[SQUEAKING]

[RUSTLING]

[CLICKING]

PROFESSOR: What we're talking about today is another patent, 7,016,539, and of course it's there in materials on Stellar. And this is going up one level. So it builds on what we've done before and its purpose is to detect objects, recognize objects, determine their pose in space, inspect objects, and do a few other things.

So what's the problem they're trying to solve? Well, we're trying to manipulate, perhaps using a robot arm or some other machinery, objects out in the world. And we want to know where they are and what they are. And we are going to start off by assuming that we have very accurate edge information. We went through all of that.

And so what came before this? So when we look at the patent in a minute, the prior art for doing this had four different components. So one was blob analysis, and more properly, this is binary image processing. So if there's a way of distinguishing object from background, we can create binary images.

And binary images have lots of advantages. One of them is they're much smaller because you've got one bit instead of eight. And they're much easier to process because there are only that many things you can do with one bit. And so they require less computing power, less memory, and in the days when memory was not so easily available, that was very attractive.

So what do you do with a binary image? Well, we're not going to do a whole lot of that. But obviously one thing you can do is find some kind of threshold, and then you get hopefully connected areas, and you can find properties of those connected areas, such as area, perimeter, centroid.

So there are a few things that are fairly easy to compute, area of perimeter and centroid being amongst them as is Euler number. So what's Euler number? Well, in this context, the Euler number is the number of blobs minus the number of holes. So for example, the Euler number for that letter would be 1 minus 2, or minus 1.

And it turns out that some of these can be computed in very efficient ways. So first of all, there's Green's theorem that where we saw that there are certain computations on areas that can be changed into computations along boundaries. And area, perimeter, centroid are all in that category.

And then there are some low level binary image processing operations. And it turns out that these three can be implemented in a very efficient parallel way by performing local computations. So for example with perimeter, all you need to do is look for places where there's a one pixel next to a zero pixel. And if you can identify all of those places and count them up, you're done.

So the idea is that there's certain things that can be done by counting local properties, because area is an obvious one. Is the pixel 0 or 1? And if it's 1, you count it, and otherwise not. And you can do it by going sequentially through the image, but that's of course very expensive. So you can imagine parallel methods, including parallel hardware. People built parallel hardware for this kind of thing which will compute area perimeter and very surprisingly also Euler number.

So that was one approach. And one problem with it is it involved the threshold. So if you're faced with a real world image, you have to somehow distinguish background from foreground. Maybe they are a different color. Maybe they're a different brightness, maybe different texture. So you have to find some way of separating, and that means you have to make a decision early on.

And as we mentioned last time. That's not always a good thing because your decision may be wrong, and there's no way to undo that. So now once we-- oh, I said "shape" in quotation marks. So a lot of these are computed using moments. So this is based on the zeroth moment and this is based on the first moment.

And what's a moment? Well, just to refresh your memory, so the zeroth moment is where you just take the integral of e of xy times 1 times 1. And if e of xy is a binary image, that's just computing the area that is 1. So and these others are computed in a similar fashion. So those are particularly easy to compute.

And what's the problem with this method? Well, in a lot of cases, there is no thresholding method. There's no magic trick that will distinguish foreground from background. In the autonomous vehicle, you're looking at cars ahead of you and some of them are brighter than the background and some are darker than the background. So that method is limited to cases where we have a clear distinction.

It's very important, though, to realize that these methods are widely used, but not on raw images. They used on the result of some computation where the computation combines brightness levels in some way to arrive at some conclusion and give you a binary result and then you need to know these methods.

OK, there's also something called binary template where you use a master image to define the object. So this comes up a lot in these methods where you have a sample that you consider the standard, often called the golden template. And you try to make it so that its image is as clean as possible and it doesn't have any defects and so on.

And you then compute something, for example, a template by thresholding. And that is then used as a method for recognizing and determining the pose. So you take that template and you move it around in the image until it matches what is there in the image. And we won't say much more about that. So the binary template method's accuracy is limited to perhaps a pixel.

Then we get to things that were more widely used. So let's call that two. We get to normalize correlation. And in fact, when cognates started that was their big claim to fame that they could do normalize correlation at high speeds and get reasonable accuracy, like a quarter of a pixel position.

And so what is that? Well, the idea is kind of let's try at all possible positions for the match and see which one is the best. So if we have two images, E1 and E2, where perhaps E1 is the golden master, then what we're looking at is something like-- so we take the difference of the two images and we-- well, let's square it.

So this is max, sorry, min. So it's clear what we're doing here. We're taking one of the two images and moving it around and trying to find the alignment where the difference between the shifted image and the other image is as small as possible. And so there's a related method called correlation.

Now, notice that right now I'm only allowing for translation. So we're just allowing for shift. And the reason for that is that, of course, we want to be able to deal with rotation, scaling, aspect ratio changes, slant, whatever. But this method is so expensive computationally that translation is about all we can do.

Why is it computationally expensive? Well, because basically, you have to look at every pixel for every possible position. So if the n pixels and you're going to try m different positions, that you need n times m computations. So with n being a million or something or more and the possible positions being also all of the shifts in the image, maybe half a million, you're talking a lot of operations.

These two are related in that if we expand this out, expand out the integrands, we get E1 where we have the correlation appearing in the middle. So maximizing this is minimizing that, assuming that this is constant.

And it should be, right? Because this is just take the second image, and you just square all of the gray levels and add them up, and that doesn't change. This one's a little bit more questionable because you take the first image and you shift it around and then add up the squares of the pixels. So maybe some of the pixels go out of the frame and stuff like that. But to a large degree, those terms are constant. So minimizing this thing basically means we have to maximize that.

