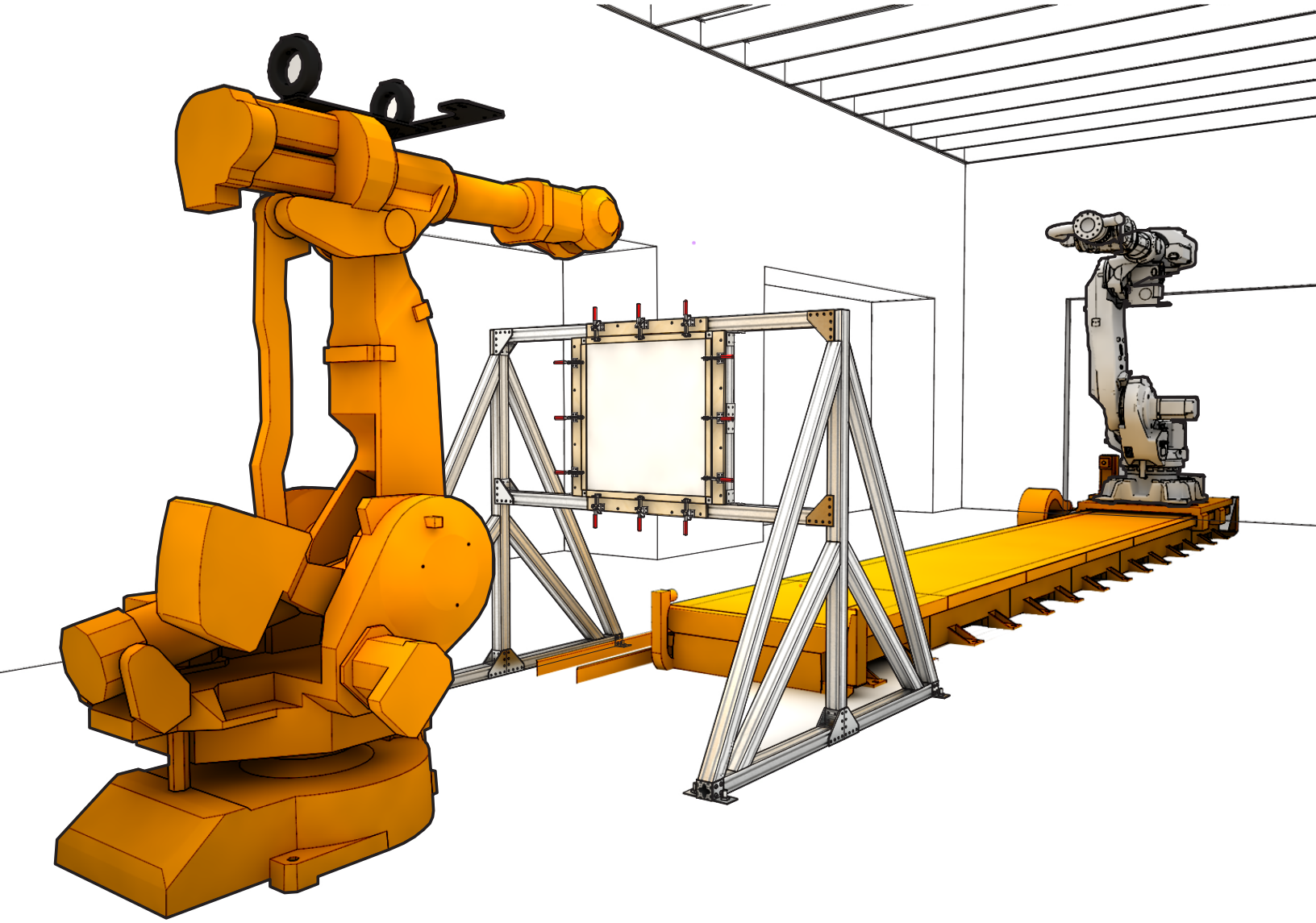
Embodied Computation:

Exploring Roboforming for the Mass-Customization of Architectural Components

Advisors: Jeremy Ficca, Josh Bard, David Kosbie

Alex J Fischer.com/thesis



What I will be presenting:

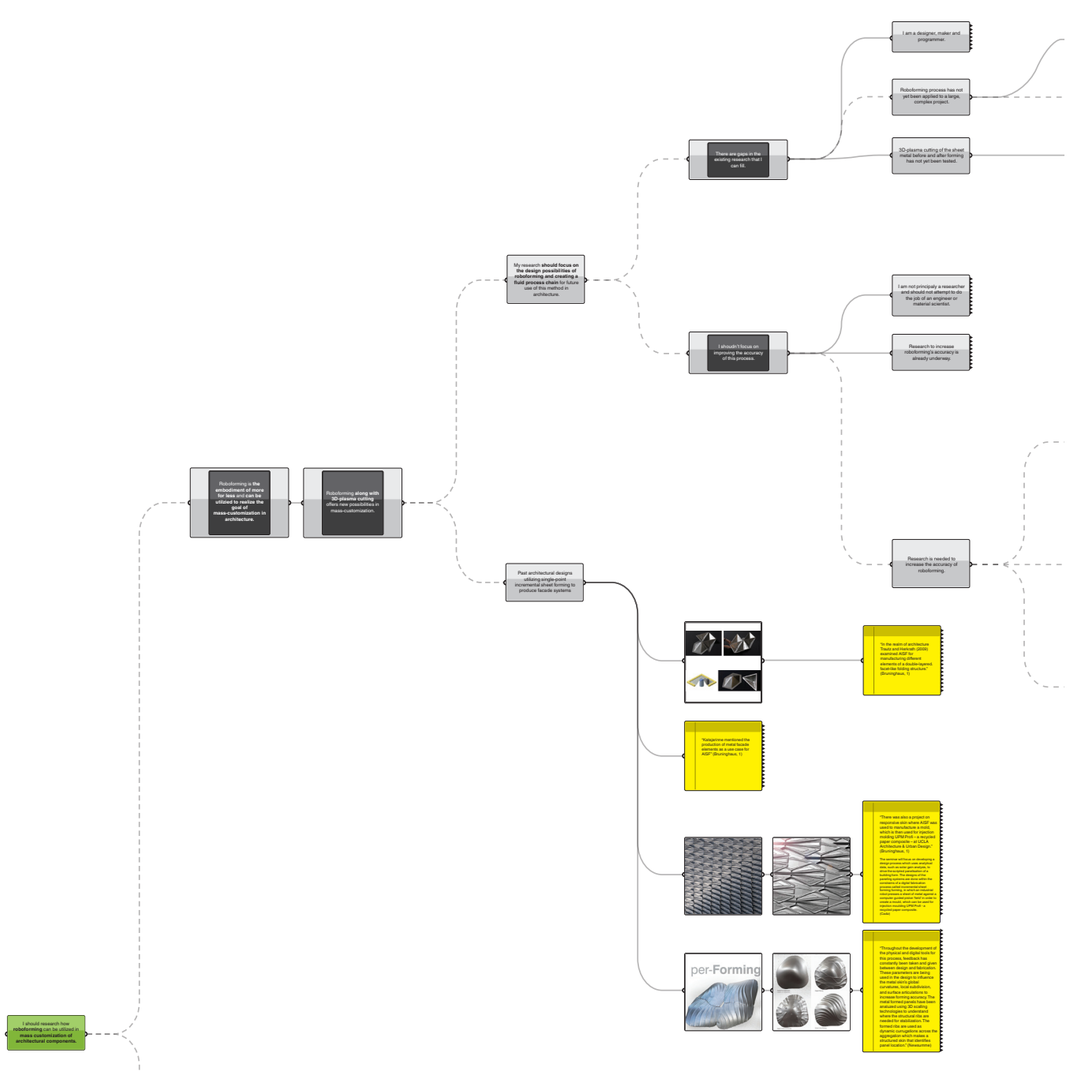
- Conclusions from Theory research
- Conclusions from Roboforming reseach
- Physical and Computational Tools
- Precedents of designs utilizing sheet forming
- Experiments I plan on carrying out
- Speculation on final design output

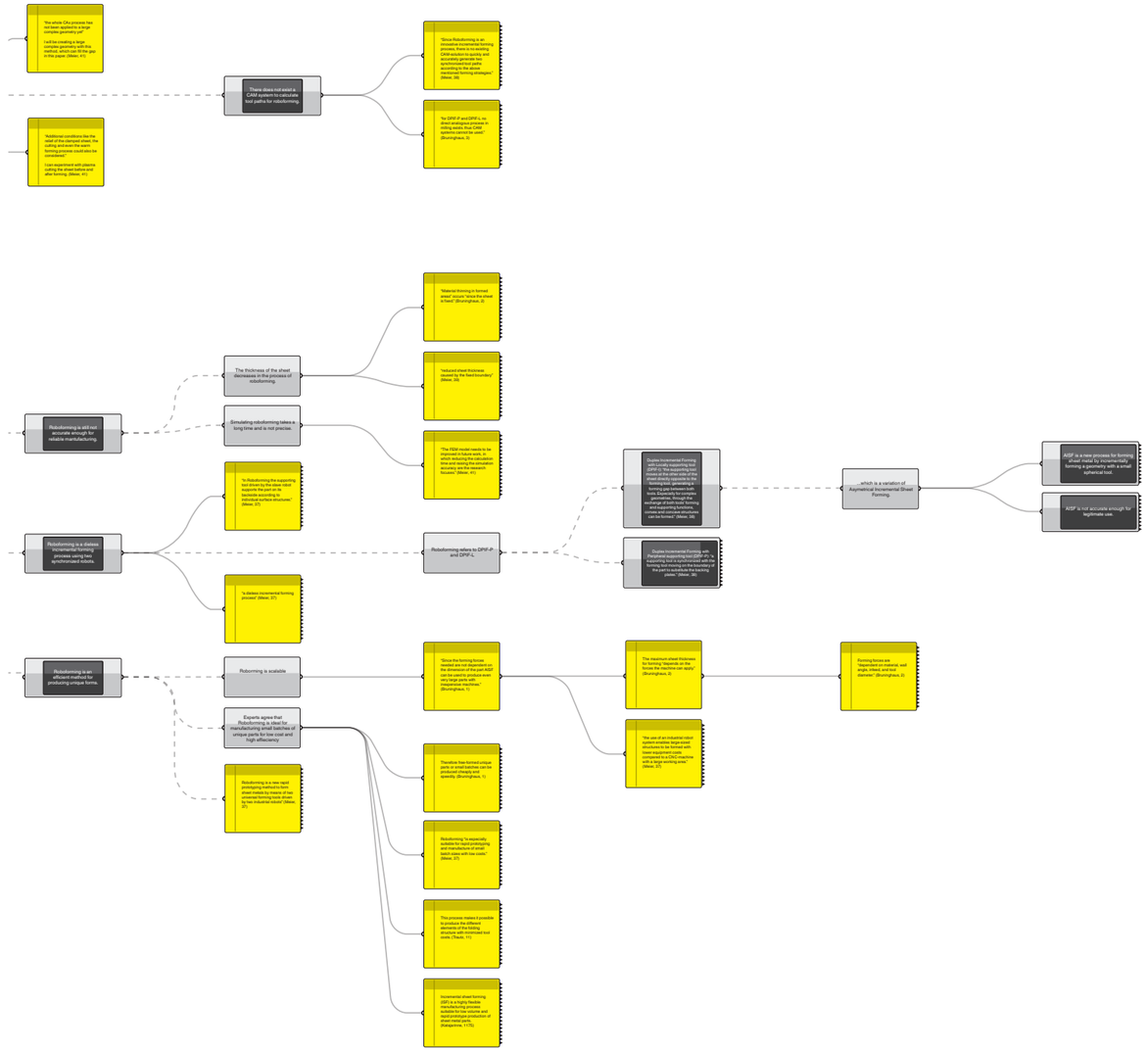
What I hope to gain from this review:

- Ideas for experiments
- Ideas on workflow
- Ideas for design output

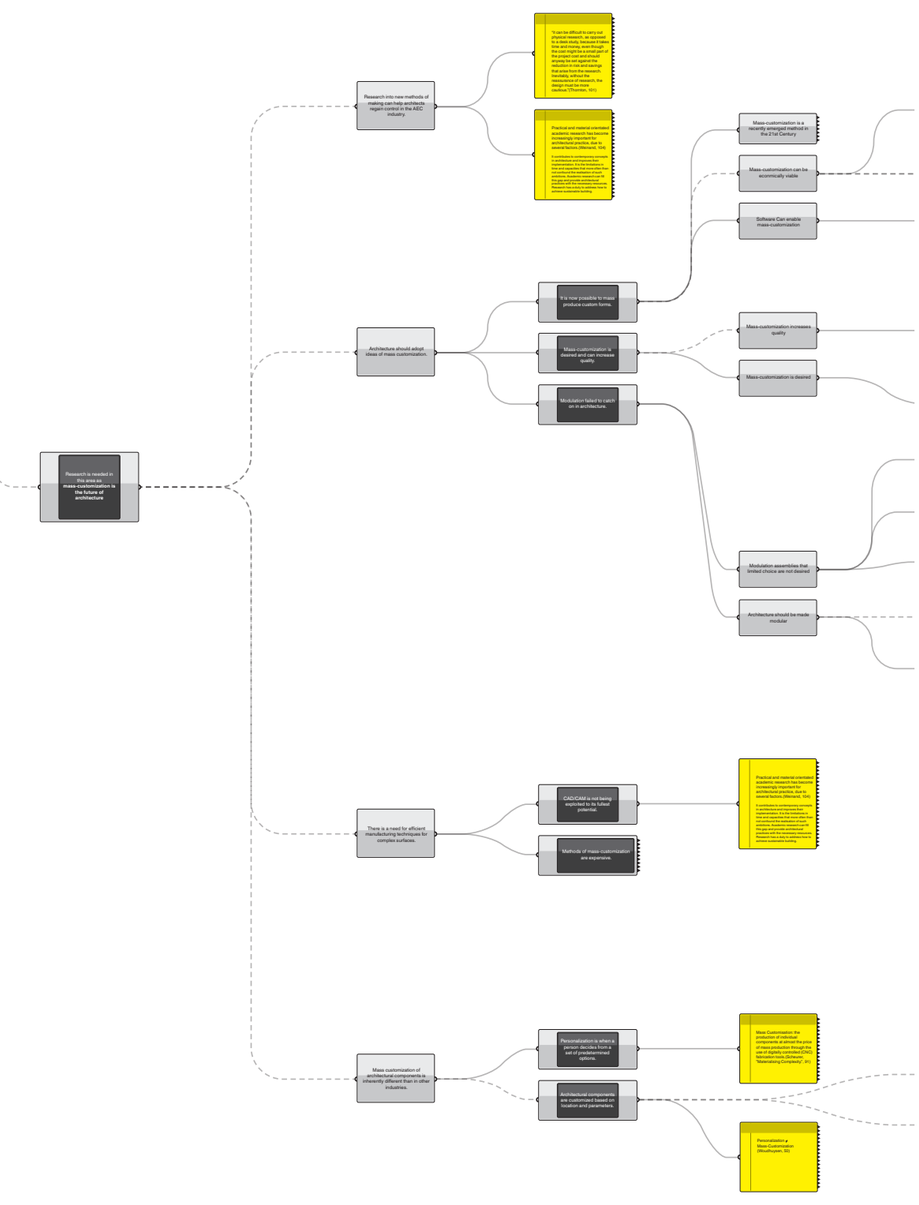
Introduction

Argument Map - Part 1/2 - Roboforming





Argument Map - Part 2/2 - Modulation and Mass-Customization



What I learned from my research:

- Designs stemming from material understanding and hands-on making will be more informed.
- There is a demand for mass-customized. architectural elements
- Roboforming is an efficient method for fabricating small batch sizes of unique parts.
- There does not exist a software for Roboforming.

Conclusion

- Design and Implement a Process Chain for Robo-forming at Carnegie Mellon's Dfab Lab.
- Utilize this Process Chain to experiment and learn more about Roboforming.
- Output an best-use example of an application of Roboforming in an architectural context.

Thesis Roadmap

Research mass-customization, material computation, robotic fabrication, and designers as makers.

Write a thesis statement conveying the conclusions from this research.

Research the parameters of Roboforming.

Draw diagrams of the parameters and side-effects to understand and convey the research.

Design scripts based on these diagrams to aide in the determination of robot targets.

Design and build a frame to hold the sheet metal between the robots

Explore the process of Roboforming sheet metal using these scripts and frame.

Reflect upon the processes involved and resulting artefacts.

Hypothesize on how these processes lend themselves to the creation of architectural components.

Demonstrate the potential by fabricating an aggregate of Roboformed components.

