the part of the image that you're scanning. They could be outside the frame. And typically they will be, particularly if we make the camera's face more or less parallel out at the world. Then we would have this point move out that way, and this point move out that way, and these lines would be more like parallel.

So here, it's like your eyes are converging. If you're looking at a nearby object, the two eyes are not parallel, but they're converging. So the way I've drawn it, I'm assuming they are converged slightly.

They could also be parallel, in which case these points would move out. Or they could, in fact, be diverging, except that's not the greatest thing. Because then the overlap-- the part of this image that's actually appearing in that image-- is reduced.

So part of the point of actually trying to converge is that you try and have as much overlap between the two images as possible. But it's not necessary. It just means that you don't have a lot of extraneous image information that's of no use to you for recovering depth.

Now, actually, I said that's the image of the right-hand side. It's actually the whole baseline. The whole baseline images in one point in the left image, and in this point in the right image. And the other projection center happens to be on the baseline, so it ends up there, as well.

And so one of the properties that we mentioned was that if we know the geometry, then there's a correspondence between these lines. And if you find anything on the line in the left image marked in red, then you only need to search along the corresponding line in the right image. So we're trying to find the relationship between these two cameras.

And that involves, obviously, the baseline. That's a translation. In the equivalent motion vision case, there's a translation from that position to that position.

And the other thing is the relative rotation of one camera relative to the other. And so we know that rotation has 3 degrees of freedom. So we might think that as in absolute orientation, it's a 6 degree of freedom problem.

But it isn't quite, and the reason is the scale factor ambiguity we talked about. So if I take the whole world, and I just expand it by some arbitrary factor, the image positions won't be changed, because in perspective projection, we're dividing x, y, z, and if you expand both x and z, the result doesn't change.

So from the binocular situation here, we can't get absolute size without some additional factor. And so that means that we can't get the absolute length of the baseline unless we have some additional thing. Now of course, in images, there might be other things like you've imaged a Walmart and you have an idea about how many acres of land Walmart occupies. Then you can scale everything according to that.

But without that, we have to treat the baseline as a unit vector. And so we only have 2 degrees of freedom for that component. So there's a total of 5 degrees of freedom. So we have five unknowns, unlike six for absolute orientation.

Then comes the question of how many measurements do I need? What is the minimum number of correspondences to pin this down? And it's hard to come up with a good heuristic argument.

What's the mechanical analog? Well, the mechanical analog is that we take the image points, and we create a wire that goes in that direction, a wire that goes in that direction. And they have to intersect, but we don't know how far along.

So imagine passing these two wires through a collar that forces them to intersect somewhere, but the collar is free to move all along. So if that's all I have, two rays, one correspondence, one ray from the left and one ray from the right, then it's obvious that I can play all kinds of games.

I can change the baseline. I can move this around, rotate it. I mean it's constrained, but it's only one degree of freedom constraint.

So that obviously won't do. So let's add a second correspondence. So we have a second image point, and we have a collar around there. And that's going to be more constrained, but you can already see a number of ways that this can't be enough.

For example, you can draw this axis. And you can rotate the right camera and all of its wires coming out about this axis, and they will still pass through. So that reduces the degree of freedom, but it's not enough.

So the answer is 5. And one hand-waving argument is that each correspondence gives you one constraint. And why is that?

Well, if you compare the two images, you will often find that there is a disparity, and the disparity has two components. So this is a situation where we're superimposing the two images, left and right. And we're comparing the image position of some particular point.

And let's suppose that we approximately line things up-- that the optical axes are parallel, more or less, and they're more or less perpendicular to the baseline, so that if things are infinitely far away, they would emerge in more or less the same place in the image plane. But in practice, they won't. And so there'll be an error, disparity, in the horizontal and the vertical direction.

Now the horizontal "disparity," as it's called-- should put that in quotation marks, but anyway-- corresponds to depth. The closer the object is, the more those two rays have to cross over. The angle between them gets larger, and so the image position changes.

If I don't converge my eyes-- I just keep my eyes parallel-- and then I bring an object closer and closer, then the difference between where it images in the two eyes gets further and further apart. And we started there. We had a formula where the depth was inversely proportional to that disparity.

And so that's the horizontal disparity. So that's the thing we're trying to use to get depth. But there could also be a vertical disparity, and what's that from?

