[SQUEAKING]

[RUSTLING]

[CLICKING]

**BERTHOLD HORN:** We're starting a new part of the course where we'll be dealing with industrial machine vision. And in particular, we'll be looking at patterns on that. And we'll start off with one by Bill Silver at Cognex. So Cognex is arguably the leading machine vision company. And it's an early starter. I tried very hard to persuade him not to join a startup because the success rate wasn't very high at the time and maybe still isn't.

Anyway, he joined anyway. And he's their technical guru. And there's a bunch of patents that describe what they did. At this point, you can't manufacture integrated circuits without machine vision. They're needed all over the show for alignment and inspection. And you can't manufacture pharmaceuticals without machine vision. For example, there's a mandate that the label should be readable.

So what's the language of the mandate? It's not, most of the labels should be readable. It says, the mandate-- the labels should be readable. That means every single label has to be inspected. Well, I don't think anyone wants people to do that. So of course, it's done using machine vision. And those are the two areas that Cognex managed to capture a large market, starting off with integrated circuits mostly in Japan. So most of the market was there.

Anyway, let's dive into this patent and learn a little bit about patents. So first of all, why patents? So the basic idea is that you come up with an idea to do something, produce some chemical, build some machine. And you set up to build such things and sell them. And your neighbor looks over and says, oh, that's nice, you can make some money that way, and basically competes with you without having put in the effort of actually coming up with the invention.

And so the whole idea of a patent is a compromise, whereby you get a limited monopoly to use your invention in return for explaining exactly how it works so that after a certain amount of time, anyone else can build it. So it's a contract with society where you get some benefit for a while. And in return, there's some benefit long-term for society. And this is how they get you to explain in detail how things work.

And there are different opinions on various types of intellectual property. And we won't get into that. There are obviously some benefits and some disadvantages to this. And it's changed over the years. There's a constant revision. And it's remotely possible that some of the things I tell you are no longer true because they were true two years ago and whatever.

So let's look at what's there. So first of all, then, we'll get into the technical stuff later. But the structure of this and the metadata, so obviously, there's the number. So at this point, we're up to six million. I just got notice that one of our x-ray patents got issued. And it's in the 10 million range. So since 2002, we've added a lot of patents.

And a lot of companies, large companies in particular, file patents at an incredible rate. And why is that? Well, it's largely because they want ammunition in a patent war. So say if you're IBM and there's Microsoft, I hold 6,000 patents in my hand. And if you promise not to sue me, you can use that technology if I can use your 8,000 patents, that, kind of, thing. So patents can stand in the way of doing useful stuff. They can also sometimes prevent wasted money on litigation.

So here's the date of the patent. We have a title, apparatus, and method for detecting, and subpixel location of edges in a digital image. So this is aimed at mostly a conveyor belt world, where you're looking down at things and you're measuring positions and attitudes. And it's also aimed at integrated circuit world, where you're largely dealing with two-dimensional images.

So one of the things we know is that images often have large homogeneous regions of uniform intensity. They aren't very interesting. The interesting stuff is where there's a transition between different brightness levels. And so edges are where it's at. You can greatly reduce the bits needed to describe something by just focusing on the edges.

So that's what this is about. And it's about finding them to subpixel accuracy. And that's very important, because you can always get more accuracy by adding more pixels. Instead of using a million pixel camera, use 100 million pixel camera. And you get 10 times the accuracy. And obviously, that's expensive. And also, if you can do that in software for relatively low-cost, then that's a huge advantage. Or if you have that 100 million pixel camera, you now have 10 times the resolution of whoever else is working in that field.

What accuracy can you achieve? Well, it's fairly straightforward to get to 1/10 of a pixel. And at that point, you start to see all, kinds of, problems that you didn't think of. If you push it hard, you can get to maybe 1/40, which is what they claim. And that's, obviously, huge. Two dimensions, that's a factor of 1,600 in terms of accuracy.

So that's what this is about. And then the authors are listed. And the assignee is listed. That is, the authors don't necessarily get the benefit. Typically, if you work at a company, they make you sign over some piece of paper that says that whatever you invent is theirs and so on. And I guess MIT wasn't that careful when I was hired. And I don't think I ever signed that paper. But it didn't help me much.

Anyway, then we have the date filed and the fields of search. So there are some numbers there that tell you what area it is in. Those fields of search were established a long, long time ago. And so there isn't even something for computers. It's a subpart of electrical stuff. If you're into rubber making, there are 10 categories for different types of rubber. Of course, that was-- so those categories are things that patent examiners know a lot about. But they're not particularly interesting.

And what happens is that you submit this thing. And it has to be in a form that's acceptable. And there are lots of rules. It used to be you couldn't have any equations in there. Well, this one has equations in it. So that obviously is no longer the case. You couldn't use grade levels. It had to be black and white, line drawings. And so the only way you could get something like grade levels was halftones, dots with varying sized dots and dot spacing. You can't have color.

The language is somewhat arcane. There are particular terms that you come into again and again, like plurality, I can never pronounce that word, which means several, and things like comprise. So one way to say the apparatus contains certain things is to say that. But that's not very accurate. So comprise means there is at least one. There may be more. And so the patent lawyers make their money knowing this stuff.

Then we have references. And there's a bunch of patents listed along with their fields. And the asterisk means that they were added by the examiners. So it goes down. And the examiner sits on it for a while. And it used to be-- well, it changed in history. But in the '90s, when this was submitted, 1996, there was a multi-year delay. And at that point, Congress changed the rules a little bit, including charging you for that process. And it speeded up a bit.

And the patent examiner will add things that they think of. You can see here that the inventors only thought of this one patent by, well, one of the inventors. It's like when you write your technical paper, there are probably going to be references to your own previous papers. So they didn't think of all these other people as being relevant.