Thesis Statement

Embodied Computation is an investigation of Roboforming as it applies to design, fabrication, and architecture. Roboforming is a new rapid prototyping technique for forming sheet metal efficiently that allows the fabrication of complex unique surface geometries without the need for expensive dies by utilizing two industrial robots. (Bruninghaus, 1) There is sigificant potential in this forming process in the context of architecture which at its most base level can be applied to ideas of mass-customization and at its most abstract, a rethinking of how architecture is designed and concieved of. Mass-customization offers production of individual components at almost the price of mass production. (Scheurer, "Materialising Complexity.", 91) Roboforming, which follows this paradigm, is still largely unused in other manufacturing industries since it is not appropriate for mass-production in large quantities. (Meier, 37) However, I argue this trait makes Roboforming perfect for the production of architectural components, which are custom to their site.
This thesis' role in the advancement of Roboforming is to fill gaps in the research of engineers and material scientists, who admit there is no existing software to easily output tool-paths for Roboforming and thus, Roboforming "has not been applied to a large complex geometry yet." (Meier, 4) In addition, 3D plasma-cutting of Roboformed parts has not been explored, and could offer new possibilities. (Meier, 4)
This project will be realized through a feedback loop, as Menges describes or a Persistent Model as Ayres of sixteen*(makers) describes, of computational and physical tool-making, robotic fabrication, and analysis. This reseach differentiates itself from the work of engineers and material scientists, who mainly study methods to increase the accuracy of Roboforming, by focusing on issues of customized design and haptic responses that a formed part can generate in the context of architecture and at an architectural scale. Rather than seek to decrease deviations in the formed part, this project embraces the inaccuracies and side-effects and incorporates them in the design. The uniqueness of a formed part due to the inherent inaccuracy of Roboforming can actually add value to it. The materiality and method of manufacture play a tacit role in the response by the user. (Ayres, 221) Achim Menges argues that architecture attains its relevance "through the articulation of material arrangements and structures," thus, "the way we conceptualize these material interventions- and particularly the technology that enables their construction - presents a fundamental aspect in how we (re)think architecture." (Menges, "Integral Formation and Materialisation", 198)

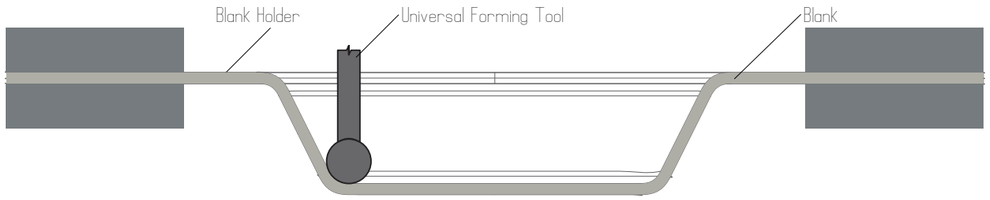
The interest in Roboforming stems from a broader scope of research that addresses ways to improve architecture's specificity and offer architects more choice and thereby more control. Roboforming should not be dismissed from serious investigation out of nonconformity to the current ideals of dimensional accuracy and fully predetermined outcomes before a thorough evaluation of its potentials and implications. (Ayres, 222)

My background in robotic fabrication, architecture, and computational design will enable me to design a process chain for Roboforming which includes: constructing the blank holder, writing scripts to output tool-paths based on geometrical input, synchronizing two 6-axis industrial robots, and outputting an example of what Roboforming in capable in relation to customized architectural components.

A more detailed examination of Roboforming and its parameters and side-effects will provide useful insight into how the process works and lead to a richer, more informed design.

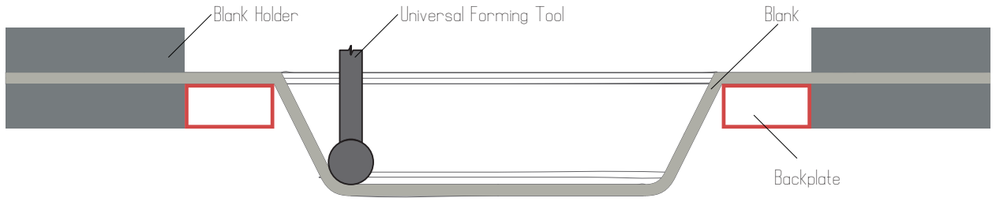
Roboforming Parameters

Types of Incremental Forming



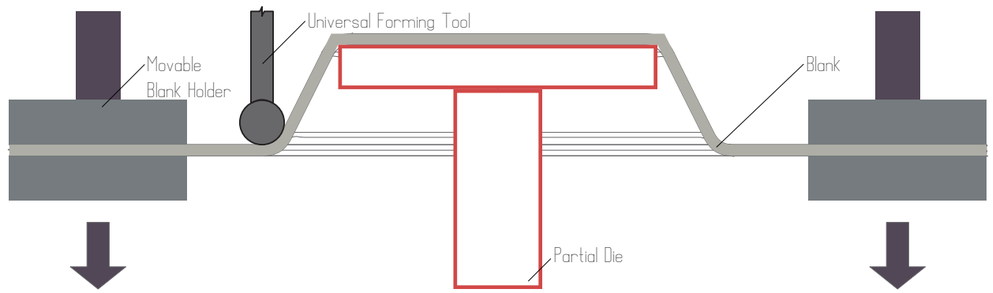
Asymmetric Incremental Sheet Forming (AISF):

A flat sheet of metal or plastic, known as a blank, is secured in a blank holder. A universal forming tool with a spherical head, follows the contours of the geometry to be formed, causing the sheet to be deformed along the tool-path. This is an efficient method, but causes inaccuracies and deviations from the desired geometry due in large part to the springback of the sheet.



Single Point Incremental Forming (SPIF):

A variation of AISF in which a backplate with the outlines of the desired geometry are fixed to the sheet on the opposite side of the tool-head. The backplate provides leverage at the initial crease of the form, reducing some of the deviations caused by springback. However, this method increases accuracy mostly at the periphery of the shape.



Two Point Incremental Forming (TPIF) with Partial Die:

A specialized tool or mold, known as a die, is located on the opposite side of the sheet from the forming tool. In this case a partial die is used, meaning it is not the exact shape of the desired geometry. Either the blank or the partial die is movable and synchronously forced in the opposite direction as the infeed direction of the tool. Inversely from SPIF, this method increases the accuracy at the apex of the form.

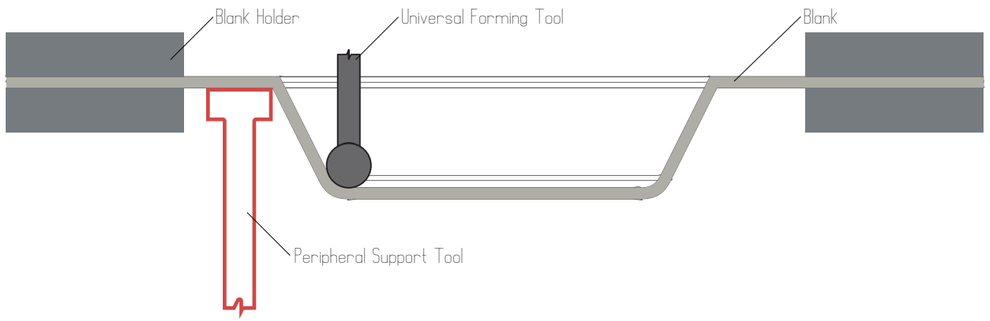
Two Point Incremental Forming (TPIF) with Complete Die:

A specialized tool or mold, known as a die, is located on the opposite side of the sheet from the robot, allowing the metal to be formed according to the die's shape. This method enables higher geometric accuracy but at a higher cost and longer manufacturing time for each part.

Sources:

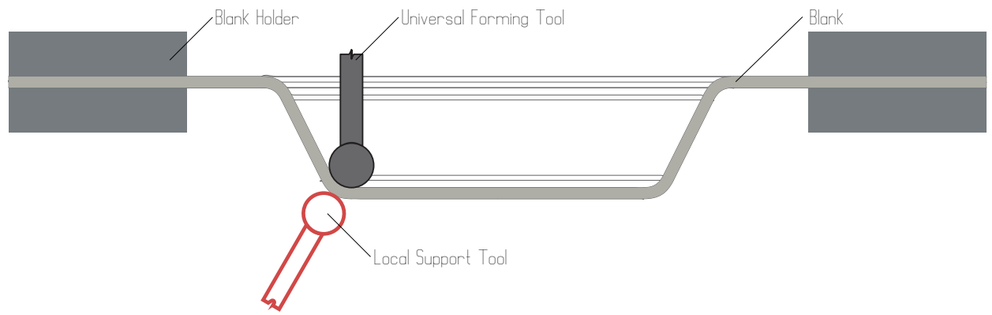
Bruninghaus, 1
Meier, 38-39

Types of Roboforming



Duplex Incremental Forming with Peripheral Supporting Tool (DPIF-P):

Two industrial robots are placed on either side of the blank sheet, secured in a sturdy frame. The 'master' robot holds the forming tool and the 'slave' robot holds a support tool. The master robot pushes incrementally on the sheet, forming the sheet in the shape of the tool-path. The slave robot moves the support tool along the boundary of the part, acting like a backplate, providing leverage on the opposite side of the sheet for the master robot to push against.



Duplex Incremental Forming with Locally Supporting Tool (DPIF-L):

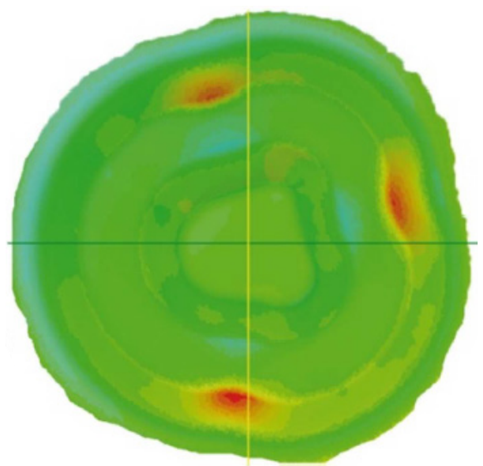
A blank sheet is secured between the two robots in a sturdy frame. Each robot is with a universal forming tool. The 'master' robot is supported by the 'slave' robot. The master robot pushes incrementally on the sheet, forming the sheet in the shape of the tool-path. The slave robot's tool follows directly opposite the forming tool, creating a forming gap between the tools. By interchanging the master and slave roles of the robots, concave and convex forms can shaped within the same part.

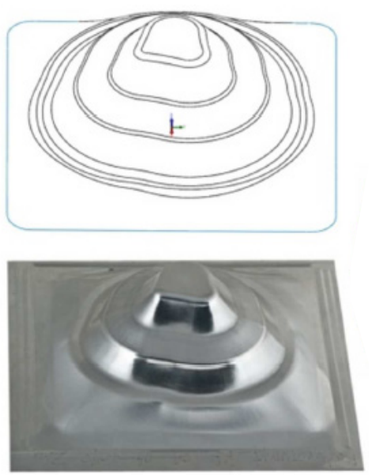
Sources:

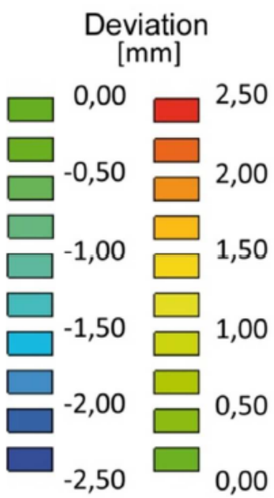
Bruninghaus, 1
Meier, 38-39

Advantages and Disadvantages of DPIF-L vs DPIF-P

DPIF-P







DPIF-L

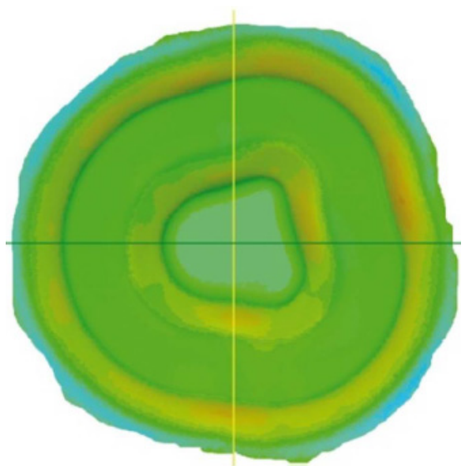


Image credit: Kreimeier, CAM, 895

Advantages of DPIF-P

- Smaller deviation in areas of greater wall angle since the peripheral support acts as a backing plate against which the forming tool can push.
- Smaller elastic deformations occur at the boundary of the geometry due to the support at the boundary.
- The side not directly acted upon by the forming tool will have a matte finish.

Disadvantages of DPIF-P

- Larger deviaions in concave areas.
- Non-uniform deviations.
- The two sides will have different finish qualities.

Conclusions

Appropriate for mostly symmetrical concave surfaces.

Advantages of DPIF-L

- Overall uniform deviation.
- Smaller deviations in concave areas due to the ability of the support forming tool to form these areas..
- Both sides have a glossy finish.

Disadvantages of DPIF-L

- Greater deviations in areas of greater wall angle due to the increased force needed when compared to flatter areas. This force causes the sheet to buckle inwards without the periphery support.
- Greater elastic deformations occur at the boundary of the geometry, which can lead to insufficient forming in other areas.

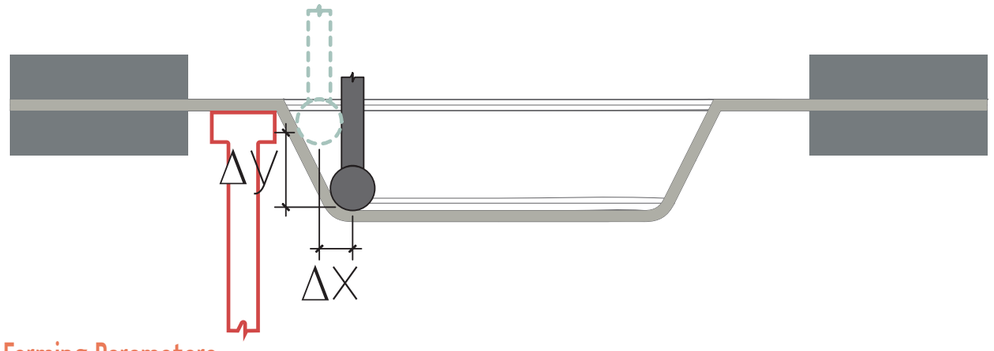
Conclusions

Appropriate for complex geometries with concave and convex areas.

Sources:

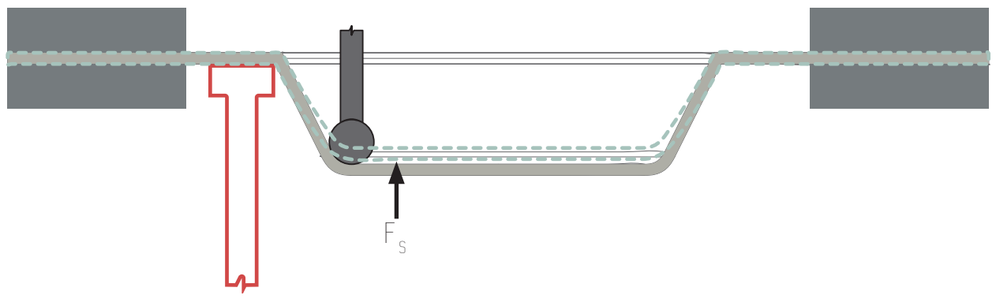
Kreimeier, CAM, 895

Causes of Deviations



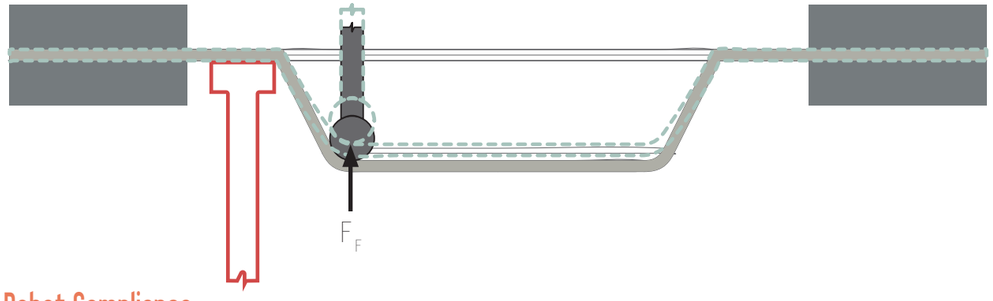
Forming Parameters

Material, forming geometry, infeed, velocity, stepdown, stepover, and tool-path type (contoured, stepped, or helical) affect the accuracy of the part.



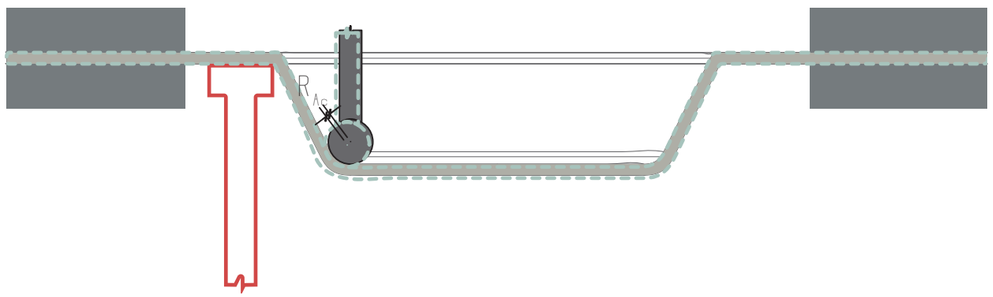
Springback

Due to the regression of the elastic forming ratio, the formed sheet will rest in a state somewhere between the desired geometry and the original flat sheet. This springback force, F_S, is inherient in the thickness of the material.



Robot Compliance

Forming forces, F_F, acting on the tool cause the robot to deflect from its predetermined path. If a robots detects enough force against its tool-head, it will pull-back to reduce the forces.



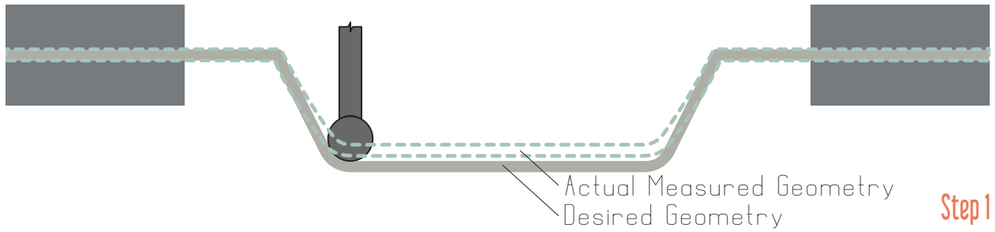
Positioning Accuracy

Due to the 6 joints of the robot the tool-tip does not always end up exactly where it was planned. The domain of the positioning accuracy, $R_{Ac.}$ can be as much as several tenths of a millimeter. This is determined by the resolution of the kinematic solvers and the accuracy of the robot's many joints.