OK, well, while I there want to relate this to our gradient based methods because we're talking about fixed optical flow, optical mouse, all of that stuff. And there we were computing an offset, typically a small offset like fraction of a pixel or maybe a couple of pixels. And so it looks like here's another way of doing it. We maximize the correlation.

So let's look at that. So let's suppose that delta x and delta y are small, then we can use Taylor series expansion and we get E1 of xy minus-- and so then we had a minus delta x Ex minus delta y Ey.

And then we have the difference of these two, which we can take to be the change in time because we can think of these two images as two images that correspond to two different times. So we're going to, let's see, minimize this.

So if I divide through by delta t and take the limit as delta t goes to 0, I get that because delta x over delta t goes to u. And of course, the minus signs don't matter because I'm squaring. And delta y over delta t goes to v and so on. So that's all very familiar. Yeah?

AUDIENCE: Can you remind us what the goal of all this is again?

PROFESSOR: The goal is to find an object, to recognize it, and determine its pose. So right now, what we're talking about is focused on this part, the determining the pose. So we have an image of what the integrated circuit is supposed to look like. And then we have an image of a runtime image of an integrated circuit.

And we're trying to put it in the right place for the next step of the manufacturing process, for example. And so we need to find out, how far is my runtime image shifted relative to the training image? And then I can move the stage and do the next stage of-- and so these are various methods of trying to get that alignment.

So for example, the sum of square of differences is saying, OK, I'm going to shift one of the images until it matches the other image as close as possible. And then interestingly enough, that's actually equivalent to the correlation, which isn't obvious. Why should the integral of the product of the two things be maximum?

And then, in turn, for small displacements, that's actually very similar to our gradient-based methods, right? We had that optical mouse. We wanted to know how far it moved-- very similar problem. And interestingly enough, if you read the patents on optical mice-- you probably didn't think they were, but there are dozens of patents on optical mice because, individually, a mouse doesn't cost hardly anything, so you're not going to make much money. But you can sell a billion of them, so people have thought it worth patenting. And almost all the patent recommend the gradient-based methods.

And when you take them apart and you reverse engineer them, a whole bunch of them don't. A whole bunch of them use-- so the patents all say correlations, sorry. I think I said it the wrong way around. The patents are all about correlation. Hewlett-Packard, Agilent, all of those people have patents on this.

But then when you take them apart and you reverse engineer them, which you're not supposed to do, you often find that they're actually using gradient-based methods because they're cheaper, more accurate, and whatever. The limitation is that the movement is not very large. So often, those mice run at a higher frame rate to compensate.

A lot of the gaming mice work that way. They have a very high frame rate, so the motion from frame to frame is very, very small. So anyway, so these are all connected. So I just thought I'd throw this in because we're talking about determining the pose. So in the simplest case, pose means the shift. Of course, it will be more interesting in cases where it also includes rotation, and maybe scaling, and so on.

So OK, so back to correlation-- so sum of squares of differences is used sometimes, but it has some drawbacks. So for example, if one of the two images is substantially brighter than the other, then they won't match very well, whereas the correlation will give you a high match anyway. Well, I guess, one problem with the correlation is, how high is high? Where's the threshold?

I mean, so when we plot that versus delta x delta y, we expect there to be a peak at the correct displacement. And that will still occur even if you-- it's obvious. If you take E,2 and multiply it by k, it's really going to disturb the sum of squares of differences. But it's not going to remove the peak off the correlation. The peak will just be k times higher. So that's one reason people very often use correlation, even though it has some other drawbacks, computational.

OK, so let's suppose we do that. Well, now, there are issues like-- as I mentioned, if the contrast is different, you will get a different peak value. It would be really nice if, instead, you had a number that said 0 is no match, and 1 is match. Here, we have something that can get big, and you're looking for the biggest.

But if it's not very big, do you know? How good a match is it? Are you matching up a picture of a cat with a picture of a dog? You don't know because you're just looking for the peak. So what to do about that?

Well, that's where a normalized correlation comes into it. So first, offset-- so if I add a constant to one of the images, it shouldn't change the match, but it is going to disturb this calculation. And it's, therefore, advantageous to first subtract out the mean.

OK, so we'd have a new-- so we compute the average of E,1, and we subtract that from everything so that if the overall level goes up or down, it has no effect. We cancel that out. Of course, it does mean now that E,1 prime can have both positive and negative values. Well, in fact, it will have both positive and negative values, and the same for E,2.

OK, so what does this do? This gets rid of any offset in the image brightness. And so it makes the process less sensitive to changes in the optical setup. But in addition, what we'd like to do is make it independent of contrast so that, in many circumstances,

If the contrast changes by a factor of 2, it shouldn't change your match. Of course, there are cases where you want to know about that because it could be an indication of a defect. But in this alignment problem, we may not want to care about that. And so in that case, what we do is we compute the correlation again as before. But now, we normalize it.

Oh, if you like, you can turn it into two square roots, whichever way you want to do it. So we're trying to remove as many things that could disturb the computation as possible. One is a shift in mean, and the other one is a shift in brightness and contrast.

And so this one here has the great feature that if we have a perfect match, if E,1 equals E,2, what do we get? E,1 equals E,2 means this one is the integral of E,1 squared. And this one's integral E,1 squared. This one's integral E,1 squared. And multiply these. Take the square root, so the whole thing, you can see how it's canceling out. Yeah?

Oh, yes, thank you. We went to all the trouble of subtracting out the mean, and then we dropped it. So OK. So the ideal value for this is 1. So if we have a perfect match, that is 1.

And you can show that this is limited to the range minus 1 to plus 1. It has to lie in that range. Minus 1 is you, in some sense, have a perfect mismatch. What that means is that one of the images the negative of the other-- so not very likely in practice. And 0 means, well, there's no real correlation.