And then there's more, other publications. To these people, patents are important. Technical papers aren't. But they're still listed because the author is from the scientific world, in some sense. And it doesn't-- it says, let's continue on next page, because there wasn't enough on the front page, enough space on the front page to list all the papers that he wanted to refer to.

And then there's the abstract. And this patent is unusual in that it's understandable. It's detailed. It's technical. There's not much-- and it's because it was written by an engineer, mostly. So if you read the abstract, you get a pretty good idea of what it is. There's a figure. This figure is chosen by the patent examiner out of the figures that are submitted with a patent as somehow most distinctive, most likely to let you know what the patent is all about. So we'll get back to the abstract later.

So let me go to the next page. So here, you can see a whole bunch of additional papers on edge detection. Of course, edge detection is an old topic in machine vision. It goes back to the 1950s, if you can believe, when people first started scanning in images and finding a need to somehow compress the information, measure things, and so on.

So one of the early famous papers is by Roberts, who was at Lincoln Labs in 1965. And I guess his input device was a drum plotter. So in those days, one of the exciting output devices was a drum that rotated. And you had a ballpoint pen. And you could control its position and, in some cases, color. It might have multiple tips. And you could make great plots of circuit diagrams, PC board layouts, and so on.

And so what he did was he replaced the pen with a photodetector, a vacuum tube photo detector. And then he scanned his 8 by 11 black and white glossy photographs. And then he came back the next morning, and looked at the data, and he had a very simple edge detector, which was misleading in the sense that he had a scanner that produced incredibly good images, because there was just a single detector he could afford to build, a 12-bit a-to-d.

He had much better quality images than we would get out of vidicons and other devices later. And so his edge detector, which worked OK on his pictures, wouldn't work for people on other pictures. And so for a long time, there was a competition, OK, my edge detector is better than yours. And they all had names. And I remember being at a conference, and standing in the passage, and trying to be social, and chatting with this guy I'd never seen before.

And we got on to edge detection. And I said, well, it's too bad that all the edge detectors that have people's names on them are worthless. Well, he was Irwin Sobel, whose name is on one of the edge detectors. And he's forgiven me since then. But he's still trying to claim his legacy and make sure that everyone knows that he invented that edge detector. And he's blogging and whatever. Anyway, so those are some more papers.

And then we get to the figures. So the top one is the Roberts cross edge detector. And you can see that it's a directional derivative. The left-hand one is a derivative in the 45 degree direction. And the right-hand one is in the 135 degree direction. And why that? Well, we'll talk about the advantages of that type of operator later.

And then so Irwin Sobel's operator is figure 1B, which is somewhat advantageous. It takes more computation, is somewhat more resistant to noise, is not as high-resolution, and so on. And then Bill Silver, having been to my classes, was interested in hexagonal teselations. And so he proposed these alternate operators, which would work on a hexagonal grid.

Unfortunately, no one ever built hexagonal grid cameras. They have some advantages in terms of resolution and rotational symmetry. But it's not a huge advantage. It's like 4 over pi. So there's an advantage of some sort. But it's not huge. And so the extra trouble of working with a hexagonal grid apparently was too much for engineers.

This one here goes off in three different directions, which is appropriate for hexagonal grid. But of course, that's redundant. If you have de dx and de dy, you've got everything. So there are two degrees of freedom to the brightness gradient. You don't really need three numbers. So he came up with this alternate pattern here, where this one is estimating the derivative in the x direction and this one in the y direction. And the square root of 3 is because these cells are further apart than those cells. So you have to compensate for that.

So a key step right up front is to convert from Cartesian to polar coordinates, in that you want the magnitude of the gradient and its direction rather than the brightness gradient itself, the x and y components. And so up there is a formula. You could take the square root of the sum of squares. And you take the arctangent. And part of that is put out as a straw man, which is like, why on earth would you do this? This is very expensive.

Now maybe today, we wouldn't say that. But if you have 100 million pixels and you're trying to do things at 100 frames a second, you still don't want to really take square roots and arctangents. Actually, square roots, depending on where you grew up, you might know that square roots are as easy to calculate as divisions. It's just a slight twiddle on the algorithm. And unfortunately, in the Western world, we don't teach that.

So people think of square roots as these really nasty things. But unfortunately, the people who design digital computers were also not from this other world. And so for us, square roots are more expensive than division. Anyway, the argument in the patent is, we can do it this way. But it's expensive. So let's find a better way of doing it.

Then figure 2B is an alternate solution using lookup tables. The idea there is that you encode the x and y components of brightness in a small number of bits. Then you stick the bits together. And that's the address of a lookup table. And it creates the table index, lookup table. The lookup table gives you the magnitude and the direction. And that's a great solution if arithmetic operations are expensive, which was true at the time, and if memory access isn't a problem.

Then, well, things keep on changing. So for a while there, arithmetic operations became cheap, because people just built 32 by 32 bit multipliers and be done with it. And cash misses were a big deal. So you didn't really want to use large lookup tables. And so for that reason, you might want to go back to a method that does a lot of arithmetic rather than have a large lookup table. And that's changing again and so on.

So anyway, at the time, the idea of using a lookup table was also not particularly attractive. And so he came up with this method illustrated down here, which we'll talk about, CORDIC. So CORDIC is a way of estimating the magnitude and direction of a vector from its two coordinates. And it was actually pioneered in World War II.

You probably know that a lot of early computing machineries were built for unpleasant purposes like war and making sure that you could aim your guns in the right direction. And being able to compute arctangent and square roots was very important. And so people came up with this method, which is an iterative method that rotates the coordinate system to bring it more into alignment with the brightness gradient vector.

And at each step, it reduces the error, the difference. And when you then add up all the results, you can get the magnitude and the direction. And amazingly, you can do it without much arithmetic. You only end up needing a shift, which costs next to nothing, adders, and subtracters, and an absolute value operation. So if you think about rotation, you think of a matrix, cosine theta, sine theta, minus sine theta, cosine theta, and multiplication.