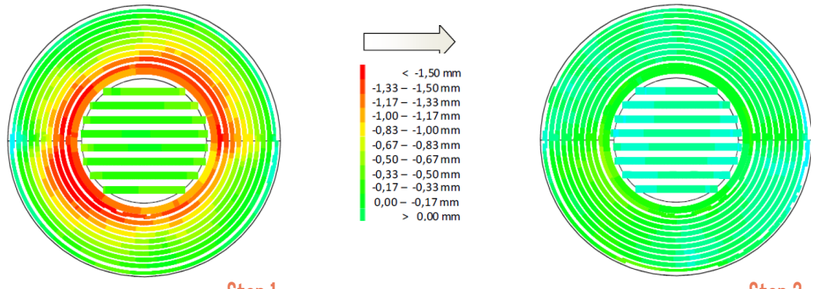
Sources:

Meier, 161

Methods for Increasing Part Accuracy



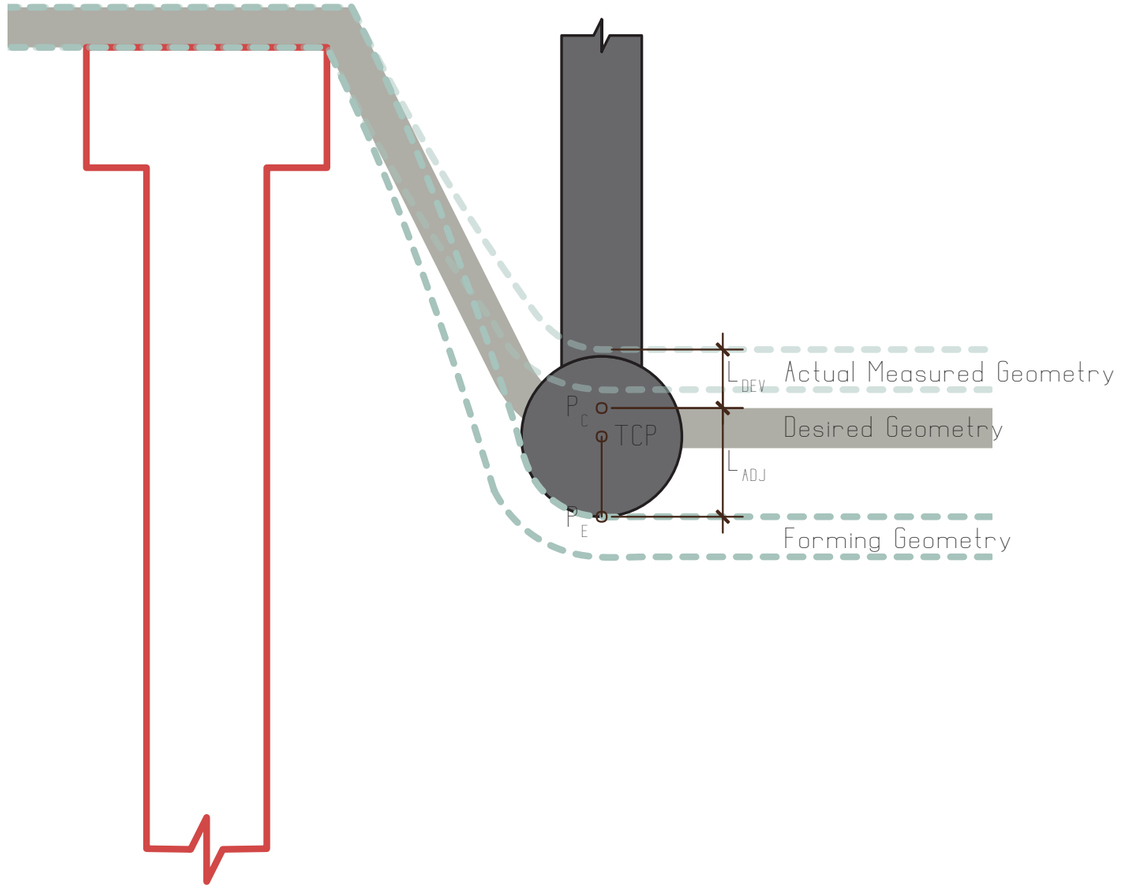
Deviations in normal direction of a truncated cone



(Meier, Accuracy, 7)

Multiple Forming

To increase the accuracy of the final formed geometry, run the forming process twice. This is effective because on the first round, the forming forces can get very high, depending on the depth of the part. These high forming forces and robot compliance cause the robot to pull back and ultimately the tool tip ends up not as far as described in the program code. Once the sheet has been formed, the second forming process will have significantly lower stresses since the sheet is already deformed close to the desired result. This allows the robot to follow it's input path more accuractly. The metal is also hardened in the first forming process due to strain hardening, so the metal is less able to bow and avoid the forming tool.



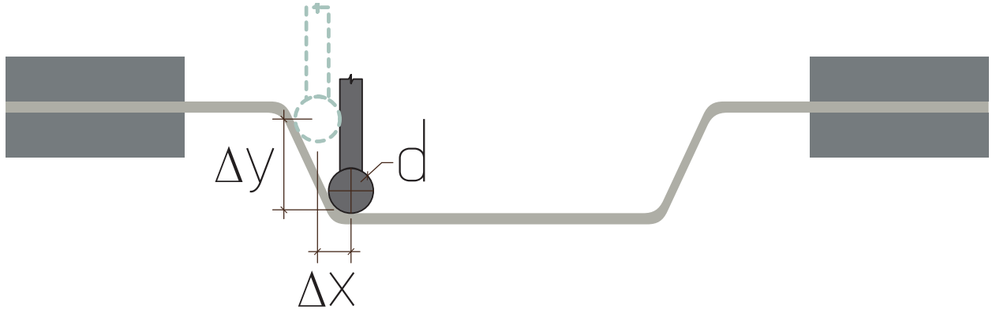
Adjustment Vector

This methods requires a feedback loop between sensors on the robot and tool-generation software. This could be done with finite element simulations of the forming process to predict deviations, but this is a time-consuming process, to the point where it is not viable with current software and hardware capabilities. Instead, a process of forming, scanning, adjusting the tool-path based on deviations, and forming again must be used. Rather than regerated the CAD geometry, it makes more sense to apply a spicific adjustment vector to each tool-path point. Ultimately, this means forming the part further out than the CAD geometry to accont for springback.

Sources:

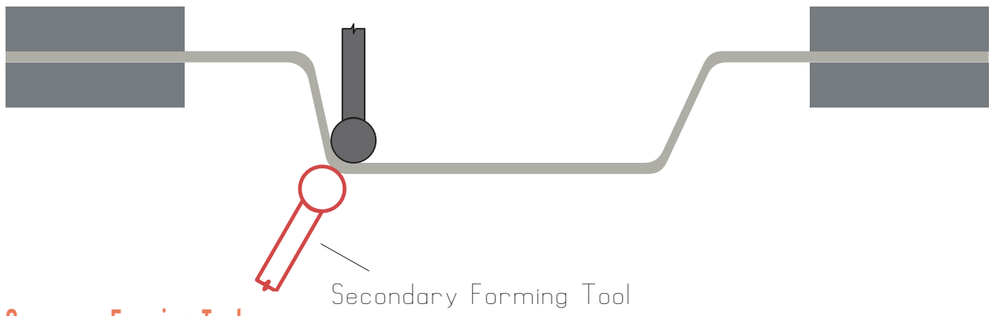
Meier, Accuracy, 1-7

Maximizing Surface Quality



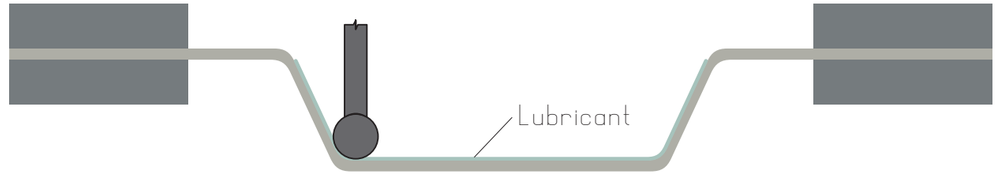
Forming Parameters

The appearance and roughness of both sides of the surface is dependent on the tool the forming tool's diameter, d, stepdown, y, and stepover, x. There is a direct relationship between the the tool diameter and the smoothness of the surface. In addition, there is an inverse relationship between the infeed parameters, x and y, to the surface smoothness.



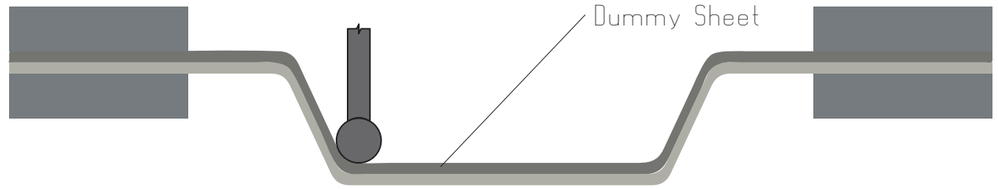
Seconary Forming Tool

In AISF, the side of the surface which is acted upon directly by the forming tool has a glossy appearance. This side also has grooves and stratches due to the movement of the tool. The other side, which is not in direct contact with the forming tool, has a bumpy matte finish with an 'orange peel' effect caused by the stretching of the sheet. So in DPIF-L, where there is a forming tool on each side, the result is both sides have a glossy finish.



Lubricant

Friction caused by the interaction between the forming tool and sheet is a main component of the surface quality. By applying lubricant such as tapping fluid, the friction forces are reduced and scratches are minimized.



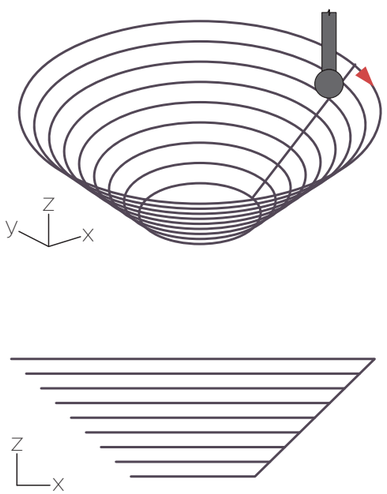
Dummy Sheet

Grooves caused by the tool can be prevented by layering an additional sheet on top of the sheet you wish to keep. This top sheet is known as a 'dummy sheet' and picks up all the tool marks while the lower sheet is formed indirectly by the deformation of the dummy sheet. As a result, the dummy is glossy on both sides and the lower sheet has a matte finish on both sides.

Sources:

Bruninghaus, 2
Meier, DPIF-L, 327

Tool-Path Types



Contoured

The tool-head follows the contours of the geometry. As it moves to the next level on the z axis, it moves diagonally in the xz axes. This creates a line where the tool moved from level to level.

Alternating

The tool-head follows the contours of the geometry. The direction of each contour alternates. As it moves to the next level on the z axis, it moves diagonally in the xz axes. This creates a line where the tool moved from level to level.

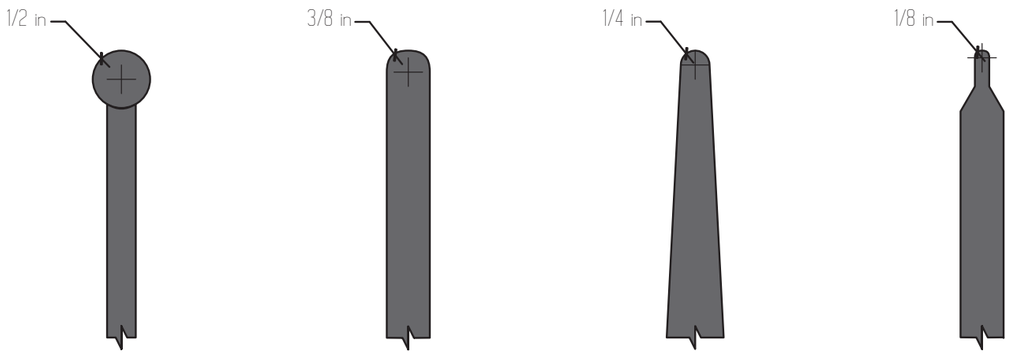
Stepped

The tool-head follows the contours of the geometry, but when it steps to the next level, it moves in the xy, then z axes, removing the line created by the tool as it moves from level to level.

Helical

The tool-head follows a continuous path outlining the shape of the geometry. The tool is always moving in the x, y, and z axes, creating a helix pattern in the formed part.

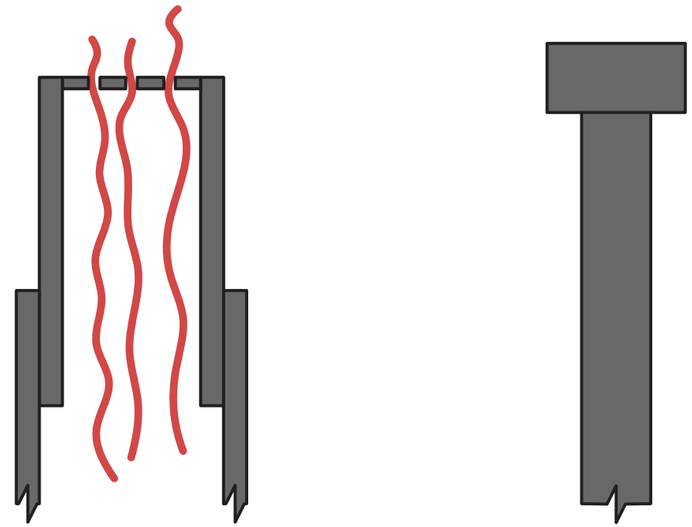
Tool Options



Dapper Punch | Straight | Tapered | Tipped

UNIVERSAL FORMING TOOLS

A small, mostly spherical, tool is attached to the master robot. This tool can come in many forms. A **dapper punch** is a ball-ended tool used for stamping half-spheres in metal, but can be repurposed for Roboforming. However, since dapper punches are not made for uses where forces are applied laterally, the tool can fail and bend as it is forming. A **straight** profiled tool is stronger, since it does not have a thin joint where the ball connects to the shaft as with the dapper punch. The **tapered** variation is even stronger and allows for smaller radii since the shaft is not the same width as the head. The **tipped** tool is ideal for small radius tools as it provides a small tip with a wide enough shaft to prevent excessive bending.



Heat | Support

SECONDARY TOOLS

On the slave robot, opposite the forming tool, there are two common tools that are used. A **heat** gun can be used to follow the path of the forming tool on the other side of the sheet, causing the material to heat up and become more malleable. This allows for a great maximum forming depth and sheets with increased thickness to be formed. Heat application can also aid in increasing the accuracy of the formed part. The other option is a generic **support** tool with a flat head and a thick body. This tool provides leverage at the periphery of the shape, increasing the accuracy of the forming process.

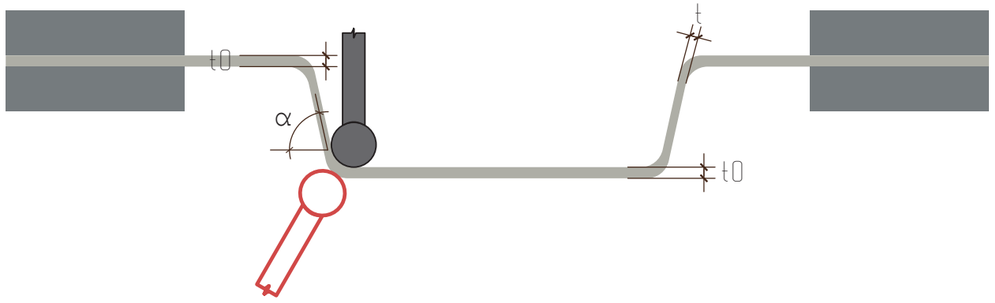
Maximum Wall Angle

Cosine's Law

Since the sheet is stretched by the forming process material thinning occurs. So there is a maximum wall angle, α_{max} that can be formed in a single step. Beyond α_{max} the sheet will tear. As the draw angle, α, of the forming tool increases, the sheet thickness at the formed point, t, decreases. For most materials, angles up to 65° can be formed. The thickness can be approximated by cosine's law, defined as:

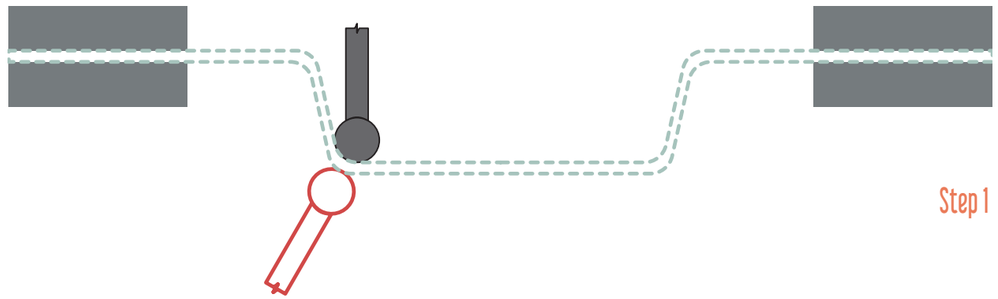
$t = t0 \times \cos\alpha$

t = thickness at formed point
$t0$ = initial sheet thickness
α = draw angle

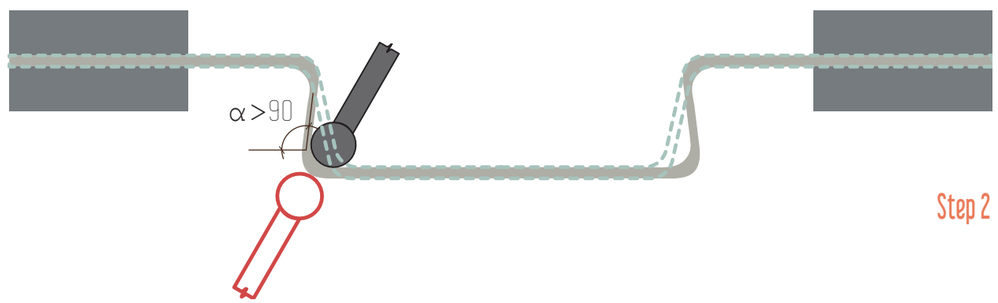


DPIF-L

One way to increase the maximum wall angle is to use a secondary support tool local to the forming tool. This provides additional force applied to the formed point, and greater leverage to form the sheet. Consequently, α_{max} increases by about 12.5° for a new α_{max} of about 77.5°.