Of course, if you take two arbitrary pictures, you're going to get some number. It's very unlikely you'll get 0. But you won't get a number close to 1 unless there really is a reasonable match. And so this is called Pearson's Correlation.

OK, so that means that the normalized correlation method for finding the pose has the property now that, not only can we find the peak, but we can have some measure of success. If the correlation is 0.6, that's not great. If the correlation is 0.95, well, that's probably a very good match. So it probably is an acceptable match. Again, as I mentioned, it's expensive. And one of Cognex's early claim to fame was that they were able to speed it up tremendously by clever programming and making use of the multi-byte instructions that were available on Intel platform.

So what's wrong with this? Well, there are a number of things. One of them is that if part of one of the objects is obscured-- so you've got your golden template where everything is perfect, and then you have the real-world thing where maybe there's a piece of paper lying over the corner of the object, or a second object is partially lying on top of it, well, that's going to mess this up because in that area, the E,1 and E,2 won't be matching.

Or if you have missing parts, you've got a gear with various teeth, and some of the teeth are not there, then that's going to affect this match. Nevertheless, it was used quite a bit for a while. And now, we're going to the patent which says, we can do better. Let's see if I have a bit more luck here than I did earlier today.

OK, so patent number 7016, so we've come up a bit. The last one was in the six millions. Method for fast, robust, multi-dimensional, pattern recognition. So it's a little confusing terminology.

Multi-dimensional? What is it talking about? It's primarily aimed at 2D images and flat surfaces. But multi-dimensional means they want to deal with not just translation but rotation, scaling, skew, and whatever.

And they emphasize that because the previous method, the best method was no more or less correlation. And it was so computationally expensive that you couldn't really do anything other than translation. And so this one is, quote, multi-dimensional.

And then the usual stuff-- the list of inventors. The assignee is, again, Cognex, and then the field, which we said wasn't particularly important, and then the references. So the first bunch of references are their own citations. And I guess, up at the top on the right, some starred ones, which the patent examiner threw in the pot along with the field from where they came. And then there are several publications general public, which isn't that common in patents.

OK, so abstract-- so they say it right up front. Disclosed is a method for determining the absence of presence of one or more instances of a predetermined pattern in an image, and for determining the location of each found image within a multidimensional space. A model representing the pattern to be found-- the model including a plurality of probes. So I'll talk about what probes are. Each probe represents a relative position at which a test is performed. So this is very different in nature from all of the other stuff we talked about before in that it's creating this abstract representation and using that in the match.

The method further includes a comparison of the model with a runtime image at each of a plurality of poses. So he tried different positions, rotations, et cetera. And a match score is computed. And then you can define a match surface. And you're looking for the peak of that surface. But now, remember, it's in the multidimensional space.

So you could be in five-dimensional space looking for a peak. And the match score is also compared with an accept threshold. So we do finally have thresholds, but the attempt is to delay any decision making, have thresholds right at the end when you have a lot of information.

And used to provide a location of any instances of the pattern of the image. So that may include the case where there is more than one instance of the pattern in the image or zero instances of the pattern in the image. OK, and then, of course, the examiner pulled out one of the figures which he thought was most relevant. And actually, let's just look at it.

So the idea is we have a training image, the golden standard. And this could be the top of Phillips head screw or something. And then there's, quote, training, which produces a model. And then there is a runtime image, when you're actually running the system. The robot's moving stuff around. And you combine the model with a runtime image using a list of generalized degrees of freedom-- so generalized in the sense they don't need to be just the obvious things like translation and rotation.

And so at runtime, you then do this comparison, then you produce a list of results-- a list of potential matches along with scores of those matches. And there are more references that went over from-- the idea is to get all of that other stuff on the front page, so if there's too much of anything, like references, they end up on the second page. And you can see these last few were put in by the examiner.

And I guess, Bill Silver, coming from an academic world, references-- you may recognize some of these names like Eric Grimson. Anyone know about Eric Grimson? Big shot here at MIT.

So at that time, he was doing vision. See what can happen to you? You can be working on vision. Next thing you know, you're an important person. So sorry-- shouldn't make fun.

Training image-- so there's Figures 1. We just saw that. And here is Figure 2. And now, this should look very familiar to you because most of this is what we talked about in the previous patent, right? So starting here, we estimate the gradient, the x and y components. We do the Cartesian to polar conversion, which could be done using CORDIC or some other method. And it produces the gradient magnitude and the gradient direction, again, at every pixel, so we can think of it as an image.

We do a peak detection in the direction of the gradient-- quantized direction of the gradient. And from that, we get column, row, magnitude, direction. And then we do the subpixel interpolation and all those clever things like the plane projection and the bias removal. And what we end up with in the end is a list of boundary points. So that's the output of that previous one.

There are a couple of extra things here that weren't the top row. So in the previous discussion, they did mention multi-scale, but they didn't talk about it much in the patent. But here, it's more important. And so we are working at subsamples of the image, possibly several different layers, different resolutions. And of course, before we subsample by Nyquist, we have to try and remove high frequencies, or we get aliasing.

So we have a low-pass filter or an approximation to a low-pass filter. So we take the source image, we low-pass filter, we subsample. And we may do this several times at different resolutions. And then we do all of this edge detection stuff. And we end up with a list of potential boundary points.

OK, now, talking of multiple scales, what would be nice to do is to use the course of scale that will work because the computational load depends, of course, heavily on the scale. If you're working at the full image resolution, there's a lot of computing going on. If you work at half resolution, it's a quarter the amount of work and so on.

So one of the aspects of this patent is, how do I decide what is the course's level that I can use? And they go through a whole story about that. And for some reason, it ends up being the first step out there.