So trigonometry, that's expensive. Multiplication and addition, expensive. Well, this is a method very cleverly designed to not do any of that. And it can be implemented at low-level. So in Intel architectures, you have an assembly language. But you also have clever things you can do with instructions that access multiple bites. And so this can all be implemented extremely efficiently. So that was the motivation for that.

Then once we know the brightness gradient in magnitude and direction, one of the things we want to do is find the places where the gradient is large. That's where the edge is. But we don't want to just take a local maximum because along an edge, the gradient is going to be large. So there'll be some places that are local maxima. But they're meaningless. They're arbitrary points along the edge.

What we really want is to look across the edge and take the maximum in that direction. It's 1D, not 2D. And so we need the direction of the gradient so that we know which direction to search in. And we need to have this maximum finding, which is called a nonmaximum suppression, meaning we'll ignore stuff other than the maximum. A funny term for finding the maximum. But that's what it was called. And since we're working on a discrete grid, we have some limitations.

And particular on a square grid, we have to somehow decide to quantize the possible directions of the gradients. We compute the gradient direction with high-accuracy using any of the three methods we talked about. But then we're restricted to-- we only have those pixels. So what do we do? Well, we quantize it into these different directions. And for purposes of finding the extremum, we pick one of eight possible compass directions. On the hexagonal grid, we have six possible directions. Same idea.

Now if we want to, we can look further afield. So the previous figure was looking at a 3 by 3 area of the image. So we just had 3 by 3 values of gradient up there. But if we're not happy with that course quantization of direction, we can go to 5 by 5. And now we have a larger range of directions. So they talk about this. But it's not what was actually implemented. And of course, you can go further.

So the next step is, now we're looking along a certain direction in the image. And we found the place where there's an extremum. And now we'd like to know where the actual peak is to subpixel accuracy. And so one thing we can do is, figure 4A, those are the three values. So imagine that 0 is the pixel where you found the maximum. And plus 1 is one over. And minus 1 is one over in the other direction.

And those don't need to be in the x direction. They could be in the diagonal direction. Or they could be up and down. But three values anyway. And we can fit a parabola to that. Well, that's the best we can do. We've got three numbers. We can only fit something with three degrees of freedom.

And so ax squared plus bx plus c, we can fit that. We could fit a cubic. But it would be ambiguous because there would be one parameter that was hanging loose. We could make it anything we want. So we fit a parabola. And we know how to find the peak of a parabola. We just differentiate that the result is equal to 0. And there we are. We can find a peak indicated by that dotted line.

Now if our model of the world is Mondrian, i.e. We have patches of equal brightness with sharp edges between them, then we know that the brightness will be constant in each patch. And if the edge is very sharp, we might expect there's a different local fitting, which is shown on the right. So up to the edge, the brightness is increased. The gradient is increasing. And then it's going down.

And it might seem a little bit odd. But that will give us a different result. And again, we need those three numbers to make that fit. And we can find the peak, find where those two lines intersect. And we can find the dotted line in figure 4B. And those two will generally be different. So which one do we take? Which one is more accurate? And so we'll get to that in a moment.

Then there's something they call the plane position interpolation. So the idea is that when we found the extremum, we quantize directions, horizontal, 45 degrees, vertical, et cetera. But the actual gradient to higher precision, we know its real direction to maybe 8 bits of accuracy. And so one step we can perform is to actually go in the gradient direction from the center pixel and find the place in that direction where the edges.

So the diagonal line is our quantized approximation of gradient direction. And at 413, we found its peak using the method of figure 4. And the error of 410 is the actual gradient direction. And now what we do is we draw a perpendicular to that gradient direction and from going through 413. And we see where it intersects 415.

So the idea is that with the edge, when we find an edge point, its position is well-defined in the direction of the gradient perpendicular to the edge. But along the edge, of course, it isn't. We could be anywhere on the edge. So this is just like our aperture problem optical flow discussion. And so which of those points do we take along the edge?

Well, one argument might be, the further you get away from stuff that you really know, the less accurate it's going to be. So let's find the closest point. So we've found an edge. Which point on the edge do we actually record in the output data? Well, we pick the one that's as close as possible to G0. And that's this point that we constructed this way. So that's that construction.

So then we discover that we can get a pretty good subpixel accuracy, maybe 1/10 of a pixel. But to go further, we need to take into account the fact that our constructions are based on some assumptions about how the brightness gradient varies with position. And we have those two models. And we could come up with some more.

And what is the real thing? Well, the real thing is going to depend on properties of the optics of the camera you're using. It's going to depend on the fill factor of the chip that's sensing the image. It's going to depend on how accurately in focus the image is. So the edge transition, of course, is not a perfect step transition, which is a good thing.

But its actual shape is not something that you may be able to predict ahead of time. And it's the same for every possible camera and image situation that you can think of. Now if it doesn't fit those two models, what happens? We'll get the wrong answer, but not hugely wrong. So there's a bias introduced by our choice of that model. And we can calibrate out that bias.

And so that's what he's doing here. So these are exaggerated curves. So S, one of them is the actually computed position based on our little peak finding and parabola fitting. And S prime is the actual edge position. And ideally, it would be an diagonal line if you plot those two. And because of the factors I mentioned, it's not a straight line. It has a little bit of a bow to it.

And as I mentioned, this is hugely exaggerated. The bow is much less. But it shows that it might be three different shapes for three different cameras, or focused positions, or phase of the moon, whatever. And so what do we do? Well, think of this. This is now just a correction on top of a correction. So we don't need to be hugely sophisticated.

So he comes up with ways of removing that bias. And in particular, you can think of just powers of S. If you take S to the 1, then that's our diagonal line. If you take S 1.1, then it's a little bowed down. If you take S 0.9, it's bowed a little bit the other way. So that the idea here is that we can increase the accuracy even more by finding out for your particular machine vision system, what the S is that gives you the best fit.