Step 1



Step 2

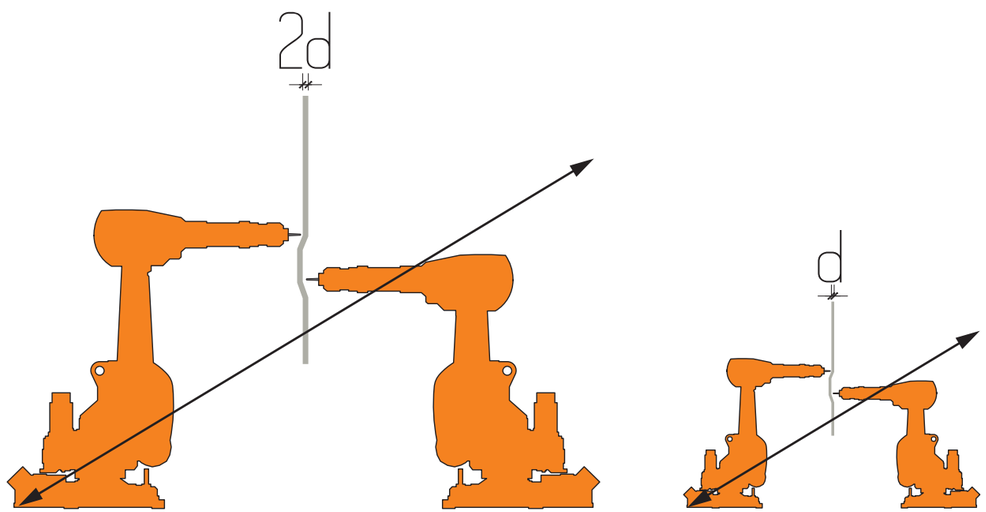
Multiple Forming

If angles greater than amax need to be formed, a process known as multiple forming can be utilzied. In this process formed areas are formed twice by non-identical paths. The first pass forms the material up to the amax of the material. The second pass forms the desired final angle. This process even allows for angles greater than 90° to be formed, creating what would be considered undercuts in a milling process.

Sources:

Bruninghaus, 2
Meier, DPIF-L, 327-328

Roboforming is Scalable



Maximum Sheet Thickness is Dependent on Forming Forces

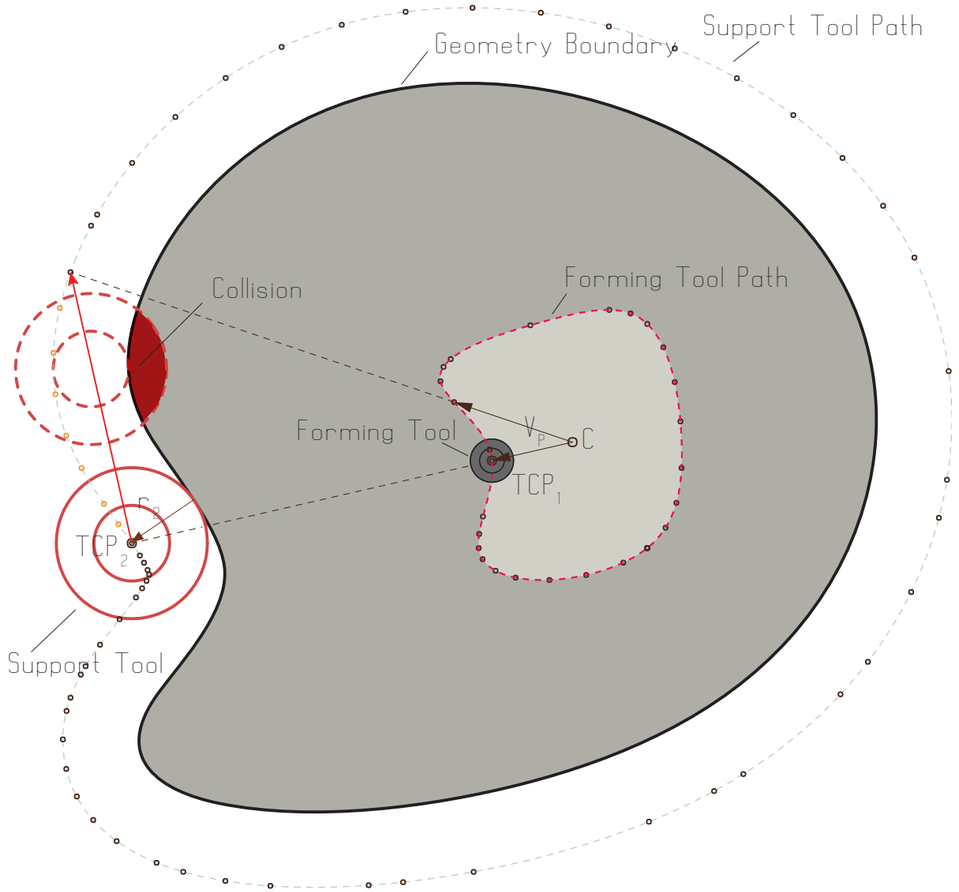
Roboforming is scalable. The maximum thickness d for roboforming is an equation on the maximum force f the robot can apply. So the thicker the material you wish to form, the stronger the robot needs to be. However, the width x and height y of the sheet are not a factor. The forces resulting from forming an $x \times y$ sheet are the same as the ones from a $2x \times 2y$ sheet. This makes industrial robots more cost effective, since a robot is generally less expensive than a CNC-machine with a comparable work area. This makes roboforming appropriate for applications at many scales.

Sources:

Bruninghaus, 1-2

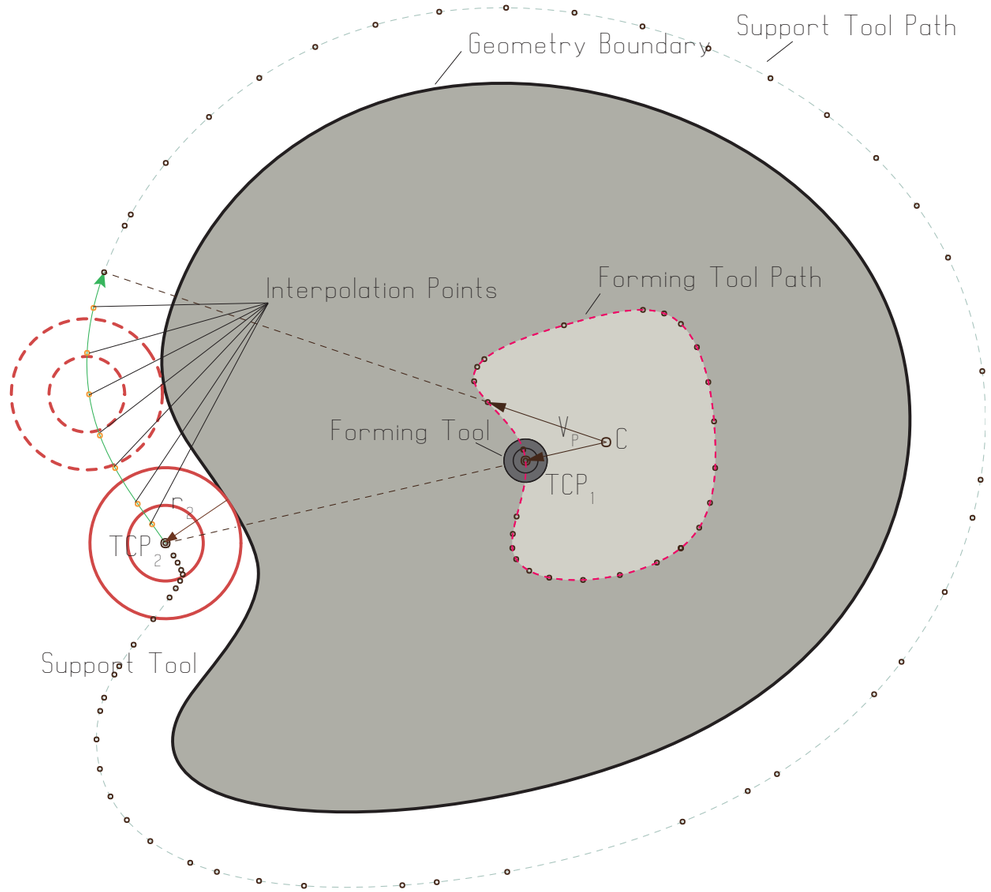
Meier, 37

Calculating Support Tool Position - Peripheral Support



The Problem

In DPIF-P, the support tool moves repeatedly in a constant plane along the boundary of the forming geometry. The support tool's position, TCP_2, is calculated based on the forming tool's position, TCP_1, relative to the centroid of the geometry, C. The centroid is typically the center of the final contour of the forming tool's path. The resulting vector, V_p, is extended until it intersects the support tool's path. The center point of the support tool, TCP_2, is located at the center of the base of the tool and it's path is simply an offset of the geometry boundary with a magnitude of the tool's radius, r_2. However, since the contour paths later in the run of the file have fewer targets than the border curve, there is a danger of the support tool running into the formed geometry on its way to the calculated next target.

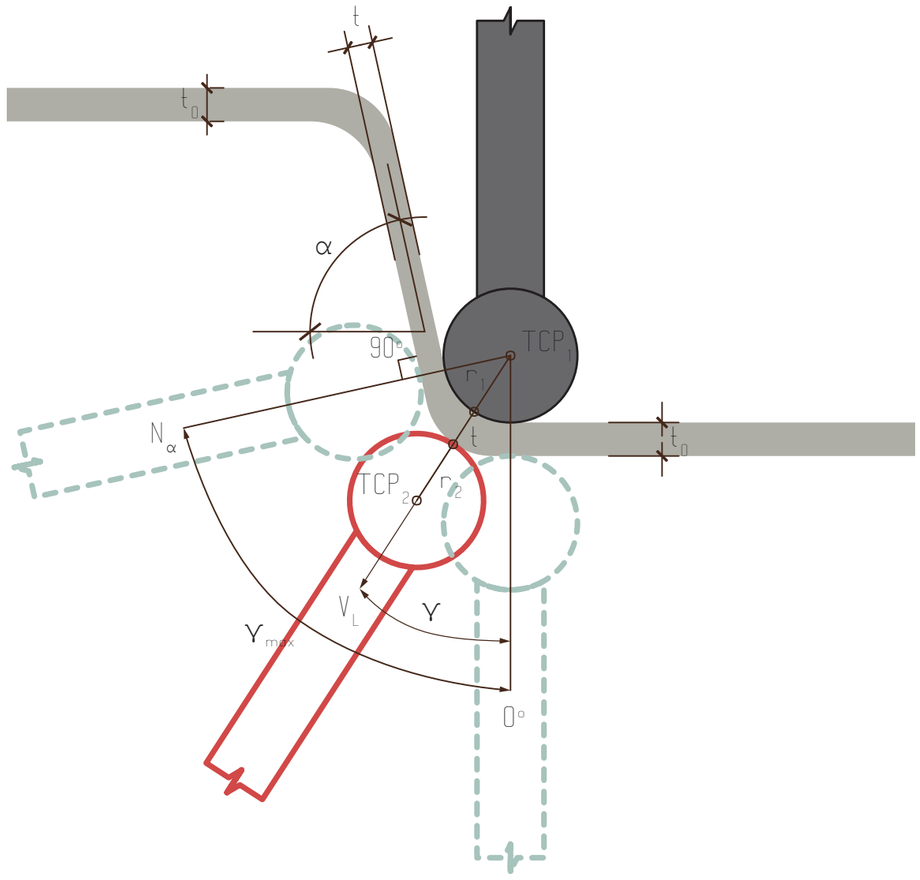


The Solution

To solve this problem a series of interpolation targets along the border curve from the current tool target to the next tool target. will need to be inserted into the Support Tool Path. What this ends up meaning is that the Support Tool Path will have more targets than the Forming Tool Path. This complicates the calculation of the synchronized times. However, a function can be written to calculate the new times based on this interpolation method.

Sources:
Kreimeier, CAD, 6-7

Calculating Support Tool Position - Local Support



Zone for Local Support

The center point of the supporting forming tool, TCP_2, needs to be along the vector, V_L, from the center point of the master forming tool, TCP_1, to the point of contact, Pc. TCP_2 is translated from TCP_1 along V_L with a magnitude, M_v, of:

$M_v = r1 + t + r2$

$r1$ = radius of master forming tool
t = thickness of formed sheet
$r2$ = radius of supporting forming tool

This allows the supporting forming tool to be in a variety of positions and orientations. The shifting angle, γ_{max}, is bound between 0°, when the tools are directly opposite eachother, and N_α, the angle normal to the wall angle, α.

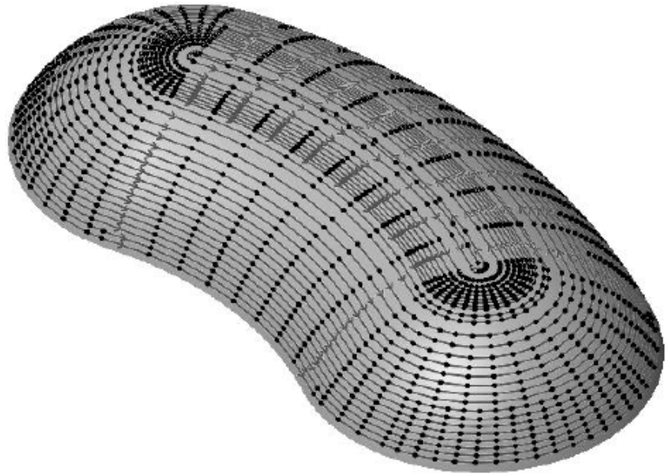


Image credit: Meier, Accuracy, 5

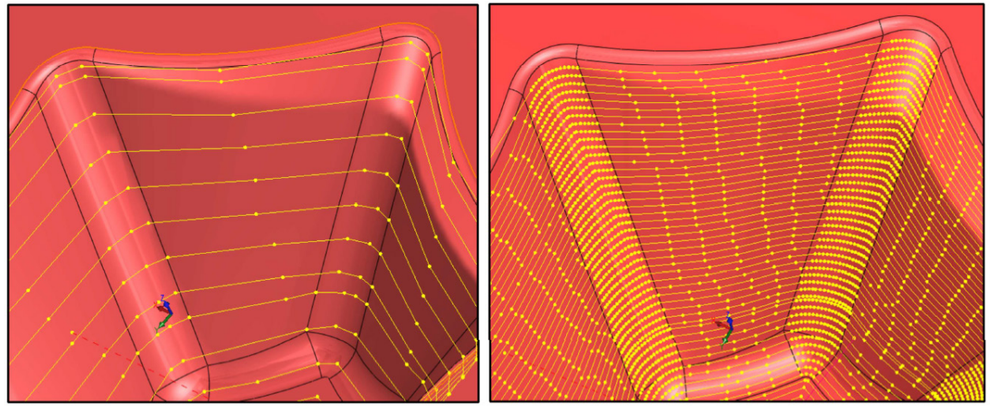


Image credit: Kreimeier, CAM, 892

Divide by Curvature to Increase Computational Efficiency

The robots do not follow paths or curves but rather are instructed to go to a series of targets. The tool-head can only move is a straight line between two targets. Therefore, the resulting toolpath is a faceted polyline. To increase the accuracy and surface quality of the forming, a smaller step size is desired. The more targets, the higher the accuracy. However, it is a waste of the computer's resources to place targets at every small interval as inevitably targets would be placed where they really were not needed. Ideally, there should be more targets in areas of high curvature, and fewer where the curvature is less.

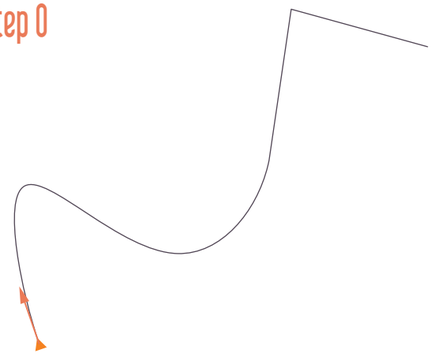
Sources:

Meier, Accuracy, 5
Kreimeier, CAM, 892

Scripts

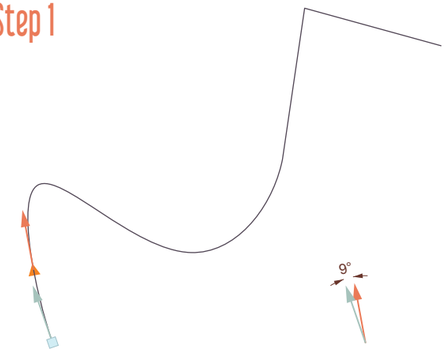
Divide by Tangent - Concept

Step 0



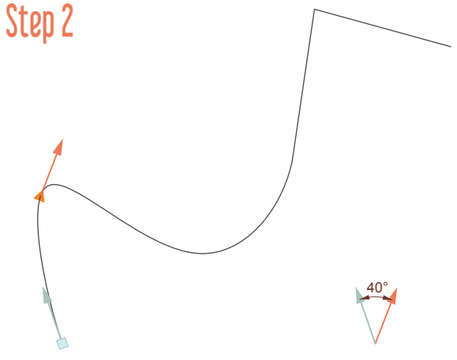
A rover travels along the path, saving the tangent vector at each point. The rover places a point at the start of the curve.

