PROFESSOR: So we are honored to have Professor Tomohiro Tachi here today visiting from the University of Tokyo. We first met in SIGGRAPH 2006, which was in Boston, big graphics conference, when he was a PhD student. And Tomohiro has very quickly become a star in the area of computational origami, and it's really exciting to see in particular his perspective coming from an architecture background and how that influences his work. He's going to talk about lots of exciting things in the context of architectural origami for our grand finale lecture. So please welcome Tomohiro.

[APPLAUSE]

PROFESSOR: Thank you very much, Eric. So I'm Tomohiro Tachi from the University of Tokyo, and I'm going to talk about architecture origami, which is architectural form design systems based in computational origami. So first, introduction.

I have been doing origami as a hobby, as well as my research. Well, it was before my research started that I began folding. These are examples of folding using a traditional way of box pleating or a 3D curved surface. This is pure origami, and there is also applied origami, which is applying origami for engineering purposes.

For example, we can use sheet folding for manufacturing a 3D surface or use it for increasing the structure's stiffness. These are examples of a formed 3D surface from a sheet of metal. We can use a dynamic property of origami for deployable structure, such as this is folding of solar panels used for spatial structures.

A good thing about origami is that unlike the truss structure or scissors structure, you can form basically a continuous surface, and that continuity is preserved all the way of the transformation. So this makes origami potentially useful for adaptive environment, which includes context customized design or personal design so that you can make some kind of origami structures that fit customized to your body or customised to the design context. And you can, of course, do fabrication oriented design so that you can efficiently use the material to build the 3D structure that you want.

What I'm proposing with the architectural origami, it's different from origami architecture. I have been using the word "origami architecture," which might mean the application of origami to design. So this is the direction of application. We start from some kind of shape or pattern and see how it is folded or see the behavior by simulating, or directly applying in physical world.

But that can result in very restricted design because origami is very constrained, and that would just result in just a copy of some known origami pattern to design. Or sometimes, there are lots of origami inspired designs that don't use any of the origami properties that might be useful for engineering purpose.

So what I am proposing with the word "architectural origami" is that origami theory for design and a design system that uses that. So the idea is in this direction. We want to get some kind of designed property, like how it behaves in 3D or in time, also, and we want to extract the characteristics of origami and obtain the solution from that required condition, and also design contexts that are given by the design purpose.

This is the outline. First is Origamizer. I'm going to talk about basically the software I've been developing, the systems. One is Origamizer. Some of you have attended Eric's class, so the theoretic side of the Origamizer altorithm, but I'm going to talk about the software itself and the algorithm that efficiently works.

So first, Origamizer. This is the software is available on my web page. It's freely available. It's about origami design. This is my oldest work as research. We had a tree method or circle river packing method developed by Meguro and Lang, which is using stick figure for representing the model and designed by packing circles and rivers. I think Jason has talked in the class.

So this is existing method for origami design, and what I wanted to do is make 3D instead of this kind of 1D figure. So this is what we can get from the circle river packing method, and with Origamizer or Freeform Origami, we can get this kind of 3D form. You can say that what you see is what you fold. In this case, what you see

is not what you fold, actually.

So I manually designed something like that. It's a laptop PC. Well, I think it's the most complicated one. So for example, you can see there is an RGB output here and USB outputs here, and this is the touch pad, and this is keyboard. Good thing about that is the function key is a little bit narrower than the normal key here.

Anyway, this is what I've been working on. Well, I think it was my hobby, but I was trying to make some kind of box pleating to make anything. And also, this is more three dimensional figure. The body is represented basically by blocks of cubes.

And I did more purely geometric things with concave vertex. This was very important for me. It's very easy to make a convex polyhedron with paper. Just wrapping paper will result in folding. But it's very hard to make concave vertex.

When I finished this model, I thought everything is possible, so I did a software to do this for me. The program is to realize arbitrarily given polyhedra surface with a developable surface by folding. The geometric constraints is, of course, developability so that it can be folded from a sheet of paper.