And while we're there, notice that when our quantized gradient direction happens to be diagonal, 45 degrees, then the points we've got are actually further apart by the square root of 2 times as far apart as when we're horizontal. And so you would imagine that the bias would be different. And indeed, it is. And so another refinement of the patent is that you make this bias depend on the gradient direction to get even higher accuracy, so.

And so this is the overall diagram of the whole thing. So we start with the image. We estimate the brightness gradient, which we call Ex Ey and he calls Gx Gy. Then we find the gradient magnitude and direction, G0 and G theta. Then we do the nonmaximum suppression. And we find the neighbors.

We detect the peak in that direction. And from that, we get a position of the edge point. And we still know the gradient there. So we use that as well. And then we interpolate using that parabola or the triangle. We compensate for the bias. And we do that plane position method that finds us the closest point to the maximum that is on the edge.

So here's the patent itself. This is, of course, part of the materials in the course. So you can study this yourself. And I'm not going to go through it word-by-word. But just focus on the sections. So it starts off as usual with the title. And then the field of the invention, is this about making rubber from stuff coming out of trees? Or is this about machine vision? So there's very short-- this invention relates generally to digital image processing and particularly to edge detection in digital images.

Then we get to the background. So the background is where you acknowledge that people have already done certain things. So that's considered a prior art in the terminology of patents. And the main point of this section from the point of view of the inventor is that you end with where you say, therefore, you need this thing I invented because all this other stuff, it's great. But it doesn't really solve the problem.

And so in this case, what do they say? Consequently, there is a need for an inexpensive and/or fast method of high-accuracy subpixel edge detection. So most of this discussion is about, why do you want to detect edges? Why is that important problem? And how have people been detecting edges? There was Roberts' gradient operator. There was Irwin Sobel and so on. And then it ends up with, well, all of those are too slow, too inaccurate, blah, blah, blah. And here's why you need what I've got.

And then there's a summary of the invention, which is a short version of what it's all about. The invention provides an apparatus and method for accurate subpixel edge detection. So that's another thing about patents. Originally, the people who came up with the idea of patents, the way it's written in our Constitution, decided they didn't want to patent ideas, abstract ideas. They didn't want to patent mathematical formula.

And so they didn't know about programming. But the Supreme Court later reinterpreted that to mean that you can't patent programs. Software is not patentable. And so for a long time, people in machine vision weren't able to patent their stuff because it was basically software. So then they came up with the idea, well, what if the software lives in some physical object? You can patent that.

And so then they-- all this convoluted language to try and make it be a physical thing. And then for a while, what you did was, because it was unclear how the Supreme Court would come down on this, you chose both. So basically, you invented an apparatus, a box that has an SD card with a program in it or whatever, and the method separately. And then if someday, they decide that methods are not patentable, then you're OK because you've still patented the apparatus.

So this meant that in many cases, as here, there are twice as many claims at the end as, really, there need to be because they had to basically use the same language, just with apparatus replaced with method and some other small changes. So it's apparatus and method. There was a case in-- so patent cases are litigated in various ways.

One special one is import. So if someone's trying to import something that you think violates your patent, you can go to a particular patent court, which is in Washington. And you can argue your case there. And there was a case not related to this patent, but a closely related patent. And there's a company in Canada, Matrox, which basically didn't build a machine. But they had software that implemented exactly what was in that patent.

And so the expert witnesses for the two sides got up and presented all their very hairy theories about how this might work. And by the way, the expert witnesses get to look at your code. So if you're trying to make an argument, you don't want to show the other side of the code. But the expert witness for the other side can look at the code.

And so you put that person in a room where they can't copy things. They're not allow to bring their little USB sticks with them. And they can look at all of your code. And so it was pathetic, because the expert witnesses were being asked questions like, well, isn't finding a maximum of that function the same as finding the minimum of minus that function?

And the judges, they were completely out of their water. And in the end, fortunately, the judges found a way out of it, which was to say, oh, it's all software. And that's not patentable. So there were three weeks of people discussing cubic interpolation and stuff like that. And then in the end, it was thrown out on a technicality, if you like. So anyway, so that's part of that. So the summary of the invention.

Now in terms of patent litigation, most of this is irrelevant in the sense. That is, the lawyers immediately go to the claims. We will get to claims in a second. So this is an unusually clear explanation of the method. Again, part of what you want to do when writing a patent is make it as broad as possible so you have a clear idea of how to do it.

But now someone else, your opponent, is going to figure out a way of not doing things exactly the way you describe, making some small change so he can say, oh, this isn't your patent because I did that instead of that. So one of the jobs that you and your lawyer have is to figure out all the ways of modifying it so that it's still the same idea.

So probably, things, as an inventor, you don't think of first because you just want to get it to work and to work well. And now you've got to think about, oh, what if I replace ultraviolet with infrared, what if I use sound instead of radio waves, and so on. And so as a result, patents written by lawyers are almost impossible to read because they've generalized everything to such a large degree. And so here, Bill Silver refused to let them do that. He just wrote it himself. And it's clear instructions. This is what you do. And it's great. It's very pleasant to read.

Then we get to the figures. And so there's first of all, a short section that lists all the figures and just gives them names. And then there's the detailed description of the drawings, which is actually the detailed explanation of how the thing works. And the other conventions, for example, all of the numbers that appear in boldface refer to items in the figures. And if they appear in more than one figure, they better have the same number, rules like that.

Stuff can be rejected because you didn't follow the rules, because your rectangular boxes aren't quite rectangular. And also, in one place, you called the kernel 131. And in another place, you called it 141. You can't do that. And he has a formula for the Cartesian to polar conversion, equation 1A and 1B, which, as I mentioned, used to not be something that you'd see in patents. And really detailed. We're only up to figure 2 at this point. So they're giving really detailed explanations. And we'll talk about some of that.