Well, it shouldn't be there if we had things properly lined up. It's an indication that the two cameras are not oriented the same way. And so the vertical disparity provides the constraint.

And if we figure out the rotation and the baseline, we should be able to zero out the vertical disparity everywhere so that we don't ever see a vertical disparity. In actual use, once we've got everything set up, we should only have horizontal disparities, which are then inversely proportional to depth. And in order to set the instrument up, one thing we can do is tune out those vertical disparities.

And that's how it actually worked for decades. People had very carefully made optical equipment. They'd plonk in two glass plates that were photographs taken from the plane, and there was a binocular-like imaging arrangement.

And you set it up by noticing that this church tower is slightly higher in this image than that. And there was a sequence of moves where you would tweak out these five parameters in sequence, and iteratively. And one of the nasty features of it was that it might not converge.

And so there was an additional component where you would record the five adjustments you made and plonk them into a formula which told you how to adjust things so that they do converge. So it was pretty painful. And it's too bad that there wasn't some easy solution.

But we're going to find a solution-- unfortunately, not closed form. But in any case, nobody does it that way anymore. But it was interesting to think that there's this complicated mechanical Heath Robinson-like machine that had these knobs that you could tweak to find the parameters of this transformation in three-dimensional space.

So that's the minimum number of correspondences we need. In practice, of course, we want accuracy. And so of course, we would like to have more points.

And then there won't be any arrangement that makes them all work, so we'll do some sort of least squares thing, in that we will have the image positions match as closely as possible. So it's important that the error is in our measurement of image position. And so the thing we want to minimize is going to be the sum of squares of errors in image position, not something out in space. And we'll come back to that point.

So in practice, we use more than five points. Five is the minimum. And there's an old problem, which is it's nonlinear. So how many solutions?

And roughly speaking, we're dealing with second-order equations. And we end up with seven second-order equations, these five plus two more having to do with the baseline being a unit vector and so on. And so by Bezout's theorem, we might have as many as 2 to the 7 solutions, 128, which is kind of scary. And that's also one reason why you typically don't use five correspondences. You use more to try and get rid of that ambiguity.

Anyway, it was for a long time not known what the actual answer was. And also, there are different ways of counting. But the true answer is 20.

Now, Kupka, a century ago, showed that there can't be more than, in his way of numbering, 11, which would be 22 by our counting. And it took almost a century to get it down from 22 to 20. But in a way, for us this is kind of a curio. It just shows you that it's nonlinear.

And it shows you that you have to be careful when you get the solutions. But in practice, when you take many more measurements than needed, and when you have a rough idea of the arrangement-- I mean, you built this thing, typically, so roughly what its geometry is-- then it's not really a problem. Except for people interested in obscure theoretical-- not that there's anything wrong with it. It's a lot of fun trying to figure this out.

So how do we find the baseline and rotation from correspondences? So we have our baseline again, L and R. And then we have some point out there. And this is r l1, and this is r l1. This is the baseline.

Now, one thing we can do is notice right away that those three vectors are coplanar. That's one of these epipolar planes. So what does that mean about the parallelepiped that I can construct from those three vectors? So maybe I should put arrows on these things.

So we said that the volume of the distorted brick shape that we get by using those three vectors as edges-- so here is the construction I'm talking about. So you take these three vectors, and you build things with parallelogram inside, and the whole thing is called a "parallelepiped." And its volume is the triple product. And so what do we expect for that triple product based on that argument?

It's a flat thing, so it should have no volume. Because these three vectors are coplanar. So this object isn't something that has three-dimensional volume.

It's flat. And so in the ideal case, we expect that to be 0. And that's called the "coplanarity condition," and it's the basis of all of this stuff.

So right away, you can imagine a potential least squares method. All we need to do is for every correspondence we have, we compute this triple product. And it might be positive or negative, depending on the directions of these arrows-- do they form a right-hand coordinate system or not?

So we can't just add them up. But we can add up their squares. And supposedly, if there are no errors, the sum of those squares should be 0.

So a potential method here would be given, actually, our measurement errors, let's minimize the sum of those squares. So we could take this triple product, take the sum of squares. And if our estimate of the baseline and our estimate of the rotation, which is buried in here, is correct, then it should be 0.

Now, that's a feasible approach. It's not particularly good because-- well, let's put it this way-- if you have absolutely perfect data, you will get the perfectly correct answer. But it's not a good method.