In this case, we forget about continuous motion from a sheet of paper to the resulting state. We believe that physically it exists, so it works. It can be applied in engineering sense for fabrication by folding and bending. We put arbitrary polyhedron and get the crease pattern, and by only folding this part, you get the folded state that's the same as this arbitrarily given polyhedron.

The idea is to use tuck. This is a given polyhedron, and this grey area is our tucks. By folding a tuck, it's hidden. It's flat folded and it's hidden behind a polyhedron's surface, like in this movie. When it's in developed state, it forms a plane with a surface polyhedron. This is good because we can make a negative curvature vertex, a concave or negative curvature vertex.

The basic algorithm is to start from making the problem into laying out the surface polygons onto a plane. By properly aligning, we can get the edge tucking molecule that folds the edge to edge and vertex tucking molecule which folds vertices to

vertex, and we can pack them and tesselate the surface with these elements. We can parameterize this configuration, and we can solve it by solving non-linear equation and inequality conditions. That's the basic idea.

What we have is geometric constraints, which are represented by equations. So these are the equations, where this one represents the total sum, sums up 2 pi, and this shows that this forms a closed loop for each vertex. We assign these conditions for each vertex, and we solve that by two step linear mapping, which is basically solving this equation and this equation.

Good thing about that is that it becomes linear, which means that you can pre-compute and you can go around the solution space. I will show you later the design system. I don't go into details, but there are several inequality conditions, which are for making crease pattern and for making it to fit into 3D configuration of the polyhedron and tuck property, which is assumed place of the tuck that's hidden.

So this is a system. You put the input here in this window, and you have the layout of the surface polygons, and then you get the crease pattern like that. It automatically generates the crease pattern, and here you see it's edited in real time. By the way, this is real time. The linear equations are represented by linear equations, so you can pre-compute the linear matrix and then you can calculate the configuration interactively. You can also do some kind of boundary editing. Do you have questions?

AUDIENCE: Yes. Does this maximize the amout of paper on the outside?

PROFESSOR: It's about maximizing the size of the model with respect to the paper size. It's not. Yes, you can, of course, implement some kind of optimization onto the program, but I keep it more free so that the software user can search within the solution space that satisfies these origamizing conditions.

So this is a series of results that are folded. This is how to fold an origami bunny. First thing you do is to get crease pattern using Origamizer, and then fold along the given crease pattern like this. Interesting thing about that is that it's changing the

lighting conditions. It starts from natural light and it becomes into artificial light. It takes about 10 hours or something. And it's done. And I have this one.

[APPLAUSE]

There are two existing models, not one, so it's very easy to reproduce. One thing that I'm doing with Eric is to prove that anything is possible. The software is a little bit approximation, or it doesn't work in some cases, so we are doing really make it possible. This is the part of the Origamizer, so if you have questions for Origamizer. Yes?

AUDIENCE: So for the input of the model that you can use, does it take [INAUDIBLE], the format?

AUDIENCE: It's polygons.

AUDIENCE: What's the file format that you have to import in in order to--

PROFESSOR: Format of 3D input. It's basically OBJ file that this particular software accepts, but basically, it's OK if it's a polyhedral surface with actually any topology because you can basically assign the boundary of paper onto it so that it always becomes a disk. Yes?

AUDIENCE: You mentioned that earlier on in the process, you need to properly lay out the polygons onto the plane. What constitutes proper layout?

PROFESSOR: Your question is, what is a proper layout? This is basically given by these conditions. In an intuitive way, if you lay out polygons, I want to fold this line onto this line with, actually, a single line, so you have to keep this symmetric against some line. So that gives a constraint.

This is the basic equations that you have to solve, and there are several others, such as the boundary should be convex because we don't want to end up in some kind of very complex boundary. If it's foldable form a convex paper, then it's foldable from a square, so that's the convexity of paper. No intersection, of course, between

polygons.

It's a little bit difficult, but when you place crease lines, then it might hit adjacent crease lines. That's pretty bad, so you have to avoid that. Also, you have to fit to the 3D surface. Sometimes a tuck that's coming inside is bad, so you have to adjust that, and in order to do that, you need a condition.