Step 1



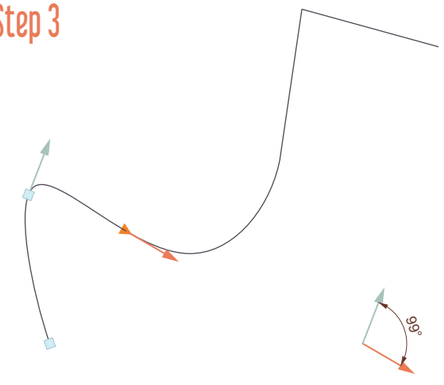
The rover steps by a given step size and compares the previous tangent to the current tangent. Here, the change in tangent angle is less than theta, so the point is not created.

Step 2



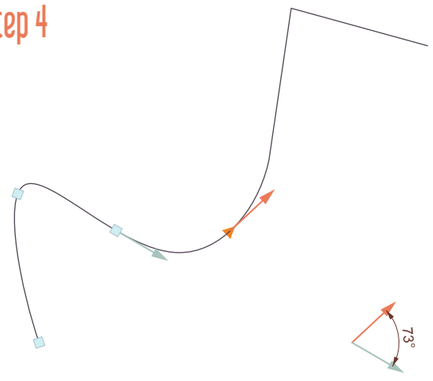
The rover takes another step and compares this new tangent to the original tangent at the start of the curve. Since the angles changes by more than theta, a point is placed.

Step 3



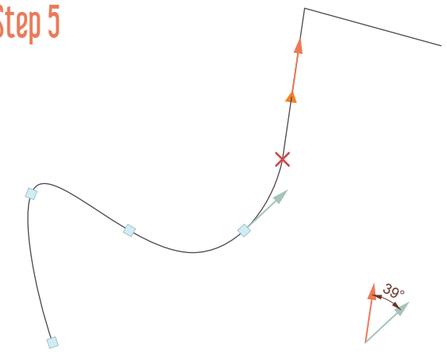
The rover steps and determines that a point should be created.

Step 4



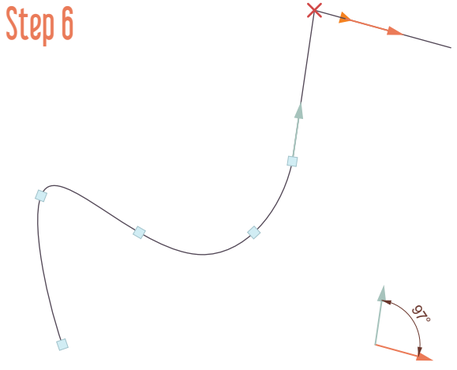
The rover steps and determines that a point should be created.

Step 5



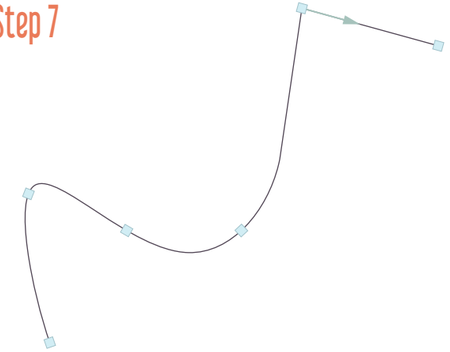
The rover steps and determines that a discontinuity exists between the previous point and the new point. So the rover places at point at the discontinuity and updates the previous tangent.

Step 6



The rover steps and determines that a discontinuity exists between the previous point and the new point. So the rover places at point at the discontinuity and updates the previous tangent.

Step 7

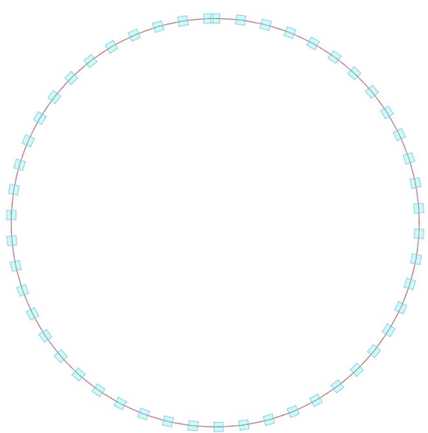


The rover places a point at the end.