And the reason is that it has a very high noise gain. And this typical of quite a number of bad methods that have found their way into machine vision, where mathematically, they're right because you give them the correct measurements, they give you the correct answer. But they are not practically useful because with a small error in the measurement, you don't get the correct answer, and you often get an answer that's wrong in quite a large degree.

And so we need to do a little bit better than that. But we can start with that as a basis. So what are we really trying to minimize?

Well, as I mentioned, the key is the image. This is where we make the measurements. And when we make the measurement, then we know that that point is known, but only within a certain accuracy.

And for our edges, we were saying that if you do really well, we can determine them to 1/40 of a pixel. But whatever it is, whatever that quantity is, it's in the image that we are trying to match things up, not some triple product volume of something. It's true that if everything is correct, that volume will be 0, but it's not true that that itself is a good measurement of error. It is proportional to the error. So if we figure out the proportionality factor, we could use it.

Well, let's go a little bit further, and say, so the measurements aren't perfect, or our baseline and rotation aren't perfect. And then those two things won't intersect. So let's see how that works out.

So we have r l prime here. So here, the two rays are not intersecting anymore. And this minimum separation, that's a measure of error, so we could think about minimizing that.

So let's take that idea little further. So it's pretty easy to prove-- I won't bother-- that this minimum approach has the property that it's right angles to both of these. Because if it isn't, then you can move it closer and get a smaller distance.

And so that means that this direction is perpendicular to both of these two vectors. So it's parallel to the cross product. And so I can draw an equation that goes around this loop.

So I start over here, and then I go around this, along this vector. Now, I don't know how far I have to go because I've only got the direction, remember. And I'm going to then add to that a vector that goes in this error direction.

And that's going to be equal to going the other way, which is b plus, and then coming out along the right way. So all I'm saying is if I add up the vectors on the left here, then they must equal the vectors on the right. Or I could have just gone around the loop, and said the result is 0.

So that's an equation I'm interested in, partly because I would like to make this small. I would like those rays to intersect or almost intersect. So there I have got a vector equation.

What do I do with vector equations? Well, the appendix teaches you some tricks. One of them is convert it to things we know a lot about, scalar equations.

And how do you do that? Well, you take dot products. So we can turn this into a scalar equation.

And actually, it's a vector equation. So it provides three different constraints. So we should be able to do that three times, and so take a dot product with three different things, and get three scalar equations, which correspond to that single vector equation.

Now, when we do that, we'd like to pick something that makes a lot of these terms disappear, to simplify the equation. So we want to pick something that makes a lot of these terms drop out. And so for example, if we take a dot product with-- why is this good?

Well, r l cross r r is perpendicular to r l and perpendicular to r r. So when you dot it with this term, that term goes away. When you dot it with that term, it goes away, because it's perpendicular to r r.

So that's a simple one. And we get-- so the first and the last term drop out. And we're left with this.

And this is nice. It's sort of intuitive. It's saying that gamma, which is measuring the error, the gap between the two rays, is proportional to the triple product. And we know that when things work correctly, the triple product is 0, and then we expect gamma to be 0. So this is a way of calculating the discrepancy in the gap there.

Then we can also use the same sort of idea. Let's multiply by something that makes a lot of terms go away, i.e.: it's perpendicular to some of these things. We've already taken care of r l cross r r. That knocks out the first and last term.

What if we try and knock out these two terms? Well, that means we need to take a dot product with the cross product of those. And that gives us-- and so that's a way of calculating beta.

So that's going to allow us to determine how far out along the ray we get to the intersection, or almost intersection. And similarly for alpha, that'll tell us how far we are out along the other one. And so this is the third of our dot products, and it gives us-- so this allows us to calculate the three quantities, alpha, beta, and gamma.

And they're important. So first of all, gamma is the thing we're trying to make small. And then there's alpha and beta. And they give us the three-dimensional position.

But there's another important point here, which is, are these things positive or negative? Now, in the case of gamma, it just means does the right ray go below or above the left ray? So that's not that interesting. We just want to make it small.

But for alpha and beta, being negative is a real problem. See, we're dealing with equations for lines. Well, they don't stop or start anywhere, so we're actually looking at the intersection or near intersection of not just these line segments, but these two infinite lines.