And that's kind of there because you can't do any of the other stuff unless you've picked a particular resolution. But actually, the way it really works is you run it at several different resolutions. Then you look at the results and decide, which is the closest resolution that gives me reliable results?

OK, then process the training image to obtain the boundary points. That's that previous patent. Then we connect neighboring boundary points that have consistent directions. So you start to chain them together. Organize connected boundary points into chains. And we talked a little bit about how you break them where there's a sudden change in direction because that's probably a corner of some sort.

So here's a form of thresholding. As we said, it's going to find edges everywhere, including background texture. And some of those, quote, edges have the feature that they have very low contrast. So we could have a threshold and say, no, we don't want those edges. But that's, again, premature decision making.

What they do instead is chain them together. And if they're consistent, even if they're weak, it's still an acceptable edge. If, when you chain them together, you can only get very short chains, or their combined weight is very low, then throw them out. They're just due to noise.

So the thresholding has been postponed all the way down here. Divide chains into segments of low curvature separated by corners of high curvature. OK, and then, right now, we've got an edge point at every pixel at whatever resolution we're working at. And that may not always make sense. That may be highly redundant.

So instead, we're going to replace them with a set of points that have a desired spacing. So you can control, I don't want more than 1,000 edge points in my model. Or you can say, I don't want edge points to be further away than three pixels. But we do not use the edge points from the edge preprocessing directly. Instead, we fit to the edges, and you'll see that in the pattern.

So that's what this is. Create probes evenly spaced along segments and store in model. So that's creating the model. And the model also has a granularity, and it has a contrast. And the contrast doesn't so much matter in terms of magnitude-- more a matter of sign.

So in integrated circuit processing, often, you'll see a contrast between different materials. But depending on the lighting and depending on the process, the contrast may actually be flipped. So what, in the master image, looked like it was darker on one side and brighter on the other side may end up being reversed in the runtime image. And so you want to be able to cope with that.

I mean, there are other situations where that doesn't make any sense, where if the contrast is flipped, there's something horribly wrong, and you should just stop. But in the case of integrated circuit, the way light reflects off semiconductors is such that you may want to deal with the flipped contrast. And so part of the training is to determine what contrast the master copy has.

OK, so this-- that's what we've done so far. We find these edge fragments. And so here's a question for you. Where are the pixels? So are these squares the pixels? Or the center of the pixels at the intersections of the lines? Well, I'll let you ponder that.

OK, and this is explaining various parts of combining the edge-- connecting the edge fragments. So how do you connect the edge fragments pairwise? Well, you have to look at the neighbors. And that means a number of things. One of them is you have to decide what order to look at the neighbors in. And you may run into trouble, and you may need tie-breaking, very similar to the tie-breaking we talked about before.

OK, well, this is about the sequence that you go around to look at the neighbors. And so now, you've got them connected up pairwise. So these are associated in the data structure. And then, you start worrying about where to break them based on curvature.

So relatively low curvature up here, so those probably should stay together. And then here, there's a maximum of curvature, and you'd want to probably break up the-- and finally, you do a probe selection. So here's the top of the Phillips screw from a perfect, golden master image. And this is our model.

So we don't store the image. We store these probes. And again, these probes are derived from the edges, but they're not the edge points. They're interpolated based on how many you want.

And you can see that they're always pointing so they have a position, and they have a direction. That's the direction of the gradient. And so that's the model plus, as I mentioned, this granularity and contrast. But the main-- this is the key thing to remember that this is the model. And what are we going to do with that model?

Well, now, we want to know whether there's a place in the image where the image matches this model. And so we take the model, and we map it onto the image. And we don't look everywhere. We only look where the probes are.

So the probes are basically like things where you go to collect evidence. So like in this case, instead of having to deal with thousands of pixels, you're dealing with, I don't know, 100 probes. And you map them onto the image. You look at the gradient there. And you say, well, if the gradient is like that, and the model says it should be that, that's no good. That's an error.

So you compare the gradients with the gradients actually observed in the runtime image. And you may take into account the direction of the gradient and the magnitude of the gradient. Now, it turns out that, in many cases, the magnitude of gradient is not very reliable. It depends on all kinds of factors-- materials, lighting, accidents of alignment, whereas the direction of the gradient is much more likely to be maintained, even if there's change in illumination, change of material, and so on.

So when you do the comparison, you probably want to focus on the direction. So that's an example. But you can do more. And here's a different example.

So each of those probes has a position, a direction, it also has a weight. And that's basically a way of saying, how important is this one? You might, for example, have some weak edges in the master image and say, well, it's probably not as important that these match very well, so you can assign a weight based on whatever. But that's one reason.

Another way to assign a weight is to manually decide, these are important edges and these aren't. So in this case, the master object is the prototypical design for cell phone shape, as Apple says, rectangle with rounded corners, we patented that. And so that's that shape. And so we have all of these gradient directions along there, which is very familiar from the previous one. But then we also have these guys. So what's that?

Well, these were introduced manually, and they have negative weights so that if you are matching against this one, that means there's something wrong because the object is supposed to stop over here. And so this is to take care of the possibility that an object has certain symmetries. Like in this case, we can slide it in this direction. And we maintain. All of these still match all along the bottom edge.

But then we wander outside the acceptable area over here, and we get penalized because these probes have a negative weight. And the same with the movement in the vertical direction, which would maintain matches for all of these things on the side. But when you get out here, it'll say, OK, there's a negative contribution to the total.

Now, it's a nice idea, but it's not really used. And the reason is that the others, you can generate automatically. These require some human intervention. I don't think anyone's really used them in some automated way. Now, the others, you can just run our edge process and the flowchart I just showed you, and you're done.