Divide by Change in Tangent Vector - Demonstration

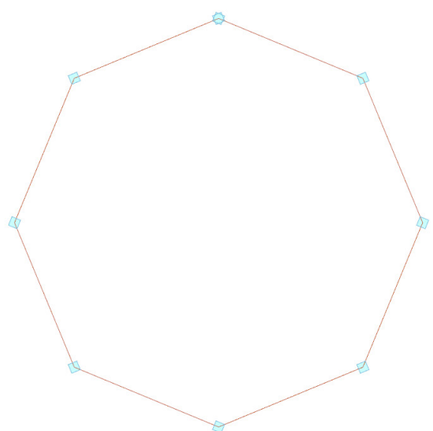
Theta: the minumum change required

Circle



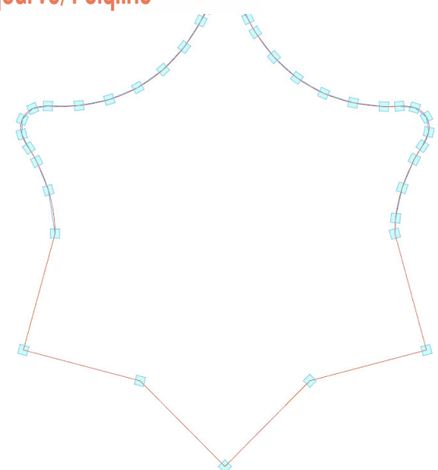
pts: 52 accuracy: 100%

Polygon



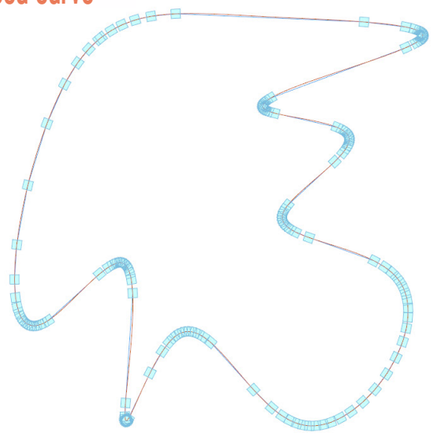
pts: 9 accuracy: 100%

Polycurve/Polyline



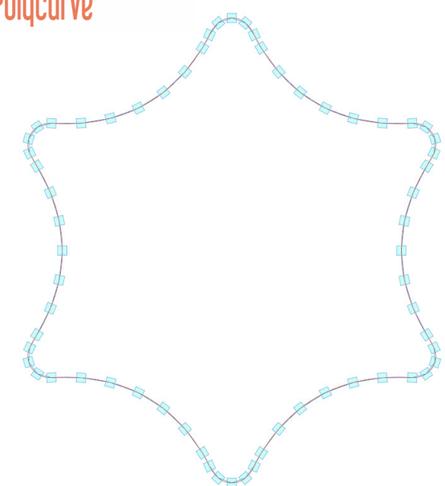
pts: 40 accuracy: 100%

Closed Curve



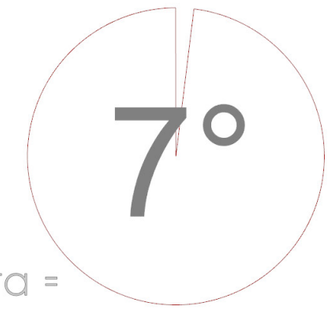
pts: 168 accuracy: 100%

Polycurve

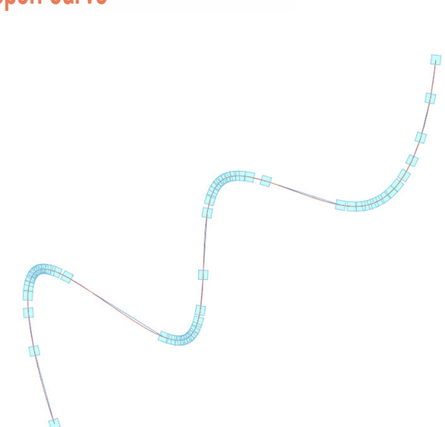


pts: 65 accuracy: 100%

theta =



Open Curve



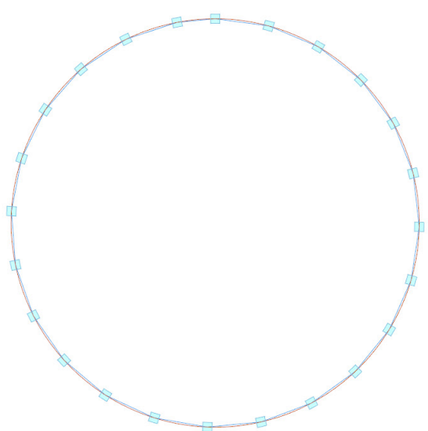
pts: 62 accuracy: 100%

interpolated polycurve
input curve
calculated points

Divide by Change in Tangent Vector - Demonstration

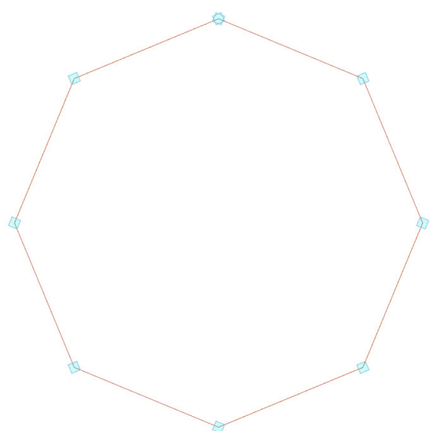
Theta: the minumum change required

Circle



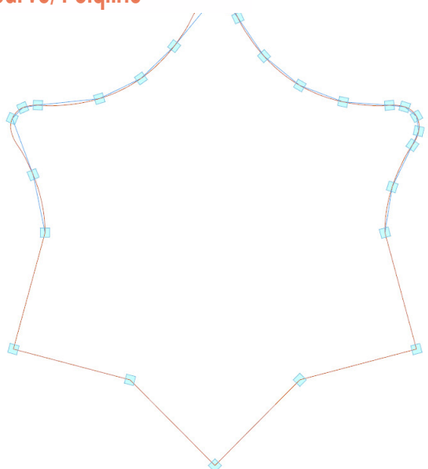
pts: 25 accuracy: 99%

Polygon



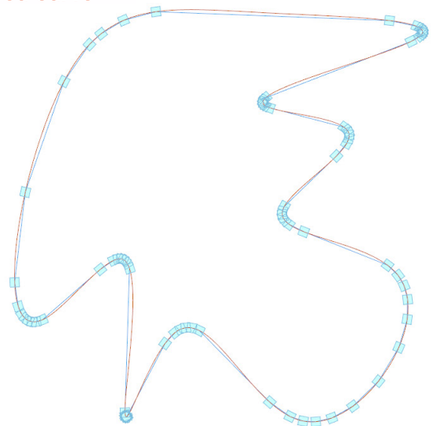
pts: 9 accuracy: 100%

Polycurve/Polyline



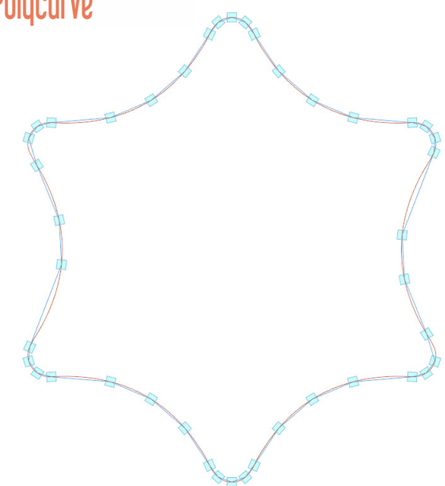
pts: 28 accuracy: 99%

Closed Curve

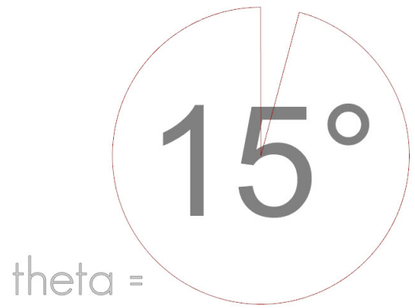


pts: 81 accuracy: 99%

Polycurve



pts: 43 accuracy: 99%



Open Curve

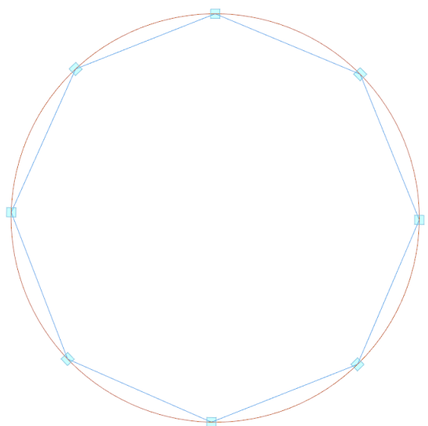
pts: 30 accuracy: 100%

interpolated polycurve
input curve
calculated points

Divide by Change in Tangent Vector - Demonstration

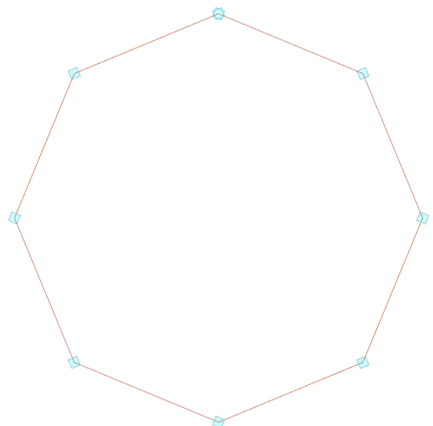
Theta: the minumum change required

Circle



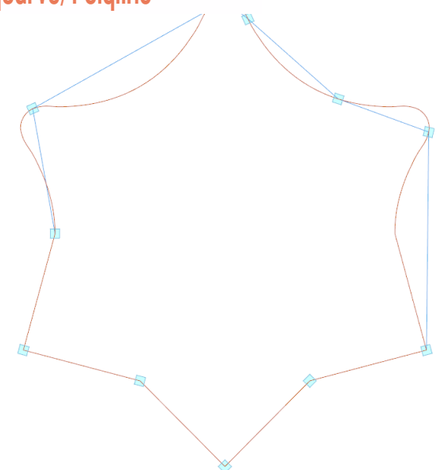
pts: 9 accuracy: 93%

Polygon



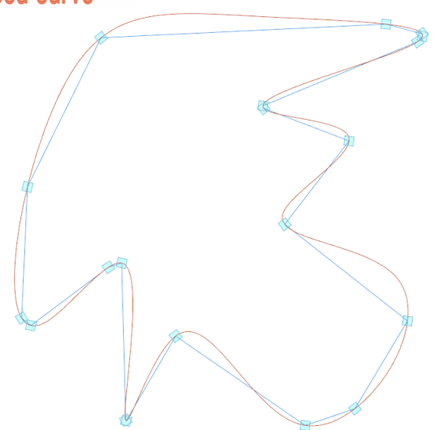
pts: 9 accuracy: 100%

Polycurve/Polyline



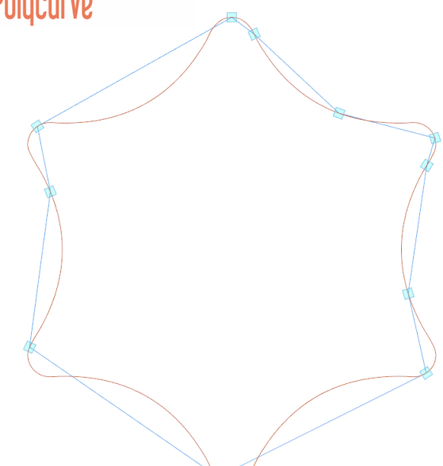
pts: 12 accuracy: 93%

Closed Curve



pts: 21 accuracy: 95%

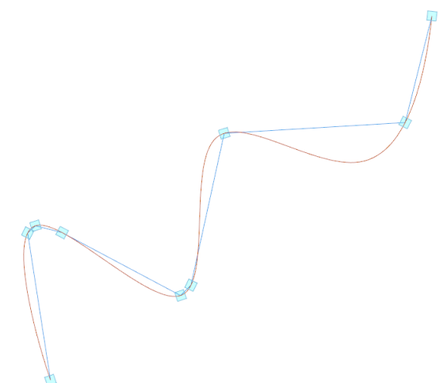
Polycurve



pts: 12 accuracy: 90%

theta = 45°

Open Curve



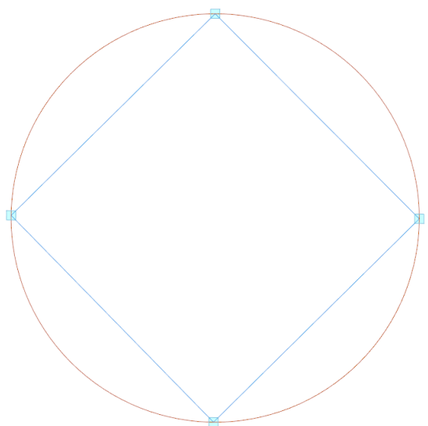
pts: 9 accuracy: 93%

interpolated polycurve
input curve
calculated points

Divide by Change in Tangent Vector - Demonstration

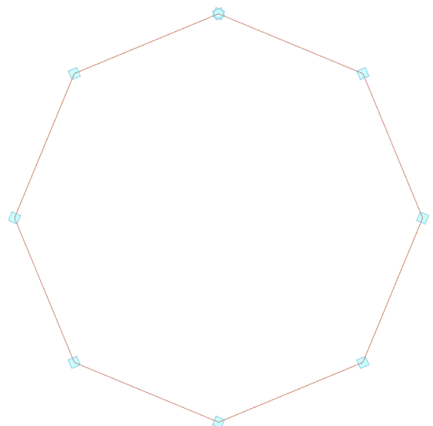
Theta: the minumum change required

Circle



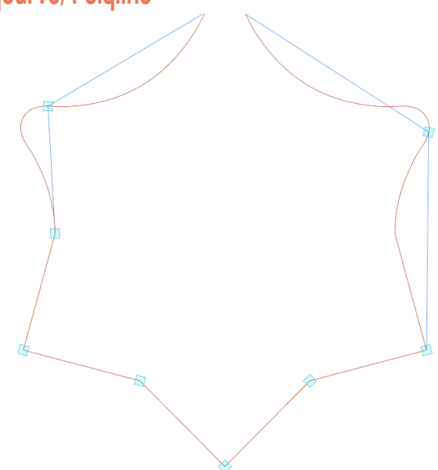
pts: 5 accuracy: 74%

Polygon



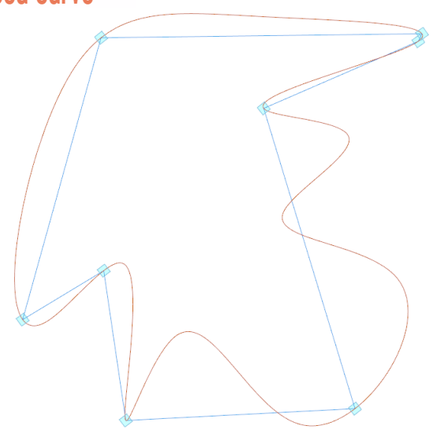
pts: 9 accuracy: 100%

Polycurve/Polyline



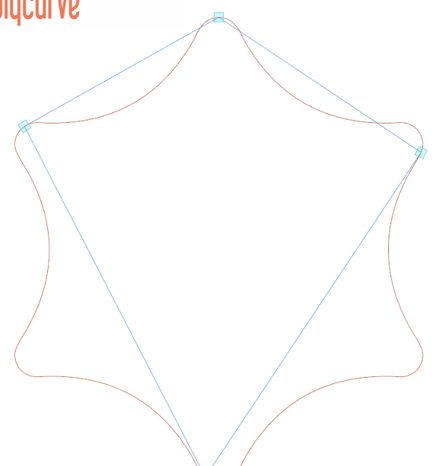
pts: 10 accuracy: 90%

Closed Curve



pts: 9 accuracy: 71%

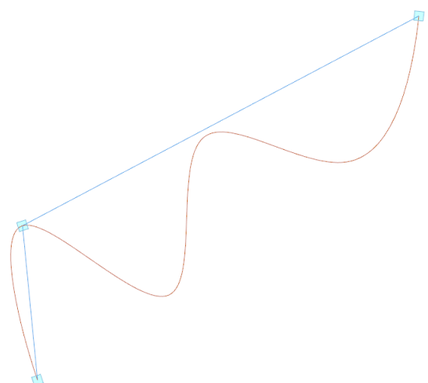
Polycurve



pts: 5 accuracy: 74%

theta = 90°

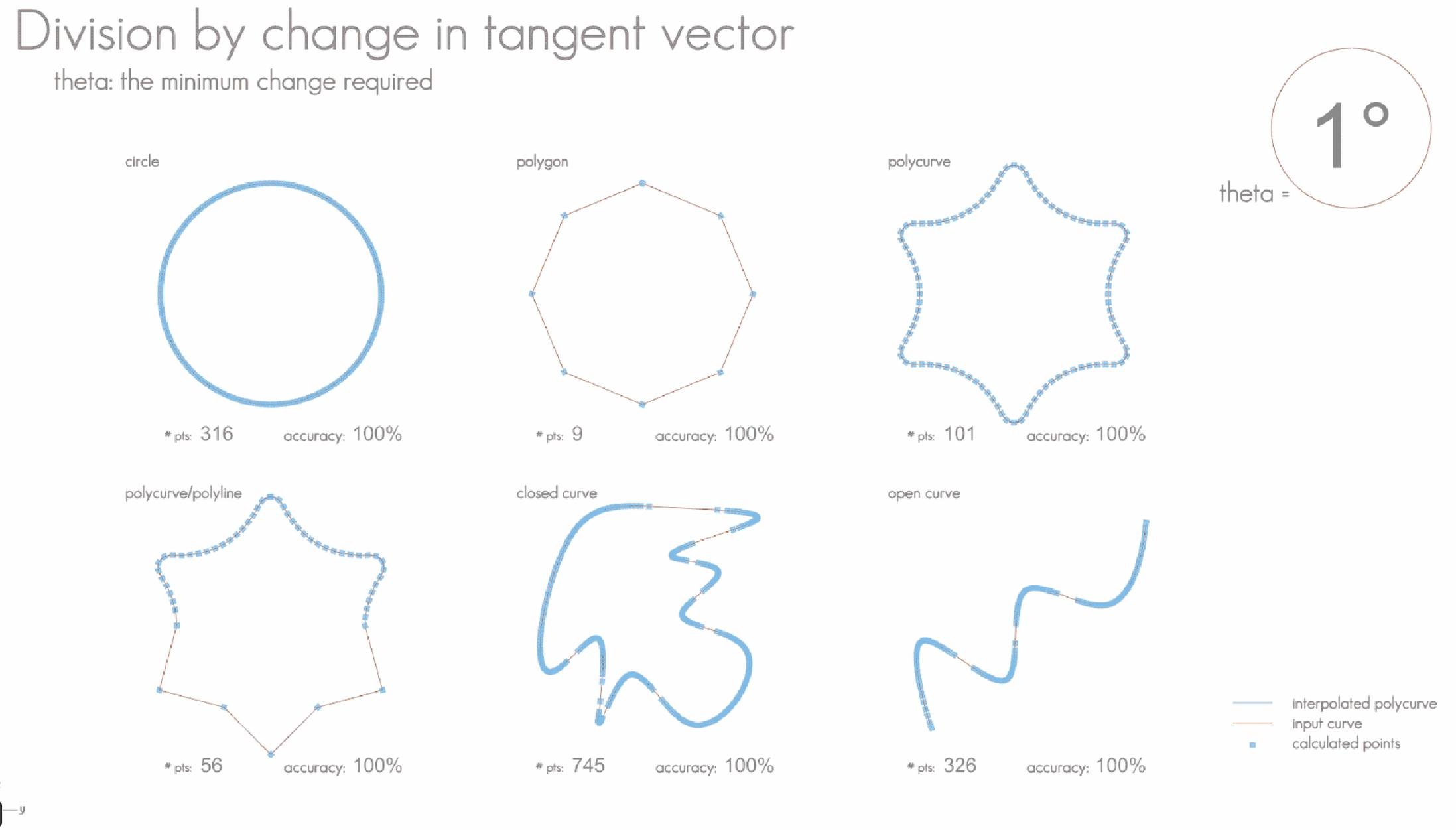
Open Curve



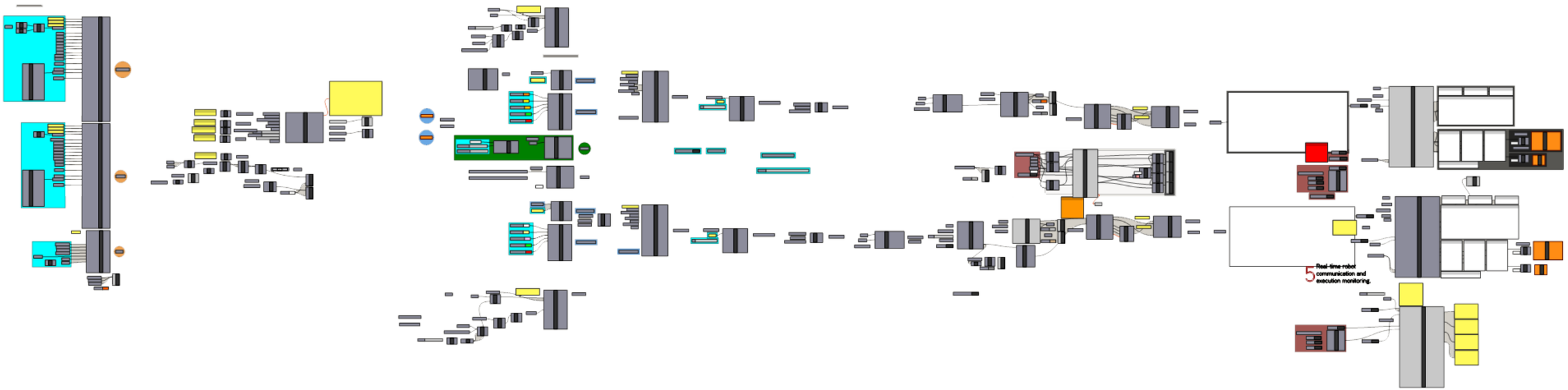
pts: 3 accuracy: 69%

interpolated polycurve
input curve
calculated points

Divide by Change in Tangent Vector - Animation



Grasshopper Script



Declare Robots

Create Robot Targets

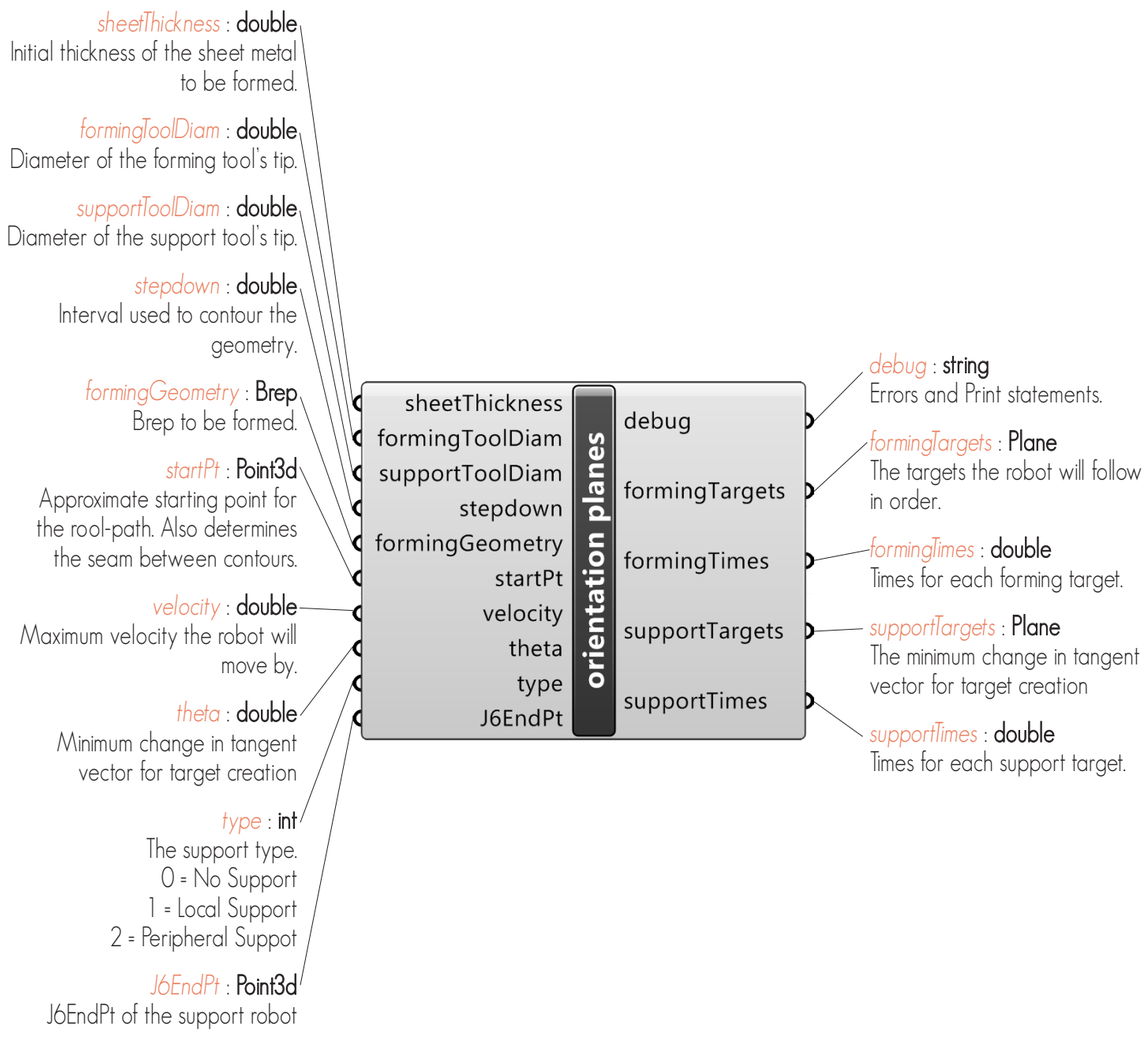
Declare Tools and Target Parameters

Simulate

Generate RAPID Code

Upload to Controller

Orientation Planes Component



The Orientation Planes component calculates the tool-path targets for the two robots. The goal of this compoenent is for it to be a safe, error-handling script that anyone with a basic knowledge of grasshopper can use without worrying about whether they are going to break the robots. An additional goal is to open source it and make it available to other students and fabricators to add additional functions. In order to realize this goal it is important that the code be clean and easy for even a novice programmer to understand and modify.


```
87  private void RunScript(double sheetThickness, double formingToolDiameter, doub
88  {
89    //Declare variables
90    int NO_SUPPORT = 0;
91    int LOCAL_SUPPORT = 1;
92    int PERIPHERAL_SUPPORT = 2;
93    Surface formingSurface = formingGeometry.Faces[0];
94    double[] times = new double[0];
95    LinkedList < Plane > supportTargs = null;
96    //Contour forming geometry based on stepdown
97    //include formingToolDiameter so the tool does not project past the surface
98    Curve[] contours = contourGeo(formingGeometry, stepdown, startPt,
99      formingToolDiameter / 4);
100   //Calculate planes where the tool will make contact with the geometry
101   LinkedList<Plane> contactPlns = divideCrvsByDeltaTan(contours, minDeltaTan);
102   //Create forming targets by offsetting the contact planes by the forming
103   //tool diameter
104   LinkedList<Plane> formingTargs = offsetPlnsFromNormal(contactPlns,
105     formingSurface, formingToolDiameter / 2);
106
107   if(type == NO_SUPPORT)
108   {
109     //Do not calculate support targets
110     //To avoid outputting null to the support targets, give it the home target
111     Plane[] pln = {new Plane(J6EndPt, new Vector3d(-1, 0, 0))};
112     supportTargs = new LinkedList<Plane>(pln);
113     times = calcTimes(formingTargs, velocity);
114     formingTimes = times;
115   }
116   else if(type == LOCAL_SUPPORT)
117   {
118     //Create support targets based on the contact planes
119     //Offset the contact planes by the forming tool diameter and sheet
120     //thickness.
121     supportTargs = offsetPlnsFromNormal(contactPlns, formingSurface,
122       -((formingToolDiameter / 2) + sheetThickness));
123     supportTargs = flipNormal(supportTargs);
124     //Calculate the times based on the support targets since they span
125     //greater distances than the forming targets
126     times = calcTimes(supportTargs, velocity);
127     //Both the forming and supports times have the same number of points
128     //so set their times are equal.
129     formingTimes = times;
130     supportTimes = times;
131   }
132   else if(type == PERIPHERAL_SUPPORT)
133   {
134     //Calculate center of forming geometry based on last contour
135     Point3d center = getCentroid(contours[contours.Length - 1]);
136     //The boundary curve will be the first element in contours
137     Curve peripheralCrv = contours[0];
138     supportTargs = calcPeripheralPlns(formingTargs, peripheralCrv, center,
139       supportToolDiameter / 2);
140     times = calcTimes(supportTargs, velocity);
141     //Both the forming and supports times have the same number of points
142     //so set their times are equal.
143     formingTimes = times;
144     supportTimes = times;
145   }
146   formingTargets = formingTargs;
147   supportTargets = supportTargs;
148 }
```