And depending on the geometry, it may end up that-- suppose these are turned outward-- they're actually intersecting behind you. And so mathematically, you get a negative alpha and a negative beta. And then you have to decide if that's physically reasonable in your case.

It typically won't be because it means that you're imaging something behind the camera. So remember the formula-- big X over big Z-- if you make big X and big Z both negative, you still get some result. But what does it mean? You typically don't image stuff that's behind you.

So we typically check whether in the solution, alpha and beta are greater than 0. And remember, I said that there could be as many as 20 solutions. Well, one way to throw out most of them is to calculate this.

And a whole bunch of them will have negative values for alpha, or negative values for beta, or both. And so yes, in the mathematical sense those equations have up to 20 solutions, but some of the 20 solutions won't make physical sense. And so you can just discard them.

So we've got gamma. Oh, I guess we haven't got the actual distance. Gamma is just the multiplier. So the actual error distance, d, is gamma-- but we keep on coming back to that same triple product. That's the key to everything.

Now, what we'd like to do is have a closed form solution, where we give five correspondences, and we end up with five equations saying gamma equals 0, and solve them. But unfortunately, they're second order. So we've got five quadratics, and so far, nobody's come up with a closed form solution.

And there probably isn't one because you can, somewhat painfully, reduce those five equations to a single fifth-order equation, a quintic. And we know that quintics don't have solutions in terms of multiplication, and addition, division, and square roots. So now, of course, from a number-crunching point of view, who cares? If you have a fifth-order equation, you can find its roots.

But in the strict way of saying, is there a closed form solution, probably not. Although I haven't seen a paper making a clean proof of that. So what do we do?

Well, we know that we would like to minimize the error in the image, not this error out in the world. You can imagine that this could get huge. For example, if there's some object that's very far away, then that gap obviously gets large, even for a small error in image position.

So this d, this quantity, is not the one we want to minimize. But it's proportional to it. And if the image position is correct, then d will be 0, and vice versa. So they're related, and we can take care of that just by some weighting factor.

So triple product, and we're going to add it up. We're going to square it because it could have different signs. So we're going to minimize sum of squared error. And we're going to-- so what's that weighting factor?

Well, the weighting factor is the conversion factor from an error out here to an error in the image. And if I know these distances, I can figure out what that weighting factor is. However, the formula for it is a mess, so I'll just not give it to you. It's in the paper if you really want it.

And then I solve this problem. So the problem here is, minimize this with respect to b and the rotation, and subject to-- so unfortunately, it's not an unconstrained optimization problem. Now, how do I actually do this, because this weight depends on the solution.

So I'm going to do this iteratively. So basically, that conversion factor between error in the world and error in the image changes with my solution. And so I will use this particular set of weighting factors, and solve this problem, and then recompute the weighting factors, and solve that problem.

And fortunately, in most cases, it converges very quickly. And that makes the whole thing feasible. Without that, it's just intractable. Yeah, what is-- oh, sorry, it's my way of saying rotation, without specifying that it's a rotation vector or rotation matrix. So it's r parenthesis, dot, dot, dot.

So this is a square of a triple product. And of course, we can expand the triple product in many different ways. For example, let me expand it some other way that's more useful.

And now, we get to quaternions because we're dealing with this quantity, which is being rotated from the left coordinate system into the right coordinate system. And so we can introduce-- so if r l is the thing that I'm actually measuring in the left coordinate system, since everything over there is expressed in the right coordinate system, I have to rotate it into the right coordinate system. And I can do that using quaternions.

So let's call this triple product t for convenience. And so I'm going now from vectors to quaternions, but these are quaternions with the special property that they have zero scalar part. So remember that if I have two quaternions that represent vectors, the formula for multiplication simplifies. It's just the dot product and the cross product

So then one of the lemmas we stated without proof was a way of moving one of these multipliers to the other side by taking its conjugate. And the next thing I'm going to do is define a new quantity, for convenience, which is the product of the baseline and the rotation. It sounds really weird, but this is a very useful quantity.

So I take a quaternion representing the baseline, and why do I do this? Well, because it simplifies it and makes it symmetric. Yeah, I guess I'm mixing notations here. I'm sorry, so this is actually r l, and similarly here. It was coming from two different papers which use different notation.