More formulas. These are the formulas for finding the peaks of the parabola in the top case and of the triangular wave form in the second case. And this is the formula for the bias, which is probably very hard to read. But it's the absolute value of 2S raised to the B, where B is that thing that I said could be near 1. And it explicitly says it has to be bigger than minus 1. But other than that, it can be any value.

Tables. Let's see. Oh, OK. So now we get to the part that the lawyers will find the most interesting. Where is it? It starts up there. What is claim biz? And for some reason, it's not a separate section. It's just a convention. The words what is claim appear. And now you list the claim. So those are the particular things that you think that you came up with and you want to protect. And the other side is probably going to argue that, well, actually, it's obvious. It was the end prior art, whatever, so.

So it starts off quite a long first claim. And I'll read it out because this is what we're actually going to study, the algorithm for that. And an apparatus for, that's claim 1. And claim 11 is going to be the same thing. Sorry, claim 21 is going to be the same thing, which says a method for. So this is apparatus 4.

Detection and subpixel location of edges in a digital image, Said digital image including a plurality of pixel values, each pixel value having been associated with respective pixel point on a regularly spaced pixel grid, said apparatus comprising-- and comprising, remember, a special term here in patent language. And then it lists three items. For some reason, they are all labeled. Well, they all start with A. So it looks like the they're all labeled A. But they're not.

So the first one is a gradient estimator for estimating gradient magnitude and gradient direction at a plurality of regularly spaced gradient points in said digital image, so as to provide a plurality of estimates of gradient magnitude and gradient direction, each set estimate of gradient estimate and gradient direction being associated with a respective gradient point on a regularly spaced gradient grid.

You wouldn't think that it would be so difficult to talk about, OK, we're going to compute the image brightness gradient. But there it is. And I suppose the idea is that, to us, it's obvious. But this is supposed to be something that anyone could understand. So that's one. We're going to get the gradient.

And the second one is-- oh, and we're also going to get gradient magnitude and direction. So that that's component one. Then component two is a peak detector, cooperative with set gradient estimate, operating such that gradient direction associated with each gradient point is used to select the respective set of neighboring gradient points.

So that's the quantization of the gradient direction. So we pick a certain set of-- certain direction, and operating such that gradient magnitude associated with each gradient point is compared with each gradient magnitude of said respective set of neighboring gradient magnitudes as to determine which offset gradient magnitudes is a local maximum of gradient magnitude inappropriate said gradient direction. So in short words, it's a nonmaximum suppression in the quantized gradient directions.

So that's two. And then the third and last part is a subpixel interpolator cooperating with said peak detector, operating such that said local maximum of gradient magnitude and a set of neighboring gradient magnitudes are used to determine an interpolated edge position along a one-dimensional gradient magnitude profile, including a gradient point associated with said local maximum of gradient magnitude and so on.

So that's the interpolation step, where we fit the parabola and we find the peak of the parabola. And that's all of claim 1. So why do there need to be other claims? Well, the thing is that someone might come along later and say, well, wait a minute. We already invented that. And so you would claim one is blown out of the water.

So what you then do is you specialize it further. You add in other things that are in your specification and refine it more, and more, and more in the hope that if in litigation later on, the early claims that are very broad are thrown out, then the narrow ones will still stand. So if they also violate all the other conditions, then you're good.

So for example, claim 2, so we have claim one further comprising a plain position. So that last step that I described, that plane position step where you intersect the gradient and another line, that isn't in claim 1. So if supposing that claim 1 gets thrown out, then if you were using this plane position feature, then you are going to violate claim 2. So that's a conditional claim.

And it goes on like that. We won't, obviously, read through all of these. Claim 3 is the apparatus of claim 2. So claim 2 depends on 1. Claim 3 depends on 2, which in turn depends on 1. And claim there has a gradient directional line. So it's giving a particular way of computing that plane position point.

And then there's a whole bunch more, obviously. And they get more and more detailed so that you protect it in case it turns out that, well, claim 1, if you put together the writings of Roberts and Irwin Sobel, it's there. And so you need something else. Then let's see. Where does it get interesting again?

Claim 11 starts all over again where claim 1 did with a change that this is a function plus means claim. So here, instead of saying that you have an apparatus for gradient estimation, gradient estimation means for estimating gradient magnitude, peak detection means for subpixel interpolation means.

So it's not specific about whether it's apparatus, or method, or whatever. And again, this has to do with the history of patterns, that they needed to do that. And so everything is repeated. So we have claims 1 through 10. And then it's repeated in function plus means up to 21. And then at 21, everything is repeated with apparatus replaced with method.

So you end up with a relatively simple problem. And it ends up with 34 claims? 34 claims. Now these days, they penalize you for that, because you've have to pay extra if you have more than 20 claims. So things change. So for example, at this time-- let's see. Before this, the rule was that your patent would be valid for seven and 1/2 years after it issued.

And that was different from the way the rest of the world did it. And also, it was unfair because things could be delayed in the patent process. So something could issue 10 years after you submit it. And then you get another 10 years just because of that delay. And they were tricks of using that.

For example, Jerome Lemelson claimed that he invented machine vision in 1956. And he had a string of patents that documented that. And some of them were called submarine patents, because you didn't know they were in the pipeline. But they depended on earlier patents. And so anyway, so Congress decided to ax that idea. And so things changed.

And I guess already, at this point, the rule was 20 years after filing instead of seven and 1/2 after issue. Also, they removed the rule that who gets priority. So suppose you think of some idea and someone else thinks of the idea, and you submit patents, and then you have these two conflicting patents. Who gets the patent?

Well, it used to be that whoever invented first. And so in the old days, all engineers carried around books with squares in them and every page numbered. And every day, they would ask their buddies to sign out page. Why? Well, because that way, you could document that you thought of this idea on April the 3. You didn't have it fully fleshed out.

But the pages were numbered. So you couldn't cheat and tear out pages or add pages. And people's signature was on there saying that, oh, yeah, I saw this. He actually wrote this at that time. And that was a giant pain because everything you did, every time you thought of something, you immediately had to write it down. And so they decided, no, forget that. Plus, it was very hard to prove sometimes.