AUDIENCE: Did you find that you were moving around where the edges of the [INAUDIBLE] model, or can you just start anywhere and randomly come up with a solution?

PROFESSOR: It's how to solve?

AUDIENCE: Yeah. Does it get a lot harder as the polygons increase?

PROFESSOR: It's not a step by step approach to laying out one piece at a time, but it's more like solving all the equations all at the same time using big vector equation.

AUDIENCE: Maybe what he means is, is it numerically well conditioned?

PROFESSOR: No. That's why we need extra work for proving.

AUDIENCE: That it's always [INAUDIBLE].

PROFESSOR: That's a question. Well, it's not always. There are sometimes that produces some kind of over constrained part and free part.

AUDIENCE: Also, are the tucks under mechanical tension? Does it want to stay tucked or does it want to come apart?

PROFESSOR: Physical property of the tuck.

AUDIENCE: Yes.

PROFESSOR: Because of the crimping that's done to each of these vertices to fit the curvature, it stays in this form. You don't have any glues or crimps here. This is actually a good property of this design method. Yes?

AUDIENCE: To approximate a curve, you're breaking it down into smaller straight edge

segments. Is that useful or plannable as to what the lower bound is to how granular you make the curve approximation?

PROFESSOR: About the condition for the input. In this system, I don't give any kind of numerical upper boundary for the regularity of triangles or something. But I think it's good if we can do for anything, so that's why I'm working with Eric on proving.

AUDIENCE: Is there a Mac version of the program?

PROFESSOR: Mac version? No, it's still Windows version only. It's written with cross platform library, so sometime in future.

So let's move on to the next topic and software, which is Freeform Origami. So the objective of Freeform Origami is that we don't want to spend too much time to make this kind of 3D form, and also, we want some kind of conditions, not only developability. We want some condition that most origami has. This is flat foldability so that it folds flat, and also transformability or elastic properties. We want to use these good properties from origami while we want to make some kind of freeform surface.

So the approach here is start from existing origami models and then modify that to a different shape while keeping the origami conditions with direct, straightforward user interface. Here, I used triangular mesh for representing the origami model, and we represent the origami models by the vertex coordinates of the model. These variables are constrained by developability and flat foldability This is a very direct way of implementing origami deformation.

First thing we want to keep is developability, which is that you can fold from a sheet of paper. In engineering sense, it means that it can be manufactured from a sheet material, which is very nice, by folding and bending only. The condition is globally represented that there exists some isometric mapping to a plane, but if the surface is a topological disk, it can be represented by a local condition of every point.

So every point on the surface, the Gauss curvature is zero. Actually, we are thinking of a non-smooth surface, but we are thinking of a piecewise linear surface, which

allows variation. Smooth developer surface is only allowed to be in these forms, but we have lots of different form variations if we allow creases.

Even in this case, we have to think about Gauss area, and it's very easy to define Gauss area for a C2 version of surface by multiplying curvatures, but for polyhedra case, we use instead Gauss area, which is represented by this, which is 2 pi minus sum of angles around the vertex. It's a very simple way to express that it's folded from a plane. This sums up to 2 pi.

This is what you can use for developability. We want to also have flat foldability for the surface. This is applicable for compactly packaging a surface from 3D to 2D and from 2D to 3D. It's also represented by isometric condition, like in developability. And also, we have a little bit of layering condition, which is actually NP complete, which is hard, but for practical purposes, we can avoid that.

The first one is isometry. That can be similarly represented by angle condition, which is called Kawasaki's theorem, so that the alternating sum of each vertex is zero. And the layer ordering is NP complete, but we can use, for example, sufficient condition given by Kawasaki or empirical condition, which basically given for each local adjacent angle, so it's very easy to implement. And it works, so we forget about NP complete part.

And also, we can give several constraints. For example, this fold doesn't want to fold so that it forms a planar surface, or you can fix the point to a point in 3D space, or we can make some edge to be rigid. So we have these coordinates, and these are the variables to represent the configuration, and be assign developability, flat foldability, or other constraints.

This forms an under-determined system, which means that it gives you a multi-dimensional solution space. Within the solution space, we move. We transform the surface so that you will get the always valid solution. That's the method we are applying.