Orientation Planes Component

```
/*
contourGeo : Brep * double * Point3d * double -> Curve[]
REQUIRES: geo is oriented s.t. the inner surface is facing in the same
direction as the x axis.
sD > 0.
ENSURES: contourGeo(geometry, stepdown) returns an array of curves s.t. each
curve is the result of an intersection between the geometry and a yz plane
with the plane's origin at the furthest point in the x axis + a displacement
vector in the negative x axis of stepDown*curveNumber.
*/
private Curve[] contourGeo(Brep geo, double stepDown, Point3d startPt,
  double r)
{
  double tolerance = 0.0000001;
  //Get depth of forming geometry
  BoundingBox bbox = geo.GetBoundingBox(true);
  Point3d p1 = bbox.Corner(false, true, true);
  Point3d p2 = bbox.Corner(true, true, true);
  p2.Transform(Transform.Translation(new Vector3d(r, 0, 0)));
  double depth = p1.X - p2.X;
  //Get number of contours based on depth and stepdown
  int numContours = (int) (depth / stepDown);
  //Set up final plane
  //This requires nudging the intersection plane
  Vector3d xAxis = new Vector3d(1, 0, 0);
  Point3d finalPt = p2;
  Vector3d vec = new Vector3d(tolerance, 0, 0);
  Transform nudgeX = Transform.Translation(vec);
  finalPt.Transform(nudgeX);
  Plane finalPlane = new Plane(finalPt, xAxis);
  //Set up parameters for contouring
  Plane[] planes = new Plane[numContours];
  vec = new Vector3d(-stepDown, 0, 0);
  Transform movex = Transform.Translation(vec);
  Point3d pt = p1;
  double seamParam;

  //Add one to the number of contours to account for the final intersection
  Curve[] contours = new Curve[numContours + 1];

  //Contour Brep
  for(int i = 0; i < numContours; i++)
  {
    planes[i] = new Plane(pt, xAxis);
    contours[i] = Brep.CreateContourCurves(geo, planes[i])[0];
    contours[i].ClosestPoint(startPt, out seamParam);
    contours[i].ChangeClosedCurveSeam(seamParam);
    //Increment
    pt.Transform(movex);
  }

  //Add in final curve to account for gaps in stepdown
  contours[numContours] = Brep.CreateContourCurves(geo, finalPlane)[0];
  contours[numContours].ClosestPoint(startPt, out seamParam);
  contours[numContours].ChangeClosedCurveSeam(seamParam);

  return contours;
}
```

```
/*
deg : double -> double
REQUIRES: true
ENSURES: deg(radian) converts radians to degrees
*/
private double deg(double rad)
{
  return rad * (180 / Math.PI);
}

/*
divideCrvsByDeltaTan : Curve[] * double -> LinkedList<Plane>
REQUIRES: theta > 0
ENSURES: divideCrvsByDeltaTan(crvs, theta) returns a linked list of planes
         along the curves s.t. there is a plane at every point along
         the curve where the change in the tangent vector between
         two points is greater than theta.
*/
private LinkedList<Plane> divideCrvsByDeltaTan(Curve[] crvs, double theta)
{
  //initialize parameters
  int n = crvs.Length;
  double stepSize = 0.5;
  Vector3d xAxis = new Vector3d(1, 0, 0);
  double rover; //steps along the curve by stepSize
  double oldRover; //stores the previous rover for comparison
  double discontinuity;
  Vector3d prevTan;
  Vector3d currTan;
  Curve crv;
  Interval dom;

  //initialize list
  Plane[] plns = {};
  LinkedList < Plane > targets = new LinkedList<Plane>(plns);
  Plane target;

  Rhino.Geometry.Continuity c = Rhino.Geometry.Continuity.C1_continuous;

  for(int i = 0; i < n; i++)
  {
    crv = crvs[i];

    //initialize data
    dom = crv.Domain;
    rover = dom.Min;

    //Add plane at start point of curve to list
    target = new Plane(crv.PointAt(rover), xAxis);
    targets.AddLast(target);

    //Increment
    prevTan = crv.TangentAt(rover);
    oldRover = rover;
    rover += stepSize;
```

Orientation Planes Component

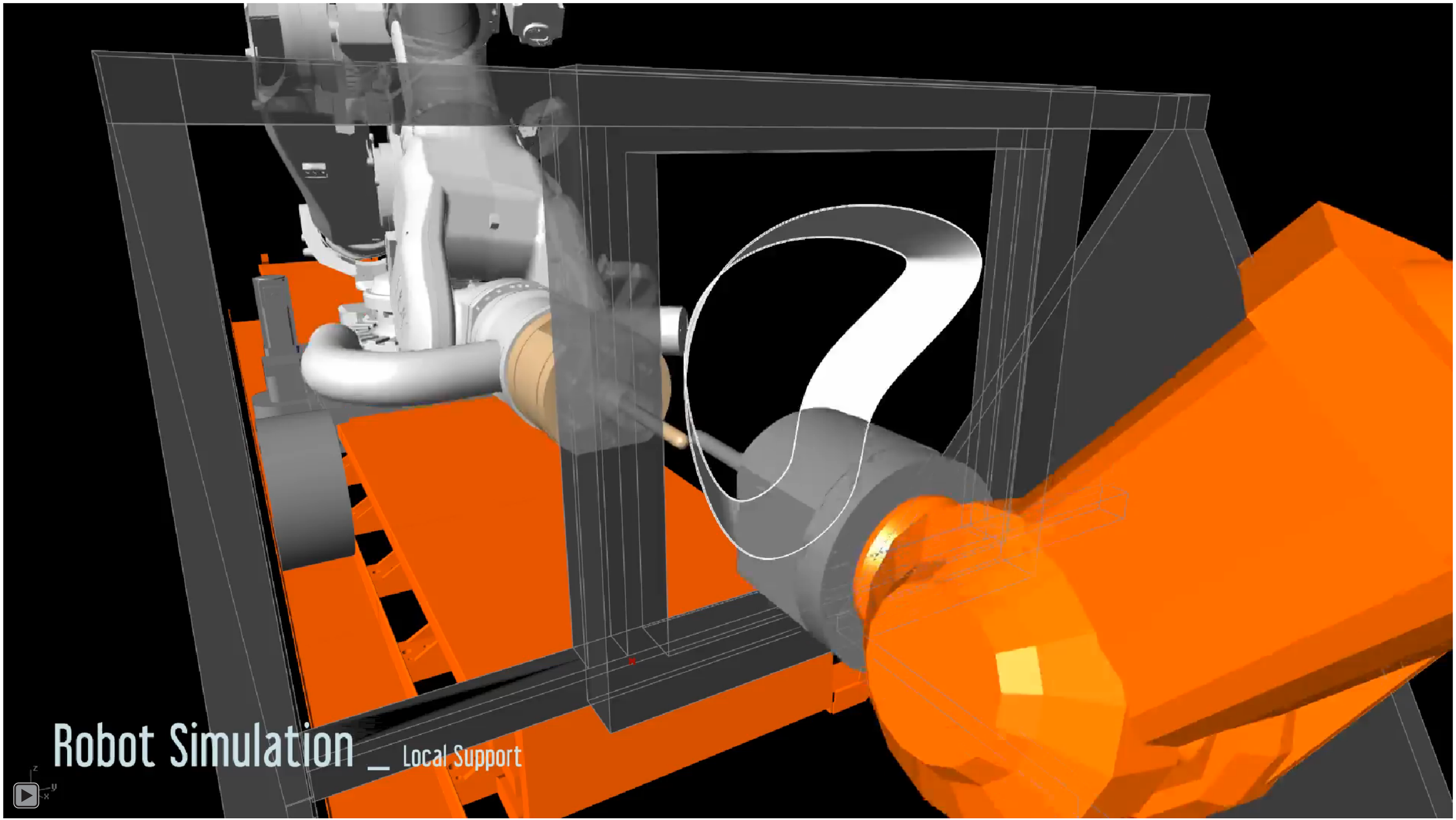
```
      while(rover < dom.Max)
      {
        currTan = crv.TangentAt(rover);

        //If there is a discontinuity between the oldRover and rover
        //then place a point at the discontinuity and update prevTan.
        bool isDisc = crv.GetNextDiscontinuity(c, oldRover, rover,
          out discontinuity);
        if(isDisc)
        {
          target = new Plane(crv.PointAt(discontinuity), xAxis);
          targets.AddLast(target);
          prevTan = crv.TangentAt(discontinuity);
        }

        //If the change in tangent vector is greater than theta,
        //then drop a target at the rover and update prevTan.
        double delta = deg(Math.Abs(Vector3d.VectorAngle(prevTan, currTan)));
        if(delta > theta)
        {
          target = new Plane(crv.PointAt(rover), xAxis);
          targets.AddLast(target);
          prevTan = currTan;
        }
        //Increment
        oldRover = rover;
        rover += stepSize;
      }

      //Add target at end point of curve
      target = new Plane(crv.PointAt(dom.Max), xAxis);
      targets.AddLast(target);
    }
    return targets;
  }

  /*
  movePlane : Plane * Vector3d -> Plane
  REQUIRES: true
  ENSURES: movePt(pt, vec) returns the pt translated by the vec
  */
  private Plane movePlane(Plane pl, Vector3d vec)
  {
    var movex = Transform.Translation(vec);
    pl.Transform(movex);
    return pl;
  }
```

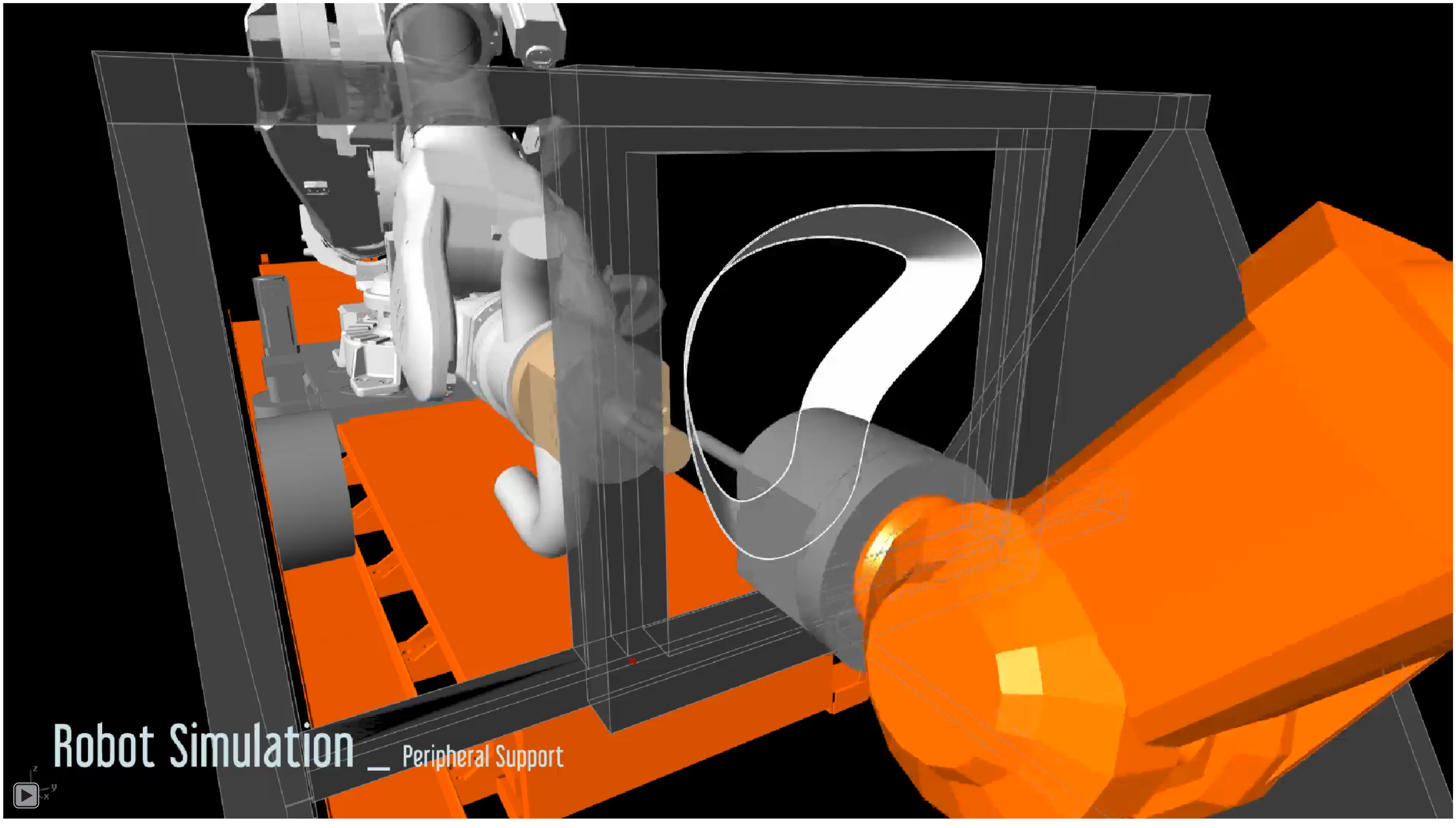
```
  /*
  offsetPlnsFromNormal : LinkedList<Plane> * Surface * double ->
    LinkedList<Plane>
  REQUIRES: true
  ENSURES: offsetPlnsFromNormal(pts, srf, mag) returns the pts translated by the
           normal vector at the closest point to the srf with a magnitude of mag
  */
  private LinkedList<Plane> offsetPlnsFromNormal(LinkedList<Plane> plns,
    Surface srf, double mag)
  {
    //Initialize variables
    Plane[] planes = {};
    LinkedList < Plane > offsetPlns = new LinkedList<Plane>(planes);
    double u,v;

    //Create a node to traverse the list
    LinkedListNode<Plane> p = plns.First;

    //Traverse the list and add the offset plane to the linked list
    while(p != plns.Last.Next)
    {
      srf.ClosestPoint(p.Value.Origin, out u, out v);
      offsetPlns.AddLast(movePlane(p.Value, srf.NormalAt(u, v) * mag));
      p = p.Next;
    }
    return offsetPlns;
  }

  /*
  flipNormal : LinkedList<Plane> -> LinkedList<Plane>
  REQUIRES: true
  ENSURES: flipNormal(plns) returns plns with the normal direction flipped
  */
  private LinkedList<Plane> flipNormal(LinkedList<Plane> plns)
  {
    //Initialize variables
    Plane[] planes = {};
    LinkedList < Plane > flippedPlns = new LinkedList<Plane>(planes);
    Plane pln;

    //Create a node to traverse the list
    LinkedListNode<Plane> p = plns.First;

    //Traverse the list and add the flipped plane to the linked list
    while(p != plns.Last.Next)
    {
      pln = p.Value;
      pln.Flip();
      flippedPlns.AddLast(pln);
      p = p.Next;
    }
    return flippedPlns;
  }
```

Orientation Planes Component

```
/*
calcTimes : Plane * Plane * double -> double
REQUIRES: vel > 0
ENSURES: calcTime(targ1, targ2, vel) determines the time necesary to ensure
the velocity between any two points = maxVel.
*/
private double calcTime(Plane targ1, Plane targ2, double vel)
{
  double distance = targ1.Origin.DistanceTo(targ2.Origin);
  return distance / vel;
}

/*
calcTimes : Plane[] * double -> double[]
REQUIRES: maxVel > 0
ENSURES: calcTimes(targs, maxVel) determines the time necesary to ensure the
         velocity between any two points = maxVel
*/
private double[] calcTimes(LinkedList<Plane> plns, double vel)
{
  //Initialize variables
  int numTargs = plns.Count;
  double[] times = new double[numTargs];
  //Set first targets's time to 30 secs to avoid the robot zooming to targ1
  times[0] = 30;
  LinkedListNode<Plane> p = plns.First.Next;
  int i = 1;

  //Traverse the list and calculate the time based on the distance/vel
  while(p != plns.Last.Next)
  {
    times[i] = calcTime(p.Previous.Value, p.Value, vel);
    //Increment
    p = p.Next;
    i++;
  }
  return times;
}

/*
getCentroid : Curve -> Point3d
REQUIRES: true
ENSURES: getCentroid(crv) returns the centroid of crv.
*/
private Point3d getCentroid(Curve crv)
{
  AreaMassProperties p = AreaMassProperties.Compute(crv);
  return p.Centroid;
}
```

```
/*
calcPeripheralPlns : LinkedList<Plane> * Curve * Point3d * double
  -> LinkedList<Plane>
REQUIRES: true
ENSURES: getCentroid(crv) returns the centroid of crv.
*/
private LinkedList<Plane> calcPeripheralPlns(LinkedList<Plane> formingPlns,
  Curve crv, Point3d center, double r)
{
  //Initialize Variables
  double tol = 0.000001;
  Point3d offsetPt = new Point3d(1000, 0, 0);
  Vector3d xAxis = new Vector3d(1, 0, 0);
  Line line;
  Plane pln;
  Point3d pt;
  Curve intCrv;
  Vector3d negXaxis = new Vector3d(-1, 0, 0);
  Plane[] planes = {};
  LinkedList < Plane > peripheralPlns = new LinkedList<Plane>(planes);

  //Offset crv by radius of support tool
  crv = crv.Offset(offsetPt, xAxis, r, tol, CurveOffsetCornerStyle.Round)[0];

  //Project center to plane of border curve
  center.X = crv.PointAtEnd.X;

  //Create node to traverse linked list
  LinkedListNode<Plane> p = formingPlns.First;

  //Traverse list of planes and determine intersection point on border curve
  while(p != formingPlns.Last.Next)
  {
    pt = p.Value.Origin;
    pt.X = crv.PointAtEnd.X;
    //Draw a Line from the center to the forming point
    line = new Line(center, pt);
    //Extend the Line to ensure that it intersects with the border curve
    line.Extend(0, 1000);
    //Convert the Line to a Curve so the CurveCurve intersect works
    intCrv = line.ToNurbsCurve();
    Rhino.Geometry.Intersect.CurveIntersections pts =
      Rhino.Geometry.Intersect.Intersection.CurveCurve(crv, intCrv, tol, tol);

    //Create a new plane and add it to the list of peripheral targets
    pln = new Plane(pts[0].PointA, negXaxis);
    peripheralPlns.AddLast(pln);
    //Increment
    p = p.Next;
  }
  return peripheralPlns;
}
/**/

}
```