And so the new rule is submission. When you send it in, that's the date. And if you take your time sending it in, too bad. You lose. So anyway, so that's a typical patent. It's a relatively simple one. But it has some interesting machine vision aspects, which we'll talk about now.

So we mentioned that in many cases, images are composed of patches that are pretty much uniform in brightness. And the interesting stuff is at the transitions. And I mentioned that this Dutch artist, Piet Cornelis Mondrian, got rich by drawing things that just had rectangular patches of color. And is it art? I don't know. I'm not in that field.

But anyway, so I call this the Mondrian model of the world. And so in that model of the world, rather than have millions of pixels, you can condense things down into just talking about the edges. Or if you want to find where something is on a conveyor belt, you just need the edges. If you're trying to line up different layers of an integrated circuit mask, then you just need edges.

So what's an edge? Well, and what is an edge detector? So again, unused, remarkably, this is actually well-pointed out here. So edge detection can be defined informally as a process for determining the location of boundaries between image regions that are different and roughly uniform in brightness. So great.

And then in more detail, an edge can be usefully defined as a point in an image where the image gradient magnitude reaches a local maximum in the image gradient direction. So that's important. It's not just the local maximum. It's in that direction. And by the way, they mentioned another one, which is where the second derivative of brightness crosses 0 in the image gradient direction. So think about that.

Now that makes sense. So if the first derivative reaches a maximum, how do you find the maximum? Well, you differentiate one more time and set the result equal to 0. So an alternate way of determining where an edge is to look at second derivatives and look at 0 crossings. And that was used as well.

And by the way, then they mention multi-scale. Now remember, this is a while back, but even then. Thus, it is known in the art to perform edge detection at a plurality of spatial frequencies or length scales as appropriate to the application. So we're just going to pretend that we're working directly at the full resolution.

But imagine that for many applications, you would not want to do that or you would want to work at multiple resolutions. And so for this patent, we're not going to get into that, as they don't get into it. So we have an image transition. So first of all, here's an ideal edge. So this is a cross-section across the edge. It's a step function.

Well, it turns out, actually, that's not good. We don't want that. We don't want infinite resolution. That seems crazy because you think that the bit of the resolution, the happier we should be. Well, the problem is, suppose we now image this onto a device that has discrete pixels. So we're measuring the brightness at those points.

Now can we tell where the edge is? Well, suppose I move the edge a little bit. Nothing changes in terms of the brightness measurements at those points. So actually, in this very idealized case, I can move it a whole pixel width back and forth. Nothing changes. So conversely, that means I can't measure it where it is within the full pixel position. So this is actually undesirable. And basically, this is a form of aliasing.

So we don't want a perfect step edge. We want something that's band limited so that when we sample it, we're not introducing artificial frequency components, which is one way of thinking about what's going wrong here. So we're looking for the edge. And we've said that one method is to take the derivative and find the peak of the derivative.

And then we mentioned also that possibly, you might want to take the second derivative and look for zero crossing. So this was actually a popular game for a while, that we would be looking at second derivatives. And in the case of images which are two-dimensional, what's the generalization of the second derivative? It's Laplacian.

So instead of these edge operators, which give us an estimate of the brightness gradient, we'll take the Laplacian and we find the zero crossings. And one of the appealing ideas about that was that, well, zero crossings are closed curves. It's like a contour map. So you have this-- we've talked about thinking of the image as a surface in 3D, where height is brightness.

Well, then we can draw contours, isophotes, certain level. And we can draw the 0. Well, not with images themselves. But once we take the Laplacian, we'll have both positive and negative values. And we can draw the contour at 0. Well, contours are closed, other than you won't know where they disappear off the edge of the image.

So that was very appealing because a lot of times, other edge detectors would peter out. And then you wouldn't quite know where the edge was and so on. So that was an interesting game for a while. But it turns out that you get worse performance in the presence of noise. And we can discuss that in terms of convolution and so on.

But we won't do that for the moment. So second derivative. And it's briefly mentioned in the patent. But it's not pursued. And then what does that correspond to in the original? E. So what we're really doing is looking for an inflection point. What's an inflection point? Well, imagine that you're driving along this curve.

You're turning left. You're turning left. You're turning left. You're turning left. Oh, now I'm turning right. Now I'm turning right. So the inflection point is where you're changing directions. And in terms of derivatives, it's the maximum of the derivatives. So those are different ways that we could define where the edge is.

So let's talk a little bit about brightness gradient estimation. So you saw in the figures there a number of operators. So this was Larry Roberts' idea. And it's very easy to compute, very cheap to compute. But it's a 45 degree angle. So what was the idea here? So why not operate in x and y? Well, it turns out that this makes it possible to have the two operators refer to the same reference point.

So where are we estimating the derivative? Well, you might say, OK, I'm estimating the derivative here by taking the difference between this value and that. And someone else might say, well, we're estimating the derivative here because I'm taking the difference of the value here and the value there. And we'll see in a second that neither of those is well-founded. It's more reasonable to say I'm estimating the derivative there.

And the resistance to that, of course, is that's not a grid point. So it's not a pixel. It's halfway offset in pixels. But who cares? Maybe it's a good place to estimate the derivative then. And so Roberts used these two operators. And then he used the sum of squares and noted that, conveniently, that's actually the same as if you'd computed the sum of squares of gradients in the original coordinate system.

Because if you do the 45 degree rotation, cosine 45 degrees, sine 45 degrees, it's 1 over square root of 2. You get the same thing except for proportionality factor, because these are further apart than the pixel spacing by a square root of 2. So these aren't the same. But they're related by some number. They're proportional to one another. So that is Roberts' gradient.

And then we had Irwin Sobel. Well, let's first address this question about estimating the derivatives. And we may have done a little bit of this already. But let's do it more carefully. So we have this computational molecule for E of x. And we know from Taylor series. And so now, we can use that to-- so when we do that, the f of x cancels out. And then we divide by delta x.