OK, so here, they're describing their notation. And remember that this was in the heydays of, oh, we're doing object-oriented programming. Everything has to be an object. So all of these things now are objects. So what are these objects?

So the model is what you get from the master image. And what's the model will consist of? It contains probes. Those are the things that have position, direction, and weight, and probes created by training step. And then they contain the granularity, which was that one number that tells you what's the course of scale it'll work at-- and contrast, which we discussed. So that's the whole-- that's what a model is.

OK, probe object-- OK, so what's a probe? Did I miss something? No. OK, so what's a probe?

Well, it has a position, a direction, and a weight. And you can see what these are-- a two-vector, a binary angle, and the weight is a real number, which may be positive or negative. What's a compiled probe object?

So while this method is obviously much more computationally efficient than, say, normalized correlation because we're only looking at probe positions of which there might be dozens or hundreds instead of millions of pixels. There still is a concern with speed. And so translation is treated differently because translation is so easy to implement.

You know, it's just like an offset in the INJ coefficient of a matrix axis. And so all the other transformations, rotations, scaling, asymmetric aspect ratio changes, whatever are done at the abstract level. But right at the end, the translational part is dealt with in a very simple nested loop, INJ loop that just shifts things in pixel increments in the x and y.

So that means that your model can be brought into that world by taking into account all the other transformations. So the compiled probe object is the set of probes which are now specialized to image coordinates. And you can just superimpose that on the image and move it around to compute the match. But you're only dealing with translation at that level. So it's kind of a detail, but it's an important detail.

And so the compiled object is very similar to-- the compiled probe is very similar to the probe. It has a direction and a weight. The big difference is that the offset is an integer in pixels, as opposed to everything up to that point was a real variable. It wasn't quantized.

Well, let's see. Compiled probe-- so this is a function that compiles the probes. I'm not going to go through all of this, you'll be happy to know. So the matrix C is the non-translation portion of the map. So as I explained, that includes all of the transformations, except translation. Then what is the map? Well, the map is this transformation that has multiple degrees of freedom that we're trying to find the maximum of. Vector of-- let's go past that.

OK, so far, we've pretty much dealt with how to build the model, how to take the nice image of the object that's free of clutter and background, smooth, and so on, and create the model. Now, we start to talk about how to use the model. So as I mentioned, we map the model onto the image. It's very important to remember this because we're going to talk later about a different pattern where it's just the other way around, where we map the runtime image on top of the model. And it turns out, why do we do it this way?

Well, you could suddenly transform the image rather than the model, but the model has a few dozen points in it. It's very cheap to transform, whereas the image has lots of pixels. So it's going to be expensive. You can use Photoshop to rotate, translate your image. Certainly, all of that's possible. But we're trying to avoid that expense. And so in this case, it's important to remember the map is from the model, and it plunks the model down on top of the runtime image.

OK, now, what do we do? Well, at every probe position, we ask the runtime image, what's the gradient here? Well, we actually pre-compute the gradient, and we sample it. And then, we compare. And as I mentioned, we're more concerned about direction of gradient than magnitude. And so how do we score this? Well, if they're the same direction, that's great. If they're 90 degrees apart, that's horrible.

And so here's a grading function which is in degrees between the two directions of the gradients. And you can see that there's a bit of slop. So if you're off by up to 11.25 degrees, that's as good as 0. And that's because all of these calculations have some limitations, some noise, and so on. So you want to allow a little bit of slop in the direction of the gradient. But then you don't want to fall off a cliff. You don't want to say, OK, if it's 11.3 degrees, then it's no match.

So instead, you linearly decrease it until you get to 22.5, at which point, you say, OK, well, this isn't the match anymore. That's too much of a difference, half of 45 degrees. And it's arbitrary, how you pick these things. But the picking of them has an effect later on the quality.

OK, so this is a probe direction difference rating function which considers the polarity, which considers the contrast, the polarity of the contrast. So of course, it wraps around because when you get close to 360 again, you will allow a small-- I mean, this is just minus 11.25 degrees, and that's minus 22.5 degrees. So that's the function that's used in rating how well a probe matches the corresponding point in the runtime image.

Now, if you say, in my situation, there may be a reversal of contrast, then I shouldn't use this because then 180 degrees difference in direction should be accepted as well. And so that's what the next figure is. So here, we are allowing a small slop around 0 and a little bit about 180 degrees, and the other angles are not accepted. And that's important for some integrated circuit situations, as I mentioned.

There's a drawback, of course, which is, inevitably, we'll be looking at completely unrelated parts of the image. And there's some small chance that they happen to have the right gradient. So how often is that going to happen?

Well, up here, it's fairly rare because we can roughly say this, let's see, maybe a 20-degree band. So that means 20 degrees out of 360, 1 in 18. So the chance of random matches actually pretending to be good matches is 1 in 18. Well, here, it's twice as much. So this is not as robust against noise as that one. So if you know that the contrast is not going to be reversed, don't use this. Use that.

And so similarly, you might say, well, I want a wider margin. Let's go out to 45 degrees. Well, that's all very well, except that means that you're going to be accepting more of these random alignments.

Then I mentioned that we can also look at the gradient magnitude. And in some cases, you might want to just ignore it because it's not as reliable. Or you can use it directly as a weighting factor. The bigger the gradient magnitude, the more likely this is important.

Or you can say, well, that's true up to a point. So you set a target level, which is, perhaps, the gradient, the magnitude, the contrast on the master image, the training image. And then it saturates. So it goes up. The bigger the gradient magnitude, the better.

But it has a limit. It doesn't just keep on growing. So there's a function for scoring how well the directions of the gradients match. And this is for scoring how well the magnitudes match up. And remember, these are applied at every probe proposition. So you map the model onto the runtime image. Then at every probe position, you check what the gradient is, and you can pair it with the gradient in-- so that's the key part of the method.