In order to solve that, we can use the Jacobian of the constraints and calculate it

numerically. So for each step, it's a given assumed transformation mold, and this gives you a valid transformation mold by using the generalized inverse or pseudo inverse. You can implement software like this.

The top is what happens with Freeform Origami. This is a crease pattern of the paper and this is the folded pattern when it's X-rayed. You can see that if you drag this point up, then all the crease pattern and the flat folded pattern changes at the same time. This is for comparison. This is kind of a simulation. In this case, the model cannot change the crease pattern, which makes a less flexible motion for origami.

So we want to do some mesh modification because if you change the crease pattern, then it will end up being degenerate triangles, so that we need to do some kind of edge collapse operation. However, in order to do that, we have several conditions. We have to keep Maekawa's theorem if the surface is flat foldable, and we use that for doing edge collapsing and mesh modification.

You see these patterns change, and these are getting very degenerate triangles that are removed by this operation. This is interesting because you will end up in different patterns. This part is similar to diamond pattern or Yoshimura pattern, and this part is kept the same as Miura-Ori pattern. Mesh modification enables to transform to produce more variations for pattern.

From now, I will show some examples of Freeform Origami. First one is starting from Miura-Ori. I think it's well known. This is actually very old, you can see from napkin folds. For paper models, I think you can see pictures from Bauhaus. Anyway, this is called Miura-Ori, and it gives you expansive motion, and also it can be compactly packaged and it's developable, of course.

So this is a variation of that, and this is another example. This model is called Melting Ice Cream. The idea is that using a variation of Miura-Ori, it forms a 3D surface that is irregular and asymmetric, while it can be folded flat so that you can roll it up, carry out. This is a model. This is another example. Here, I wanted to start from regular Miura-Ori and then to transform into a more free form surface.

You can also generalize Ron Resch pattern. This is one of Ron Resch's designed triangular tessellations. Good thing about this is that it forms a kind of composite surface when its folded. So it's composed with top surface and tucks inside, just like this bunny, but it's more flexible and you can fold from a sheet to the 3D form.

In order to design this, we assigned the condition that in 3D state, these three vertices form one vertex. That's extra constraints to enable generalization of Ron Resch pattern. So for example, you can get this kind of asymmetric polyhedral surface with this generalized pattern, or this form with this pattern. Each triangle has different shapes. A little bit of differentiated shape.

Or you can just apply to some regular triangular mesh and then get some nice crumpled paper like that. I think this is a good way for using this software. You can design crumpled paper. It's very hard to crumple paper in real life. If you feel it's very difficult, then you can use this software to crumple up paper.

This is a generalized version of Yoshimura pattern or diamond pattern that you can use for a kind of shell-like structure. This is another pattern called waterbomb pattern. It's supposed to be playing the puffer fish motion by [INAUDIBLE].

Anyway, this type of pattern is called waterbomb pattern. I think the oldest known is by Shuzo Fujimoto. It's kind of a flexible pattern. Because it's flat foldable, it can be used for deployable structure, and because of the elastic property that the folding gives, you can use for textured material or cloth folding.

This is an example of deformation. This is the normal waterbomb pattern. This is based on triangles. This one forms more planar surface when it's in 3D, and this is some generalization. This is the folded form. I think it's a little bit too small to show. This is the pattern, and you fold it into a kind of a butterfly shape, and it forms a hyperbolic paraboloidal shape, but it's a subtle shape. This is another way to make a [INAUDIBLE] in paper, and this exists.

This was Freeform Origami. Do you have any questions about Freeform Origami?

AUDIENCE: Why can't you solve directly for the final shape that you want?

PROFESSOR: The question is, why do I not search for the 3D shape instead of gradually changing? This is because the constraints are non-linear, and that makes it possible that the solution doesn't exist. If you do it continuously, then it is always true that you get the right answer.

This is a demonstration. This is good because basically, you can interact with the geometry of the origami to find some new results. It's more similar to when you design with paper, like when you interact with real material, but this gives more flexible change of material than simulation. I wanted to make that kind of interaction possible, so this moves like that, and that changes the pattern. That's the reason I do it in this way, solve the equation by deformation. Any questions?