So we get-- that's the part we want. We're trying to estimate the derivative. So that's perfect. And then we divide by delta x. So we get delta x 2. So this is the lowest order error term. So that's what we want. And that's the part we don't want. And when we talk about how good a formula it is, we'll be looking at two things.

One is the order of the lowest order error term, which here is second order. And then the other one is the multiplier. I suppose we have two methods that have the same order over the lowest error term. Then we can compare them based on the size of this multiplier. But the more important one first is the order. So this one's not very good, because even it works perfectly on a straight line. But even if there's a little bit of curvature to it, it's going to get the wrong answer because of that.

So let's instead look at f of x plus f of x minus delta x. So here, we said, OK, this is our x. And this is x plus delta x. And we're obviously trying to get the derivative at x. Now we're saying, OK, this is x. That's x minus delta x. And we're trying to get the derivative there.

And if we go through the same arithmetic, we get the same size error. But the sign is flipped. So well, what you might say is, well, gee, let's just average these two. Then we can get rid of that low order error term. And then so if we average them, which means the sum of these two divided by 2, then we just get f of x. That's what we want. And this cancels out.

Well, but then they're higher order terms. So we better fill in the higher order terms. So here, we got delta x squared over 6 f triple prime of x. And so what we notice is that now, we have a higher order error term, which is desirable. So this formula will not just work perfectly for a straight line. But even if the straight line has a bit of curvature to it, a second derivative, as long as the third derivative isn't too large, we'll be OK.

So what does averaging these two actually mean? Well, it means that we take-- that in effect, we've used that operator. All right? And that's a perfectly reasonable approximation to a derivative. We're subtracting two things. And we're dividing between the distance between where we've measured those two things.

And then you might say, well, what if, instead, I take this idea? But now I'm saying, I'm going to find an estimate of the derivative there. And the objection would be, well, it's not a pixel position. But that doesn't matter. We can have even derivatives be offset by 1/2 a pixel as long as we just remember that, mentally, that's happened.

Now I can go through and use Taylor's series, blah, blah, blah. But actually, all I need to do is use this formula and divide delta x by 2, because this is the same thing with delta x divided by 2. So in this case, my formula is going to be f plus and then delta x over 2 squared divided by 6. So now I have to compare two operators that have the same lowest order error term. And now I look at the magnitude of the factor in front of the error term.

And obviously, this one is a 1/4 of the size of that one. So this one has an advantage in that respect. I can get a relatively high order error term. And I can have a small multiplier so that the error is actually smaller. And it makes sense. Over here, we're comparing things that are relatively far apart and seeing-- so obviously, they will be affected more by higher order derivatives than in this case, where I'm looking at things that are close together.

Now that's fine for Ex. But how is this going to work with Ey? Because if I take that idea in both directions, so here is my Ex operator. And then there's a potential Ey operator. Well, that's not going to work, because this one is a good estimate of the x derivative there. And this one is a good estimate of the x derivative-- y derivative there. And those aren't consistent.

And so what I do, well, one way to deal with it is that because these now, if I go through that Taylor series story, these are good estimates of the x derivative here. And this is a good estimate of the y derivative here. And oh, those are the same point. So that's how we get to that.

And when we talked about fixed optical flow, we already mentioned that that's the way you want to compute the derivatives. And well, there, we needed not just Ex, Ey, Et, we need not just Ex and Ey, but we needed Et. And so there, we're dealing with a whole cube. And so, for example, to compute the derivative in the y direction, we can do that.

And under materials, there's the detailed explanation of how to do the fixed optical flow. And that's in there. So now the story doesn't end there, because now we might talk about efficiency. And so for example, how much work these operators-- well, here we have three-- we can subtract these two. That's 1. Subtract those two. That's 2. Then we add them up. So it's 3 operations for this and 3 operations for that.

But actually, they share these subcomputations. And this one's 1 operation. And that one's 1 operation. So 3 plus 3 is 6. So 1 operation and 1 operation. And then we combine them. We add them to get E sub x. And we subtract them to get E sub y. So that's 1 of plus 1 of. So a total of 4 ofs. So by cleverly arranging the computation, we can cut it back from six operations before. And you might say, who cares?

Well, the thing is you're doing this in each of a few million pixels. So this is one place where, actually, efficiency does matter. And I won't go through this. If you do this the obvious way, it takes, I don't know, 21 ofs to do this. If you get Ex, Ey, Et, the obvious way, you get 21 ofs. And I'll leave it to you to-- and people sometimes come up with surprising solutions that beat the obvious solution by a significant factor. And by the way, there's Roberts' cross operator. So he was way ahead of his time.

And what about Irwin Sobel? So Irwin Sobel looked at this stuff. And this is where he got to. He said, well, this is obviously a good way of doing it. It's an estimate of the derivative at that pixel. And it doesn't have a low order error term. But then he needed to do it in 2D. So he replicated it. But why did he replicate it the particular way he did?

Well, we can think of it in this way. I'm leaving out the consonants. But basically, we do an average. So this is a convolution. So that's the underlying operator. And now we're going to smooth it by doing an average over a 2 by 2 block, which corresponds to convolution with this. And if you like, we can put in 1 over 4. I'm not concerned about those multipliers at the moment.

And what do we get? Well, hopefully, you remember something about convolution. So we flip one of these. Well, it doesn't do anything to this because it's symmetrical. And then we shift it over the other one. So the first position where we get something non-zero is when we have it in this position. And what do we get? We get minus 1.

So then I take this operator and move it one to the right. Now it's overlapping in two places. I get minus 1 times 1 is minus 1 plus 1 times 1. That's 0. Shift it over one more time. Now it's only overlapping over here. And I get plus 1. Then I move it one row down. And now I line it up over here. And I get 1 1 times minus 1 minus 1. That's minus 2.