So what's my job? My job is to find the baseline and to find d. And why is that enough? Well, because if I find d, then I'm done. I can recover-- so when I'm all done, I can recover b by doing this.

So when I'm done, I just multiply d by q star. And that's equivalent to this expression. And then, of course, q and q star is this quaternion identity with zero vector part. And b times the identity is just b.

So I have replaced the problem of finding b and q with the problem of finding d and q. That doesn't sound right. Because those quaternions have lots of components.

So b has four components and q has four components. So it sounds like we've got eight unknowns. And we know that the whole problem's constrained by 5 degrees of freedom.

So what's going on? Well, there are some constraints. For example, we know the baseline is a unit vector.

And it turns out we can constrain d to be 1. And we can show-- we're not going to do that, but-- that these are perpendicular to each other. That's fairly easy to show. Oh, well, this is really the same. Forget that [INAUDIBLE].

So we've got that constraint, that constraint, and that constraint. So now the counting is right. We've got eight quantities, eight variables, and we have three constraints.

So there are only 5 degrees of freedom. But it does make it much worse than absolute orientation, where the only constraint we had was that q be a unit vector, which was very easy to implement. Here now, we've got three quantities, three constraints.

There's some interesting symmetries here that I want to just briefly talk about, including the strange thing that you can replace-- you can interchange the left and right coordinates, the left and right ray directions, which doesn't make any sense. How can that be? It's like somebody screwed up and gave you the data for the left eye and switched it to the right eye. Yeah.

Well, I'm not going to say how to calculate the weight. Unfortunately, it's non-trivial. So again, what is the weight?

The weight is the relationship between the error in 3 space where the two rays intersect and the error in image position. And we can obviously calculate it based on the rays and all of those things. But it's a slightly complicated formula.

But the only important thing to remember is that we have to adjust the weight. So we do this calculation, we get an estimate of baseline and b and d. And based on that, we recalculate the weights. Because depending on the orientation, the relationship between error in 3D and area in the image will change-- hopefully not a lot.

And so this is, obviously, then dependent on having a good first guess. And so we'll have to talk about that because that's always a handicap if your iterative algorithm needs a good first guess. So we have a good first guess, we calculate the dot weights based on that.

We solve this optimization problem. We go back, recalculate the weights, do it again. And typically, we don't have to do it too many times.

So what's this about interchanging left and right? Well, it has to do with the idea that we're intersecting lines, not line segments. So when you interchange the left and right rays-- so here's our left and right ray drawn correctly-- and now suppose that somebody says, actually, I've mixed up the data, and I'm giving you that ray for the right eye and that ray for the left eye.

Oh, they intersect-- if these intersect, these intersect. So that happens to be behind the camera, and so you would calculate alpha and beta and say they're negative, so this is not right. But there are several symmetries like this which can be useful in the numerical calculation.

And so the triple product we're interested in is this thing. Hopefully, you got it there. And surprisingly, it's also that, where we've interchanged the q and it.

It's sort of like interchanging rotation and translation. That seems pretty weird. And if you don't believe it, you can expand it out in terms of components.

This is an expression in terms of quaternions, but we can rewrite it in terms of the components of the quaternions, which of course are these vectors. And then you'll see that it's perfectly symmetric. So if you look at this, you will see a number of symmetries.

One of them is between r l and r r. If I interchange left and right, nothing changes. And the other one is if I interchange the rotation and the translation, d and q appear symmetrically everywhere.

And why is this of interest? One reason is that it means that if you're searching the space of possible solutions, and you have an approximate solution, you can immediately generate other approximate solutions by making use of this symmetry. So altogether, I think there are eight, the symmetry of eight-- I think there's a symmetry of eight, so that you will find your solution more quickly because everything you've worked out has eight different interpretations.

And so by the way, that means if this is a solution, we know, of course, that this is a solution. Because minus q represents the same rotation as q, so that's not useful to us. But out of the formulas, you're going to get those as well. Then this is a solution and this is a solution.

So this gives us a factor of 2 factor of 2-- I guess that's where the 8 comes from. So when you solve this numerically, you may find that there are up to eight solutions.

Now on Stellar, there are two papers that describe this process. And it's kind of annoying that it's this messy. It would have been wonderful if someone had figured out a closed form solution.

I have to admit that I don't think there is one. I'm fairly convinced. And so we're stuck with this kind of numerical calculation.