AUDIENCE: I'm just curious, how long did it take you to develop both this and the Origamizer?

PROFESSOR: The time for developing the software. Well, it's always continuing. I'm always containing developing, so it's very hard to say. I don't know. I think basically it starts to work in half a year, and then polish a lot of times.

AUDIENCE: Did you want to show the demo one?

PROFESSOR: No. Not yet. So then we talk about the rigid origami. Question?

AUDIENCE: I might not understand it correctly, but in either of these programs, or is there a program that exists that allows the user to design dynamically like this curved crease origami?

PROFESSOR: So curved crease origami is quite hard. I have tried with this software, and it works for some cases, and it doesn't work for most of the cases because there are too much degrees of freedom. If you simulate curve folding by very thin quadrilateral strips, so that's a future work so that the software can work with curve folding. If it's a coarse approximation, then it works.

So rigid origami. This is the closest part of computational origami to architecture. So rigid origami is plates and hinges model for origami like this. So each rigid panel is

connected to adjacent panel with a rotational hinge, one axis hinge. The panels do not deform, and it produces a synchronized motion. So that's a model when you want to apply origami's dynamic features to some designs.

And this is a comparison, and this illustrates the energy that's caused by the distortion of facets. For example, this model falls to this state, but there is no path that gives you zero energy deformation. But this is rigid foldable, which keeps each of the faces not deforming. And it's useful for a self deployable structure or architectural structure that is very large. You can substitute panels with thick panels. I will talk in the last part. So like this.

So rigid origami. Here, we want to apply rigid origami to design purposes. In order to do that, we want to think about first to also generalize rigid foldability to different shapes, and also to generalize into cylinders and other topology, like compound structures. Before that, I would like to show example of rigid folding. This is the simulation of folding.

AUDIENCE: Could you play the lower left again?

PROFESSOR: So this is a triangulated model of a tesselation designed by Ray Schamp. This is a waterbomb tessellation. So first thing we want to do is to simulate folding to understand the geometry, the kinematics of rigid origami.

This can be represented. First easy representation is a truss model, which you basically use vertex coordinates as the variables and then constrain them with rigid bars that are connected to vertices. And this one is different representation that uses folding angle.

This is a more direct way, and I have a software called Rigid Origami Simulator which basically is based on this constraint and representation. The configuration is represented by folding angles, and then the folding angles are constrained at each vertex by this equation, which is a three by three matrix form, which actually gives nine equations, but only three of them are independent. We have three equations for each vertex. That will give also under-determined system so that you can

simulate folding using also generalized inverse so that you can search within the solution space by giving some pieces kind of a force that's asserted to each of the edges, and then this results in the deformation that is valid.

In generic case, because there are three constraints for each vertex and one variable for each edge, we have this degree of freedom. First way to design rigid origami is to use this information from Euler's polyhedral formula, we can say that any triangular mesh-- well, not any. It's a generic triangular mesh-- produces a degree of freedom, which is number of edges on the boundary minus 3.

So for example, in this [? hyper ?] triangular model, this gives one degree of freedom because the boundary edges are four. And we can generalize it to cases where there are holes also. With this idea, you can make hexagonal tripod shell, which is using six edged, hexagonal boundary, so that the degree of freedom is three, and then we pin joint three of the vertices, which gives nine constraints. Because of the rigid body degree of freedom, which is six, we have nine degrees of freedom of transformation, and then that's our constraint by nine, so it produces static structure. By changing the position of these points, you can get the different forms.

For example, this is the shape, triangular form. The position of these pods changes the overall shape, and it's static when it's fixed, it's all pin joint. This is an example. If you change the position of the leg, then you get the different shapes, 3D configuration like this. This is one way to build rigid origami structures.

This is quite obvious result. Theoretically, it's more interesting to think about the one that you cannot imagine the transformation, for example, quadrilateral mesh. So for quadrilateral mesh, each vertex has also three constraints so that this itself forms a one DOF structure.