I shift it one to the right. They all cancel out, 0. Shifted one to the right, I get 1 1 times 1 1 is plus 2. Shift it down one more, I get minus 1 0 plus 1. And there's Irwin Sobel's ex operator. Same in the y direction. So we can think of it as just a way of smoothing. And smoothing has the effect of reducing noise.

So unfortunately, it also has the effect of blurring things, making them-- so there's a trade-off obviously. We can reduce the edge gradient at the same time as we're cutting down on the noise. And so Irwin Sobel managed to avoid the 1/2 pixel offset problem by having two operators, each of which is 1/2 pixel offset. And when you convolve them together, you get 0, or 1, or minus 1 offset.

So that's the very front end of this operation. And they discuss-- they don't say, you should use this or that. They just give various formulas and the preferred one. So a lot of times, it will say the preferred implementation. And that's a tricky one because later on, if someone says, oh, but I didn't implement your preferred implementation, the lawyer will say, well, you're talking about an exemplary implementation, not what's claimed in the patent. What's claimed in the patent is in the claims.

What you say in the specification is just a way for somebody to implement this. And just because yours doesn't implement it exactly the same way is irrelevant. What's relevant is whether it violates infringes on the claims. And so a lot of times, there'll be verbiage about, in a preferred implementation, the gain is 10 or whatever. So if your gain is 11, that doesn't excuse you, because the important part is whether the claims cover it or not.

So that's the very front end. And then we're going to talk about the next step. So the next step is the conversion from Cartesian to polar coordinates for the brightness gradient. And we'll do this mostly next time. But for programming people, we'll call it atan2 of those two arguments. And the reason is that we don't want to have division by 0, which we can avoid by using the two argument version of atan.

So now we have a gradient magnitude. And we have a direction. And the next step is to quantize the direction. So what we're trying to find is a maximum in the direction across the edge. So suppose the edge is running this way, brightness E1, E2. Then we want to scan across the edge and look for a maximum, E0.

And the direction is given by E theta. So that's why we need E theta and e0. And unfortunately, we don't have points in all places in the plane, only on this grid. So we can only easily deal with quantized directions. So we got compass directions, eight possible directions. And we're looking for a maximum.

And now on a different grid, we would have different quantization. But we would still have a problem. So the center one is going to be G0. And then, so suppose this is our quantized gradient direction. And then we'll call the gradient here G plus and the gradient there G minus. And so we have those three values.

And clearly, the center pixel is an important one. It's on the edge but quantized to pixel positions. So it doesn't give us subpixel accuracy. So now, first of all, we keep only if-- so we step through the image. And we ignore nonmaxima. And now what we actually want is something that is asymmetrical.

So it could happen that G0 is equal to G plus or equal to G minus. And we could say, well, that means it's not a maximum. And just throw it out. But of course, that's wrong, because we have to keep one of those two points. But we don't want to keep both, because then we have two points on the edge that are put on the edge that are a pixel apart. So that's no good either.

So we have to have a tie breaker. So it's important that this condition is asymmetrical. We can make it asymmetrical the other way. That will also work. But we need to have a way of dealing with a case, where G0 is equal to G plus or equal to G minus. And then we're trying to find the peak. Now how does the curve go? Well, we don't know. But we can imagine various-- well, it's not a very good parabola.

But so we have-- here's our parabola. We take its derivative. And there's the peak. And in terms of these quantities here, if you go through and substitute, if you fit a, b, and c, then you get this formula. And well, here I've used x. Here I'm using S.

And we can show importantly that S computed this way is 1/2, is limited in magnitude to a 1/2. Why is that? Well, if it was, let's say, 3/4, that would mean that we'd be closer to this point. And then this should be the maximum. So you can actually show that can't happen. So that's its range of solutions, provided G0 is the maxim.

And you can understand what's going on here. So if G plus is the same as G minus, S is 0. We have a balanced parabola. We have this situation. And in this case, of course you would say, oh, that's it if these two are the same. And then it gets shifted the more there's a difference between the two sides.

And how much does it get shifted by? Well, we look at the second derivative, which is this difference here. Well, it's not the second derivative, but proportional to the second derivative. So we're looking at the average of G plus and G minus and comparing it to this point. And the larger that difference is, the higher the second derivative. So it just corresponds to-- b is the first derivative. a is the second.

So that's one way of getting the subpixel accuracy. And then the other way they mention is the little triangle model. And that just seems weird. But for some reason, there are circumstances where it's a pretty good model. So again, we have G0, G plus, G minus. Let's see. So [INAUDIBLE].

So again, with three measurements, we have just enough information to fit this model. The assumption of the model is that the slopes of the two lines are the same. And then we just need to vertical position in the horizontal position. And in this case, S comes out to be-- and again, that formula is in the patent. And it's pretty easy to derive.

So we'll talk about this patent some more. But there's some interesting technical issues aside from the patentees that we'll get to. For example, one reason an edge might not be a unit step edge is because of defocus. And so the question is, what is the shape of that curve? Because this whole business here, we're just making up stuff. We're saying, oh, maybe it's a parabola. Or maybe it's a triangle.

Well, can't we just figure out exactly what it is? Particularly in the case of defocus, we should be able to figure out just what-- obviously, it's not going to be a step edge anymore. It's going to be smeared out because it's blurred. It's out of focus. But what is that? So we'll talk about that and see how that affects our method of recovering the actual edge position.

And then we'll talk about some alternatives to what's in the patent. In particular, this quantization of a gradient direction is problematic, particularly on the square grid, where we've only got eight directions. And so we'll talk about other ways of proceeding that do not have that very cause quantization.

Like here, we're finding a maximum in that direction or in this direction. But if the gradient is actually in-between, it's not. Anyway, so there's more. And it seems like a very simple problem. But it shows how, if you want really good performance, there are a lot of details to figure out, like this business about, what's a good way to compute the derivatives?