Then, we get to the degrees of freedom. So obviously, translation, movement, in x, movement in y are degrees of freedom. But we also want to deal with rotation, and scaling, and-- OK, let me go onto the next page.

So this is a description of what a generalized degree of freedom is as far as they're concerned. And then here's some examples. Well, actually, this is probably all of the ones you might ever want. So we have rotation, and it has, as parameters, an angle in degrees. And does it wrap around?

It's important that the search method be informed about whether when you go off the max you come back at the min, Well, yeah, for rotation, that's certainly true. And the cycle is 360 degrees. So it turns off the 360 degrees. And how does it map the coordinates? Well, the standard formula, except maybe the minus sign is in a place that we wouldn't want it to be.

And then you need to figure out the steps. You're going to explore each of these degrees of freedom in steps. Now, in translation, you might just do pixel steps. But what should you use for rotation? So it's not obvious. And then how does it change the scale of the image?

Well, in this case, it preserves the scale. When you rotate something, it stays the same size. Then, you might have shear. Now, that means that you're turning a right angle into a non-right angle. And here's the matrix transformation for it. And this used to be much more of an issue than it is now because, today, images typically don't have much of a shear in them.

So in the old days, images were made by devices that had electromagnetic deflection. And so the x-axis was only as perpendicular to the y-axis as somebody made it when they assembled it. So it was quite common to have some kind of angle other than 90 degrees-- not hugely different. But one or two degrees, it could be off. And so you might want to do a search for that.

Or suppose that you are working in a world where things aren't exactly two-dimensional. Something might be-- instead of lying on the conveyor belt, it's tilted slightly. Well, then, if you look at the image of that, it will not image as a rectangle. It will image as a rhombus. And so you might want to, in that situation, allow for this transformation. Of course, if that doesn't happen to you, leave it out because it's going to increase the cost of computation.

Then we get to size-- now, scaling. Now, you might say, well, we'll just increase linearly the size, the width, and height of everything. But that's not a good idea because what's a linear increase for a small object is relatively large compared to the same linear increase applied to a large object.

So it's more reasonable to work in a logarithmic scale. So that means the increment in the size is not 0.01, but it's 1% or, let's say, 10%. And then when you increase again, it'll be not 20%, but 21% and so on. You get the idea. So that's where this comes from.

So for the logarithmics size factor, it does not cycle around. And this is the mapping for it. We have the exponential part in there. And this is how the scale changes in that case. And so here's one that only scales x, and here's one that only scales y. And of course, these are redundant because in the case that x and y scale the same, we just use that rather than these two, and so on.

And then there's an aspect ratio one. Again, this used to be more common than now because the scan in the y direction was very different from the scan in the x direction. Now, people make incredible high-resolution, high-accuracy image sensors, where you know that the axes are as close to 90 degrees apart as they can be.

And then the pixel size is either perfectly square, or at least you know what it is. You've got to be careful about that because, often, it will be something like 504 out of a 497. And when you look at the chip, it looks like it's one to one, but read the specs.

OK, so you may need to have to take care of aspect ratio and so on. Now, the bottom ones, as I said, this is somewhat redundant. Like, these down here are linear scale factors, and I already said, those are probably not as interesting. OK, so those are, quote, generalized degrees of freedom. I mean, in some sense, all you really need for 2D work is translation, rotation, and maybe scaling. But they allow for more than that.

OK, some weird terminology. Probe MER is the probe Minimum Enclosing Rectangle. So for some operations, you can cut down the computation by checking only the minimum and closing rectangle. And so they carry that around as well.

And so think of this multidimensional space, now, of pose. So pose is translation, rotation, et cetera. And it can be multidimensional.

And one of the things that can happen is that, when we do this search for the peak, we end up in this space with values that are very close together because we got there in different ways using translation, and rotation, and scaling. And we don't want to report both of those values. And so we need to have a way of telling where the two poses are very close together in that space.

And that's what this is about. We won't go into that in detail, but it's to remove any overlap. OK, overlap examples, flowcharts-- flowcharts used to be quite the thing. And I guess, let's get to the text-- lots of figures. Oh, OK.

Oh, then the bottom level, the translational search should be done in an efficient way and also should allow for different resolution. And so here is our sequence of search patterns that can be used efficiently at the lowest level. And you'll notice that they're basically organize around the hexagon. And that's, as I mentioned, because you get a 4 over pi advantage in terms of work done versus resolution.

And so here's the lowest level that's just like red/black checkerboard, and then there's the next highest scale, and so on. So they get a slight-- and this isn't huge. Some of the other things they do get you orders of magnitude improvement.

This gets you a relatively small improvement. But working on these patterns is helpful and-- sorry. And remember, we talked about peak detection and how you had to deal with ties. You had to have a tie breaker. Well, it gets a little bit harder on the hexagonal grid. But here's one solution.

So you check that the center point has a larger value than these three, and it has a larger than or equal to value than those three, and that ensures that you're not going to detect adjacent things that are-- OK, oh, OK, now, let's see if I can find this place.

So one of the nice things they do is they explain the terminology. Oh, here we go. So there's a common term in patent law which is that a common expression, which is that the inventor can be his own lexicographer, which just means that if you define the term in some way in the patent, then that's the way it reads for the patent, whether there's someone else that agrees with it that's the right usage for that word or not. Sort of like I was listening to a program on Canadian Broadcasting.

And he kept on talking about the neoliberal agenda and stuff like that. And I'm like, wait a minute. Everything he's talking about is right wing. How come he's calling it neoliberal? Well, he wrote the book. He called the book The Neoliberal, blah, blah, blah. So he gets to define the term. And it's my fault for using my interpretation of it.