Think about when you form a mesh with this, like Miura-Ori. Basically, if you define this folding angle, then all the folding angles here are defined. If you have another degree for vertex, then everything is here defined, and you want to have another vertex here that is defined by this folding angle. Then you will have a problem here

because this folding angle and this folding angle contradict. In general, it contradicts so that you cannot make an array of folding by quadrilateral mesh.

However, as you can see from this figure, there exist examples like Miura-Ori that give one degree of freedom motion. This is actually a very interesting thing happening. And also, this is great because all the constraints are redundant. You can remove this part, so you can put a hole in the center, but still, it results in the same motion. You can design more freely with this kind of robust, redundant constraint structure.

What I want to do is to generalize this to a freeform. We start from Miura-Ori and also start from this pattern, which is not a developable surface, but also gives one degree of freedom motion with redundant constraints in its quadrilateral mesh design. These can be generalized to rigid, non-symmetric forms.

Miura-Ori is something like that, and we can extract the property of rigid folding by looking at these vertex. If it's a flat foldable vertex, it's a degree four vertex with flat foldability, and this gives a folding motion where the opposite folding angle are the same and these are the same, and also, that the folding angle here and here are related. If you look at the tangent of half of the angle, then it's linearly related.

This is very useful because if you find one configuration that works, then you will have continuous folding that works. In this way, you can make sufficient condition for making a quadrilateral mesh rigid foldable. This is easy to get because it's only the finite folding motion. So the continuous folding motion from a sheet to the folded state is guaranteed by one 3D configuration that satisfies these conditions.

In order to get one configuration that satisfies this configuration, we can use the Freeform Origami, so it's very easy for us to do that. This is an example of rigid origami that's using redundant constraints. It produces one degree of freedom motion and it's very light because this part and this part counterbalance the gravity. This is another example. Quadrilateral mesh can be a rigid folding motion.

You can apply to something like curve folding. It's rigid foldable curve folding. This is

kind of contradictory, but we can rationalize curve folding into quadrilateral mesh. Then you can produce one degree of freedom motion like that. Another example with thickness is here. It's interesting that it's one degree of freedom motion so that every motion is coded in the pattern here.

Another example is using Eggbox pattern. This actually has the same property as the Miura-Ori vertex or flat foldable degree for vertex, so that you can use the same idea to produce rigid foldable variations. You can actually combine with origami structure if you define this mountain fold, valley fold, and also add complementary mountain fold and complementary valley fold, defined here.

If you use that kind of extension of the origami, then you will get design variations that combine origami. This part is origami vertex but this is not. This can have positive Gaussian curvature while this part is zero Gaussian curvature.

The interesting point is that you can develop the flat folded state or actually, in this case, it's two flat folded states because in this case, it's not totally developed, but complementary fold lines are folded. So that's why I call it bidirectionally flat foldable planar quadrilateral mesh, and that can be rigidity foldable. This is an example.

This was about a disk, and we can develop into a cylindrical structure. And cylinder is not trivial because, for example, this is known origami pattern, but this doesn't transform to this pattern because this is has a different number or edges. You cannot produce this kind of motion in rigid folding mechanism. You can see what's the problem by looking at this kind of disk surface that is rigidly foldable, and it forms a cylinder at one state but it doesn't produce in a continuous motion that fix this loop. But there exists.

So the idea is to mirror reflect one part of Miura-Ori vertex. Actually, you can see from the origami model by Thoki Yenn. It's called Flip Flop. It's a very interesting model. You can develop more generalized origami cylinders like these. I think this is rigid foldable cylinders.

Since it's using quadrilateral panels, it similarly produces one degree of freedom

motion. And you can see, for example, like that. This is the motion produced. Any part produces the whole motion.

And also, you can make some kind of design variations. This is using grasshopper on rhinoceros. This is the given shape, the section, and from the section, you can make a rigid foldable cylinder and also composite structures like this. So we can make composite structures like this, so it's great that it fills the space. It tessellates while preserving the one degree of freedom motion.