So how do you actually implement this? Well, one approach is to assume you know one of these two unknowns. It turns out if you fix one of them, then instead of being quadratic, it's linear. There's a simple least squares solution, closed form solution.

And then assume that d is known-- you just calculated it-- and solve for q. And it's symmetric, so you would expect this to also be a simple least squares. And then of course, you'll have to do this again because now q has changed.

And giving a recipe like this doesn't prove at all that it's going to converge. But it does. And I'm not going to prove that it does.

So that's a very heuristic, very simple method that works. You can do better by using some nonlinear optimization package. And a popular one is this one, mostly because there are free implementations, freely available on the web.

And I briefly had a roommate when I started, Marquardt, and we had an interesting adventure where I drove him across the border to Botswana in the middle of the night. And then I never was in contact with him again. And he may be this Marquardt, but I don't know because he's apparently dropped out of sight, and not pursued science, and found something more lucrative to do or whatever.

Anyway, this is a very nice package which allows you to solve nonlinear optimization problems. And basically, you just have to give a bunch of equations that are supposed to be 0. And you have a bunch of knobs, a bunch of parameters. And it will tune the parameters until those equations are as close to being 0 as possible.

So it's like a black box-- you throw in your equations and hope for the best. But it requires that you have a non-redundant parameterization. I only mentioned this one now because so far, we've been able to do closed forms, so we didn't need to do anything like this. But this is useful not just for relative orientation. Lots of these kinds of optimization problems succumb to this approach.

So what's the problem? Well, the problem is rotation. If we do it as a normal matrix, we've got that problem-- nine numbers and only three degrees of freedom.

So one answer that is commonly given is Euler angles. And you know what I think about those. So let's x those out.

One that's actually used some is the Gibbs vector. So the Gibbs vector is tan theta over 2 times omega hat, where omega hat is the unit vector in the axis direction. And obviously, it's a vector, has three numbers, 3 degrees of freedom. And unfortunately, it has a singularity at theta equals pi.

And so that's a potential problem because rotation about 180 degrees is a perfectly legitimate rotation. Now in this particular case, you typically don't have the right camera rotated 180 degrees relative to the left camera. So in some sense, go ahead, use the Gibbs vector. It'll probably work.

Of course, what I'd recommend instead is unit quaternions. And we've worked out the details over there using unit quaternions. The only problem is they're redundant. There are four numbers, 3 degrees of freedom.

But this package allows you to add additional constraints. So you just pretend that this is another equation. And it will try and come close to satisfying that equation, while satisfying the other equation. You may need to play with weighting of different components, and that works very well. It converges pretty rapidly.

Just wondering where to go next, given that we don't have a lot of time. We talked about that. And this probably doesn't come as a surprise, if you know the rotation, there's a straightforward least squares method to find the baseline. And that's along the lines of what we said over here, so I probably won't bother with that.

Well, not much beyond. We have this formula here. So one of the ways we've been thinking about rotation is axis and angle.

And that's a pretty intuitive way of thinking about it. I mean, some people think all angles are intuitive. I think this is more.

And from there, we can go to this pretty straightforwardly. So the vector part is, in fact, in the direction of the axis. And it's just scaled according to the amount of rotation.

And then the scalar part-- not sure what else you can say about it. It has the nice property that if we combine two rotations, we just multiply these in this form. And the transformation of a vector is a little bit more complicated, but we saw the formula for that.

Then we use quaternions to represent vectors by making the scalar part 0, so that's a useful thing. And if you wanted to, you could use them to represent scalars by making the vector part 0. But that's not that interesting. So not sure if that answers your question.

STUDENT: Yes, definitely helped.

BERTHOLD HORN: Also, remember that there's a short, four-page blurb that summarizes everything you'd want to know about quaternions. Quaternions can be intimidating. I mean, when I looked at Hamilton's book, it's like 800 pages of dense math.

And then I took an engineering approach, and said, what do I do with this? How do I actually use this? And it turns out that this is all I needed to know about it. I don't need to know all of the really, truly esoteric stuff. Yeah, that's a great question.

So we've got several things we could use as an error measure. They all have the properties that if things line up perfectly, they're 0. So why prefer one over another?

And I cautioned against using the triple product in its raw form. And the reason is that they have different amplification factors. So suppose that the rays, for example, are almost parallel, and the object is one light year away. Then an error of one arc second is going to be hundreds of thousands of kilometers.