So here, they are very explicitly do that. They define all of the important parts. So Object-- what is an object? It's any physical or simulated object or portion thereof having characteristics that can be measured by an image forming device or simulated by a data processing device. Now, this is classic generalization, right?

We're just talking about images taken with a camera. And the patent attorney says, wait a minute. This has much wider applications. So they're talking about not just images that were made from real objects, but graphics. So this could be used on data simulated by a data processing device. And you can read into this X-ray images. It doesn't have to be visible light images.

Then what's an image? A two-dimensional function whose values correspond to physical characteristics of an object-- and then I guess we have to go up there-- such as brightness, radiant energy, reflected or otherwise, color, temperature, height. You can see how much fun they had trying to generalize this as much as possible. Brightness, the physical or simulated quantity represented by the values of an image regardless of source. So that's to make sure that just because you said brightness, it doesn't exclude anything else, like temperature, X-ray intensity, I don't know.

OK, granularity-- so that's where they talk about the size in units of distance below which spatial variations in image brightness are increasingly attenuated. So when you blow an image up, there comes a point where it starts to look blurry. And you can relate their definition of granularity to that point. It can be thought of as being related to resolution.

Boundary, what's a boundary? An imaginary contour, open-ended or closed, straight or curved, smooth or sharp-- take your pick. I love this thing where it says, for example, it'll say something like the words and and or mean both and and or. It's like, what?

That way, they generalize everything. And it gets worse. I mean, to us thinking about logic, and and or are very different concepts. But you can now go through the whole document, replace all occurrences of and with or, and try all possible combinations, and it's still covered by that patent.

Gradient-- well, we know what a gradient is. Vector at a given point in an image giving the direction and magnitude of greatest change in brightness at a specified granularity.

Pattern-- a specific geometric arrangement of contours lying in a bounded subset of the plane of contours, said contours representing the boundaries. So could you have come up with that definition? I mean, this takes art. You get a degree in law before you can do this.

Model-- a set of data-encoding characteristics of a pattern to be found for use by a pattern finding method. Training-- an act of creating a model from an image of an example object or from a geometric description of it. I haven't discussed that. I've assumed so far, we're making the model from a real image.

But of course, if you have the CAD for the object, you might very well say, well, that's even more of a golden standard. That's really the truth. And so you could produce the model from mechanical drawing or equivalent computer vision. The downside is, if there's some process in manufacturing that does not reproduce the CAD exactly, you won't capture that because you're working-- and this is very common, for example, in printed circuits or integrated circuits.

You have a perfectly rectangular layout with sharp corners. Well, when you go through and expose the material and put it in the chemical bath, that corner is not going to be a sharp corner. It's going to be rounded. And so those are situations where working with the real object is better than working with a CAD.

OK, pose-- a mapping from pattern to image coordinates-- we discussed that-- representing a specific transformation in superposition of a pattern onto an image. So as I mentioned, this is unusual in that they've gone to the trouble of breaking out and listing the terms that they will use, which is very good because, then, if there's any argument about what the specification means or, indeed, what the claims mean, then that's right defined there.

So let's jump ahead. So the rest of this, the usual stuff. The background of why this is the best thing since sliced cheese or sliced bread and then, what is the prior art? So they talk about all of the things we talked about-- binary templates, blob analysis, correlation, and tough transforms, which we haven't talked about.

And then they get to the summary of their invention, which pretty quickly goes into the discussion of the figures and details of the figures. And then there are some equations. See here, the explanation, it's fairly long. Here are the figures. I'm trying to get to the claims.

Wow, there's even an integral sign. That's neat. Yeah, wow. Oh, that's the noise calculation integral from 0 to 360. And it gets kind of interesting. There's some real math there. OK, come on. Oh, here we go, Claims. I preferred-- OK, must be down at the bottom here.

OK, what is claimed is a method for determining the presence or absence of at least one instance of a predetermined pattern in a runtime image and for determining the multidimensional location-- location, in parentheses, pose-- of each present instance, the method comprising. And so then they go on and describe the details. And we'll talk some about those details.

They also didn't say this up front, but they incorporate the possibility of inspection and the possibility of recognition. So the recognition is what's going to happen if you have a very good match. And so if you have the possibility of seeing more than one object, well, you run this process for each of them. And presumably, most of them won't be a good match. And one of them will. And so that's your recognition.

The other thing-- inspection-- this method works even if parts of the object are obscured or parts of the object are missing. But of course, the match is going to decrease in quality as you remove things. So if your gear has lost one tooth, you'll still get a pretty good match. If it's losing two teeth, not so good, and so on.

So that obviously tells you you can use it for inspection. So after you found the object in pose, and you've matched it, you can then qualify just how good a match it is. And that might be a way of doing rejection of a part that's not satisfactory.

So this one is mostly method claims. I'm trying to remember if this-- the method-- and then a lot of dependent claims. The method of claim 1 plus this. And I can't remember if this one still has apparatus claims. The method of claim one-- these are all method claims. Yeah, all method claims.

I think what happened is that the legal situation changed slightly, and they didn't have to do that game anymore of having both apparatus and methods. OK, where are we? OK.

So I hope the idea of probes is quite clear. And the model, the model is composed of these probes. And the huge advantage of using the probes compared to using, say, correlation-- because we're now only looking at a small number of points in the gradient image. Now theoretically, we could just compute the gradient when we need to match it. But since we're going to do this matching many, many times with different orientations and positions for the model, there's a definite advantage to pre-computing the gradient.

So we use that other patent to finish that job, and then we jump in and superimpose-- we map the model on top of the runtime image, and there's just a few places. And we're collecting evidence. So at every one of those probe places, we're asking the question, you does this support a hypothesis that there's an object with this pose? And then we add up all of the evidence. And that gives us a score.