This is an implementation with thick panels. Think I have got the cylindrical tessellation model. This is a tessellated model, which moves like that. It's kind of a volume that flattens into two states, and it's good that it's rigidly foldable and it's one degree of freedom mechanism.

I want to generalize more. This is symmetric. There exists one axis that repeats the unit structure. I want to make it more general. I like the asymmetry than symmetry. I think you do also. But you have to think about the conditions around a hole, and actually, this is a difficult condition to be solved then the disk version.

However, you can do that by fixing the first loop, then you know that this part is rigid foldable, so you can continue deformation from there. This is kind of sufficient condition, but useful for designing something like this. It's more free form. For example, this is also one degree of freedom rigid foldable cylinder. This is an example folded. It folds like that.

This is also an example. This forms a torus when it's unfolded. Or this type of structure. So it's also cylindrical. And for kind of architectural image.

This is a rigid origami for cylinder. And how to implement rigid origami to real model, because in architecture design or any other designs, we want some kind of finite thickness model. There's no ideal origami. Well, probably Origamido can produce. Anyway, we want some thick panels and rotating hinges to produce rigid origami.

This is the typical way to do that. So you put the hinge on the valley side. The main problem of this one is that each vertex, there are several fold lines that are joining at

the vertex, but this will no longer be concurrent. But rigid origami mechanism assumes that the fold lines are concurrent, which means that it includes translation constraints and it increases the number of constraints for each vertex.

So previously, it was three constraints there, but it becomes six constraints because it includes transformation, which is very bad, but there is asymmetric vertex that allows that. So using asymmetry, you can make something like that. Also, there is a way that you can slide the hinge like that, but there is a problem in the case like this. The problem of sliding is in global accumulation of errors.

So what I do is instead of shifting hinge, you can trim the volume of the vertex valley side of the model. This is actually very easy. You assume that this folds to pi minus delta, and you can just remove this part according to this maximum folding angle. And you can define this point by offsetting the edges, which is basically calculating with a straight skeleton. This also can be applied to constant thickness panels, which is more easy to be manufactured, only with three axis milling machine.

This is a thick panel, thick rigid origami using this method. This is an example using a constant thickness model where no error is accumulated, like this model, a slideable hinge. This is implemented by using grasshopper again. Grasshopper is a good tool to implement something that does not require iterative optimization calculation, but it's very good for forward calculation of geometry. That calculates the pattern, and then you can lay out the pattern.

This is a cloth, and we put the thick panels on both sides of the cloth to make a rigid foldable structure like this. In the center, there is a cloth that represents the ideal origami. This is what happens.

I would like to give an example of rigid origami design for architecture. So this example is that we are given existing building openings, and we want to connect these openings temporarily. I want that to be compactly folded to fit to this facade, but because this is quite large like that, you cannot make from a flexible material. That will be a problem if you make it as a structure.

So in order to solve that kind of problem, we can use freeform origami to design to fit the condition. So these red lines are the openings of two buildings that are actually not parallel, and the sizes are different. But still, you can connect it on freefrom origami, and also, you can keep the boundary to fit on the ground, and then you can get these form variations and you can choose one of them and calculate the paneling patterns. This is a rendering representation, but basically, you can make this kind of structure that can connect to buildings. Thank you very much.

[APPLAUSE]

This was more hand craft thing. I did this with Dukes there, and we wanted to test the metal folding, whether curved folding is useful for manufacturing things. I think that's true, but this model required a lot of hammering and it wasn't so easy. I think you can have a way to make it more easily constructed. So this is a table just by folding a sheet of metal or chair.

This one? So this one is a combination of four papers. In this case, I did not restrict to be foldable from one sheet of paper.

Well, for design, I do not do that. You can only find it by simulating and see the collision. Basically, there will be no local collision for this quadrilateral mesh, but there might be some global collisions where this and this part are colliding.

AUDIENCE: For example, for Ron Resch pattern.

PROFESSOR: Ron Resch pattern. I only do a local collision test, which is between facets that are adjacent, which still works for that kind of good model. I think I will show all of the things here. Like that.

PROFESSOR: Any other questions? All right. Thanks again.

PROFESSOR: Thank you very much.

[APPLAUSE]