And you're going to try and minimize that, where some other part of your image might be an object that's closer by. So you want to compensate for that. And this way w is simply the relationship-- so here, very roughly speaking, z, f. And so the weight w is f over z.

So the triple product measures this error. And that can be huge just because z is large. But we're interested in the error in the image position that is the result of whatever algorithm we use to find image position.

And so we can reduce that triple product area into an image-relevant error by taking that. And here, the calculation is trivial. Unfortunately in our case, we got ray's going at different angles. And the calculation is in the paper online if you want to see it.

So the thing that comes next, which I'll just touch upon, is when does it fail? We always get to that point. So now we have a method-- well, we already know that we need five correspondences, so if we don't have five correspondences, it will fail.

But is it possible that there's certain kinds of surfaces, if we look at them with two eyes, there are ambiguities, and/or high sensitivity to error, so that we can't quite figure out what's going on? And so those things we were discovered pretty early on, like over a century ago, in an original paper they called "Gefaerliche Flaechen," which I guess properly translated means "dangerous surfaces" or "planes." And I guess the English terminology is "critical surfaces."

So the idea is that there may be cases where you're looking at an object of a certain shape that make this problem hard, or actually make your method fail, in that there is a lot of ambiguity. And to make that plausible, imagine we have a U-shaped valley, so like that. And then we have an airplane up here.

And we have some landmarks. And you remember, our job here is to figure out where the airplane is. It's the relative orientation. We have the plane flying along taking measurements, and we're trying to relate those measurements.

Well, we have an ambiguity, in that if we move that airplane along the surface of this circle, we don't change this angle. So that what am I measuring in the image? I can only measure angle. I don't know distances.

So if I move and the angles don't change, then I have a problem. And I've done it for A and C, but obviously, the same is true for A and B, and any other number of ground points you want to add. So that in this situation, where you're flying right above the axis of a semi-circular valley, then it's going to be impossible to distinguish different positions along here.

Now, this is a cross-section. This is 2D. So this isn't the whole story. But it gives you a plausible way of understanding why there might be a problem.

And this is why when they lay out the flight plans for mapping, they never do that. They prefer to do this-- fly the plane over a ridge rather than over a valley. Because here, when you move over, the angles between different images of different surface features do change a lot.

And so what we would like to do is get the 3D generalization of this problem. And that's-- well, I kind of gave away the answer by showing you those pictures. It's a hyperboloid of one sheet.

So we'll see that it's a second-order, a quadric surface. And it happens to be one with the right number of minus signs, for the one sheet-- I guess it's one minus sign. And so if we're looking at a surface like that, then there's going to be problem.

You say, well, who cares? I mean, how many surfaces? How likely is it I'm going to be looking at a hyperboloid of one sheet?

Well, that's true, but then if you're looking at a surface, you only have a small portion of it, the difference between it and a section of a hyperboloid of one sheet might be small. And in that case, you will be approaching this dangerous situation. That's one argument.

And the other argument is that there are other special cases of quadric surfaces, and in particular, of the hyperboloid of one sheet, which are more common. And the most common one is the plane. So it turns out that one version of this quadric surface is just two intersecting planes. And you can see why that could be.

So the equation for one plane could be a linear equation like this, equals 0. So that's one plane. And this is the equation for a second plane. I guess I wrote it in 2D, but just add z.

So each of these planes has a linear equation like this. And I can talk about it by saying, this is the plane where ax plus by plus cz plus whatever is 0. And if I multiply them together, of course, the product is 0.

And so what surface is that? Well, this times that equals 0 is the combination of two planes. And in our case, one of the planes goes through the center of projection, so it projects into a line.

So if I have a plane, and I'm looking straight along the plane, I just see a line. So it's even weirder, because it means that a planar surface is a problem. And what's more common than the planar surface? Well, maybe not in topography of Switzerland or something, but man-made structures other than [INAUDIBLE] and have lots of planar surfaces. And so we can't dismiss this entirely. We've got to talk a little bit about Gefaerliche Flaechen, which we'll do next time. And there's a new homework problem, sadly. I almost missed it because it seems like we just had one. But then someone reminded me that it shouldn't be due on Thanksgiving, and so it isn't. It's, again, going the Tuesday after.