And then from that, we can build a score surface in this multidimensional space. Now, this is much easier to visualize if we only deal with translation. Then we have a function that varies with x and y, and we're looking for the peak. And that's our translation. We're done. But of course, we're also interested in dealing with rotation, scaling, et cetera. So it's a little more difficult, but same idea.

Let me just talk a little bit about this noise issue. So it's called noise, which is kind of a misnomer in the patent. Remember, we had this scoring function for where this was the difference in degrees between the model probe direction and the runtime gradient direction. And I said that there's some probability that if you're just superimposing this in some random place in the runtime image that you're going to get a match. And so how bad is that? Well, obviously, what you want to do is just integrate-- what shall we call this thing? I don't know, scoring function.

So if this was 22.5, and this was 11.25, we can calculate this integral. I forget what it comes out to. And then also, obviously, if we use the method that is insensitive to the sine of contrast, then we double the amount that we're going to get from random matches.

Now, there are two aspects of that. One of them is we can calculate how much is this going to offset the result, and we can subtract, so we can take out this error. But the other one is, of course, that it's going to contribute to the noise in the result, that it's not as good. So here's an idea. So this is kind of a nuisance to compute this, to take the two directions, take the difference in angle, and then look it up in this functional table.

How about if we just take the dot products of these two vectors? If you have two unit vectors, we can just take the dot product. And that's going to be the cosine of the angle between them, right? So that would be very cheap to compute because then, in the probe, I just store the V1. And in the runtime image, I've got V2 computed by that previous patent.

And then, I just take the dot product, which is going to be cheaper than using that method because, in this method, I have to use atan to get the angle. And I have to use it twice, once in the training. So who cares? You only do that once. But once in the runtime image. So what's wrong with that?

Well, that means that my weighting function looks like this because if they're lined up, I get 1. If they're opposite each other I get minus 1. And so that's a potential. This isn't in the patent. This is an alternate, not as good, scoring for gradient direction function.

Now, this is the one that takes into account polarity, but I can also think of the one that is insensitive to polarity, which would be the absolute value of that. And so I think you can see what the problem, which is this is going to be producing a fairly large result just from random matches, right? So we'd have to take the integral of cosine, the absolute value of cosine from 0 to 360 degrees and, I don't know, I forget what it is-- pi over 4 or something. But it's a large number, whereas the number here is very small.

Let's estimate this. Oh, we did. We said 20 over-- it's like, 1 in 18. I don't think that's right, but it's approximately that. This is going to be much larger, like 1 and 1/2 or something. So yes, that is much easier to compute, but it's not nearly as good, so they didn't use that.

So what remains to say about this? So this was very successful. And we do still need to talk about the scoring functions. How are these measurements combined?

But then, just stepping up a level, what's the disadvantage of this method? It's quantized. We are quantizing pose space. So we're only looking at, let's say, rotations that are multiples of 5 degrees. I mean, we get to decide how much to quantize it. But it is quantized.

How do we make that decision? Well, we could make it very fine, but then the computation is very slow. So there's a trade off there. And so how can we improve on that?

Well, one way is to use the method with fairly coarse quantization. But then, we have to search the whole pose space. Initially, we don't know anything. It could be rotated five degrees, 143 degrees, whatever. So we're forced to search the whole pose space. But once you have a number of potential matches-- and, typically, you want to retain more than one-- then you can search with finer quantization near there. And so that's another important aspect of this, and we'll need to talk about that.

So the idea is that it's kind of like multi-scale, that we-- at a core scale, we run through the whole course pose space, and then we use a block just around the areas that seem to produce a good result to get a more accurate result. So what I was-- again, the patent is online. You should probably look at it.

And what I still have left to do is talk about the different scoring functions because, in some cases, we would like a score that we can compare to an absolute threshold and say, OK, if it's bigger than 0.95, then we definitely have a match, that kind of thing we talked about with normalized correlation.

In other cases, we just want to get a result as fast as possible. And we find the peak, and we don't really care how big the peak is. So those are two different computations. And depending on what we're doing, one may be more efficient than the other.

Then, we might want to do things like remove this term, which I think this is actually 3 over 32. But in any case, it's not too hard to carry out. So let's call that N.

Do we actually want to remove that? Well, that's going to mean that there is more computation, but it will be more accurate. So depending on where we are in the process, we're getting close to the answer. We want to refine it to make it as accurate as possible. Then, we'll throw that in and just bite the bullet on it taking more computation.

OK, a couple of points that I highlighted here-- one of them is that if we're working multi-scale, then we'll want to use different probes at different scales. And this is kind of an even more sophisticated way of proceeding, where, instead of just fixing the quantization or fixing the granularity, we try it at different granularities. And if we do that, we need different models for each of those granularities.

Then there' a need for fast low-pass filtering because we're trying to build these multi levels. And we'll talk about that because another patent, which is much easier than this one, gives us methods for very rapidly performing convolutions and low-pass filtering.

Gradient direction is more reliable than brightness-- contrast, we talked about that. Each test provides direct evidence of pattern presence. We talked about how the probes will contribute evidence. Probes are not restricted to the pixel grid. Oh, that's an important one.

I mean, first of all, they're derived from edge points that we already interpolated. So those were already not on the pixel grid. But then, in addition, we throw those out, and we put our own probes in separately.

Accuracy limit-- oh, accuracy limited by quantization of search space. Why am I emphasizing that? Well, because that's going to take us to another pattern that does it a different way, which is not limited in that respect. So why are we talking about this one? Well, because the other one needs a good first guess. And so this is the way to get a very good first guess. And then the other one can refine it. And it's not limited to discrete steps in the pose space. OK, that's it for today.