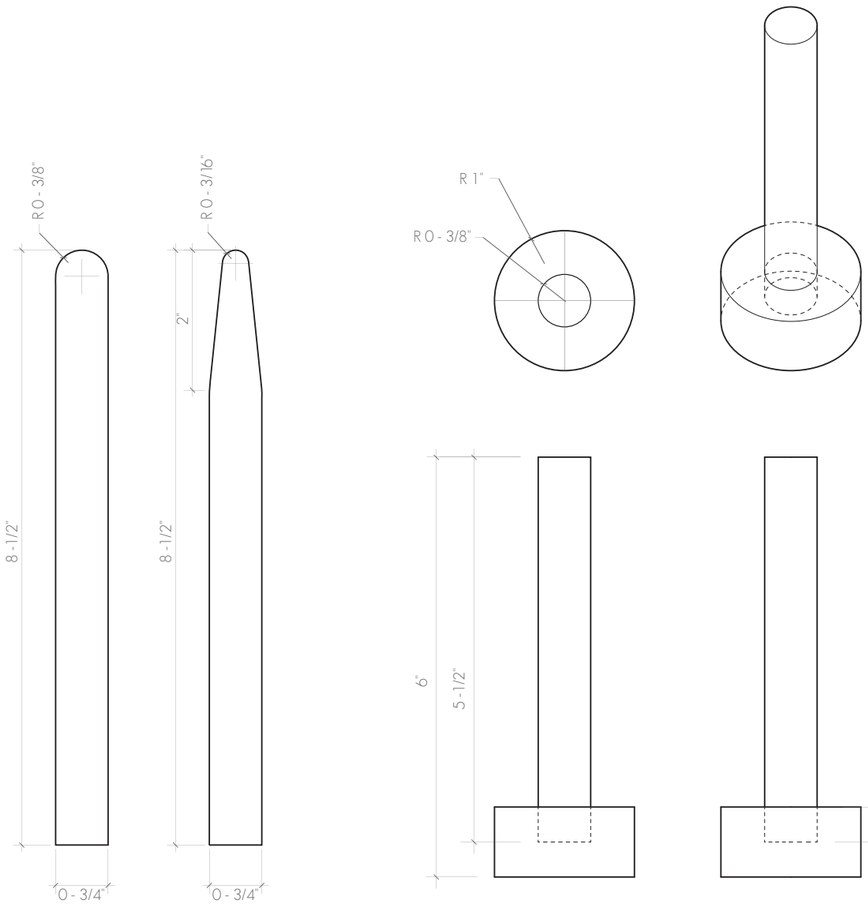

Robot Simulation _ Local Support


Robot Simulation _ Peripheral Support

Tools

Forming Tools

Design

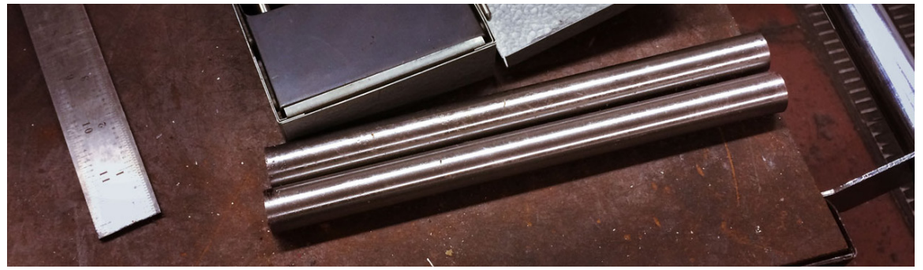


Forming Tools

Support Tool

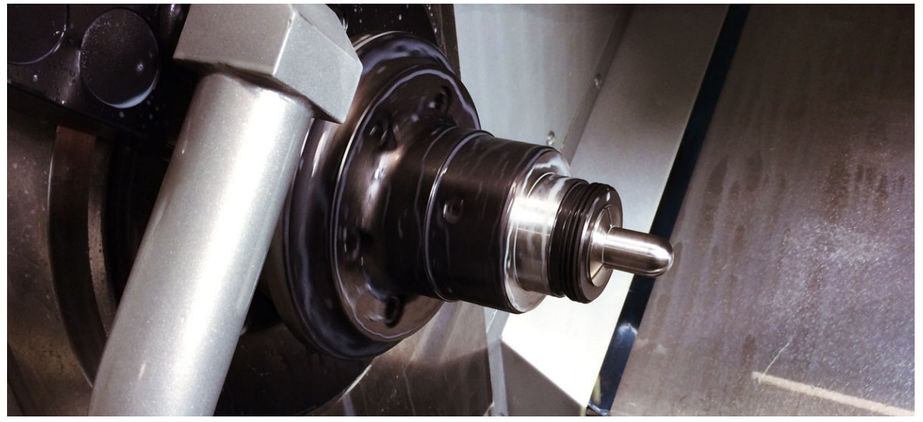
Two forming tools were made from A2 Tool Steel. A diameter of 3/4" was chosen as a healthy medium between the thickness needed to prevent the tool from bending while still within a reasonable cost range. The support tool is also made from the 3/4" rod in order to standardize the connection to the robot. Ideally the tools would be as long as possible to reduce the chance of the robot joint colliding with the forming geometry. However the longer the tool, the more likely it is to break. The tools were made on a metal lathe and heat-treated to increase their strength.

Machining Process



A2 Tool Steel 3/4" Rods

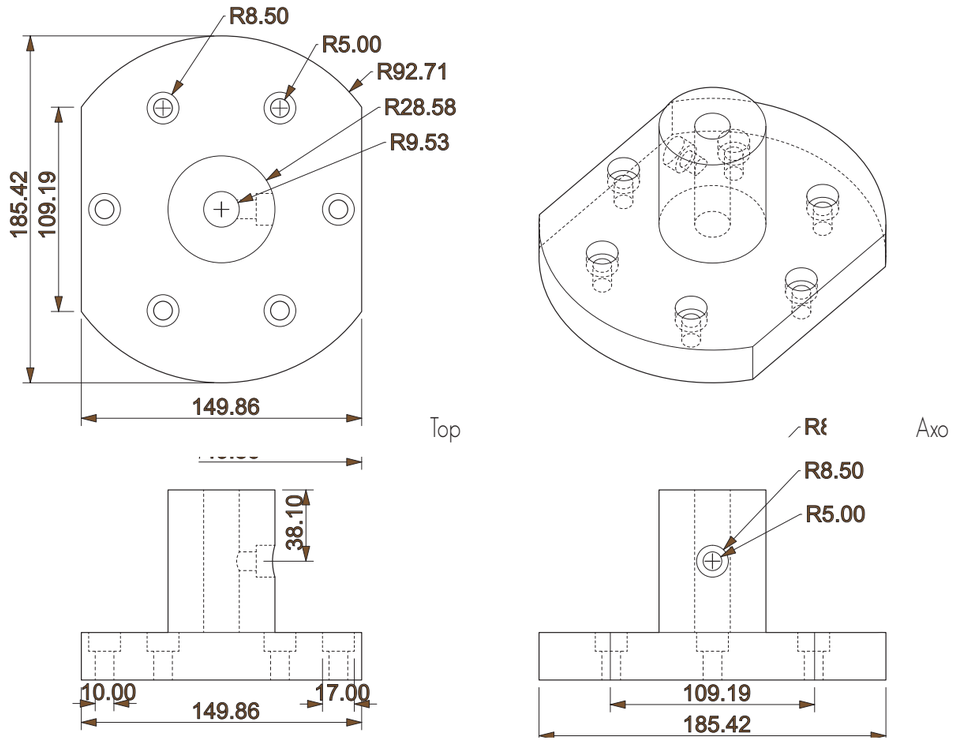
Lathe Turning



Result

Tool Holder

Design



Axo

units: millimeters

Tool Holder - Drawings

Tool Holder

The mechanism for holding the forming and support tools is straighforward. A 3" shaft with a set-screw secures the tool to the holder. The holder is bolted to the ATI AC-110 Tool Changer attached to joint 6 of both robots via 6 holes in the base of the tool holder. The manufacturing process involved turing a 7-1/2"x6" block of Multipirpose 6061 Aluminum on a lathe, then milling the holes for the bolts.

Machining Process

Lathe

Mill

The Frame

Modularity of the Frame

Precedents

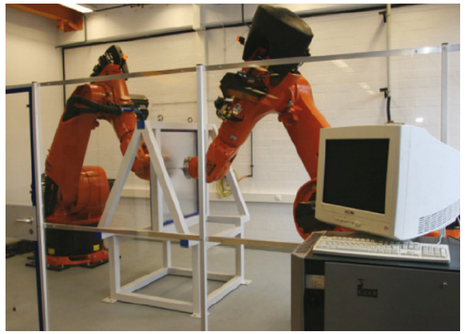


Image credit: (Meier et al., Roboforming, 601)

Image credit: (Meier et al., Roboforming, 601)

Image credit: (Meier et al., Roboforming, 601)

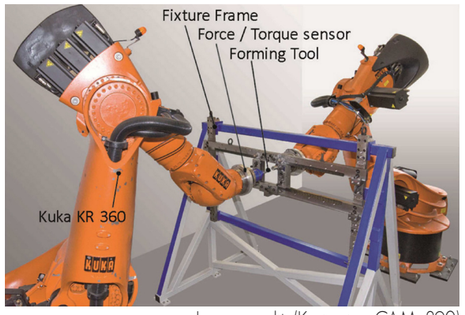
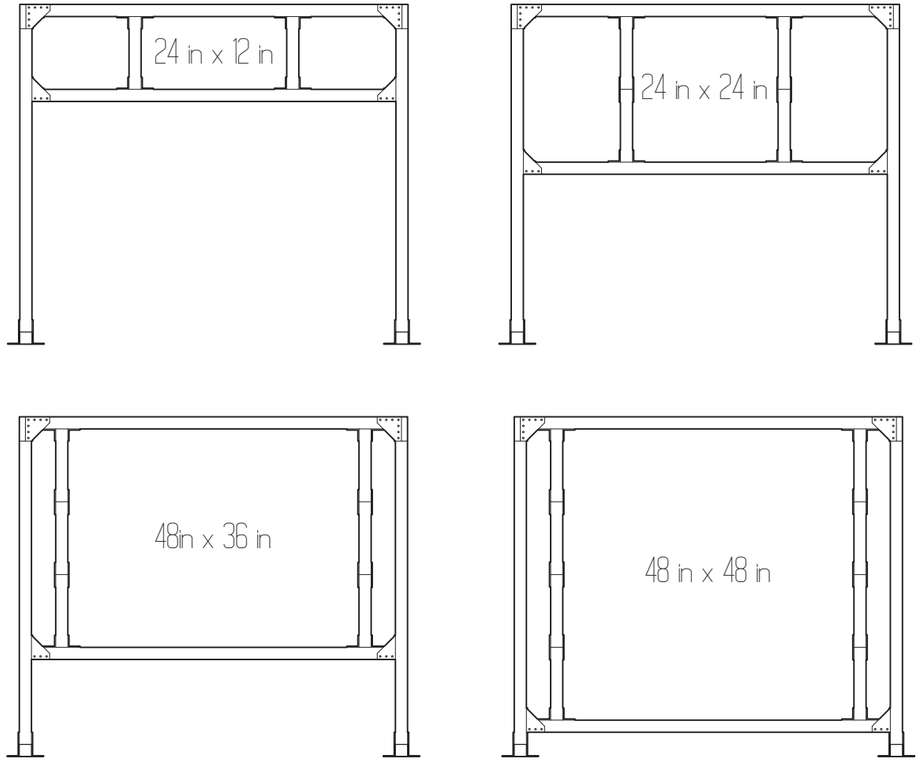


Image credit: (Kreimeier, CAM, 890)

Image credit: (University of Michigan)

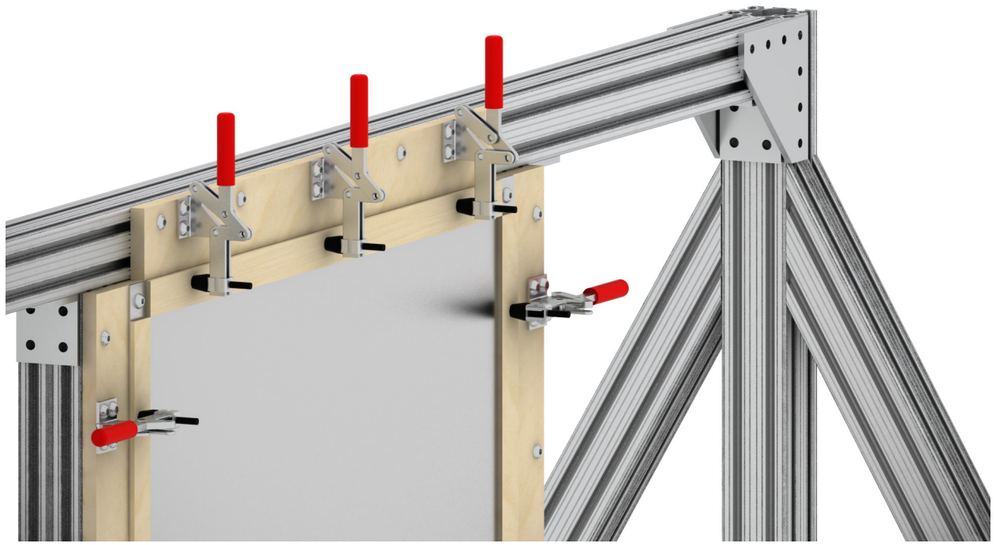
Modular Design



The frame I will be constructing needs to be modular for it to be capable of holding sheets of material of various dimensions. This is one of the reasons t-slotted extruded aluminum was chosen as the frame material. The t-slotted components allow for changes to be made to the positioning of the horizontal and vertical bars that define the sheet boundaries. The vertial bars are in turn made up of 1 foot segments of the extruded aluminum, joined together by a metal plate. This allows for varied sized sheets in the y dimension, in 1 foot increments. The largest dimension possible is 48in x 48in, chosen to reduce waste, since sheet materials come in 4ft x 8ft standard sizes. The larger dimensions, such as 48in x 36in and 48in x 48in are most likely too large for accurate forming given the thickness of the sheet metal. However, wood and other materials could be attached to the frame for other students to use in their projects. This is a multi purpose frame for any project needing to use two robots simultaneously acting on a sheet material.

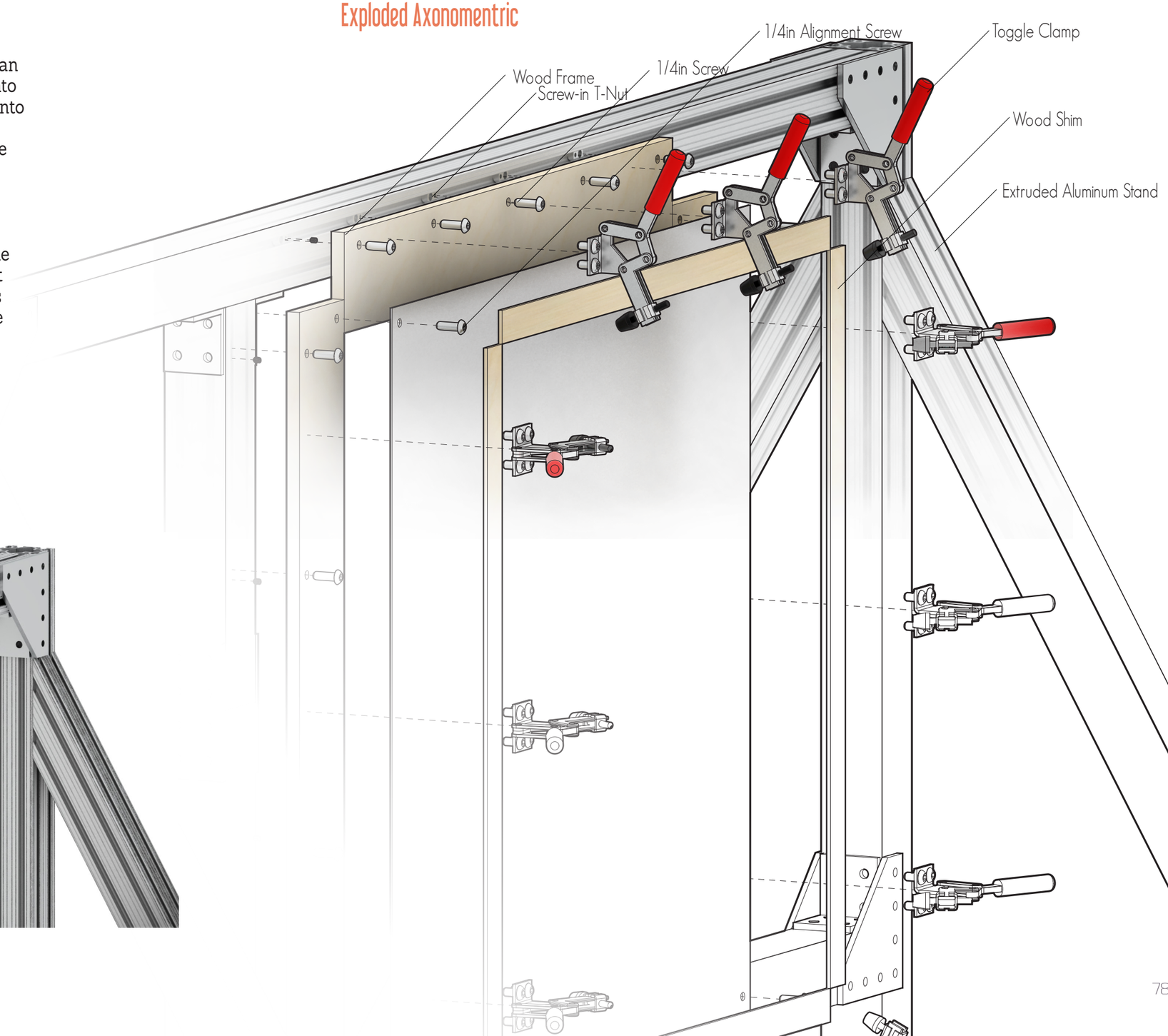
Sheet Attachment Detail

Having worked with sheet forming before, and being aware of the challenges with attachment, I have come up with what I believe to be an ideal solution. I reduced the number of holes that need to be drilled into the metal sheet down to 4 due to the energy and time needed to drill into metal.
4 **alignment holes** are mostly used to ensure the sheet is secured in the same place each time.
Toggle clamps which have a clamping force of 500 lbs. do the real holding work These are ideal as they are inexpensive and have a low profile, so the robot does not collide with the clamps.
These clamps are screwed into a 1/5 inch **wood frame** that matches the dimensions of the blank. Wood was chosen as the frame material as it is easy and relatively inexpensive to mill custom dimensioned parts as needed. The wooden backplate also provides additional leverage since the extruded aluminum stand has rounded edges.
The toggle clamps act upon a thin **wooden shim** that serves merely to disperse the force across the length of the sheet.
Screw-in t-nuts are slide into place and recieve 1/4 inch screws which hold the wood frame to the **extruded aluminum stand**.



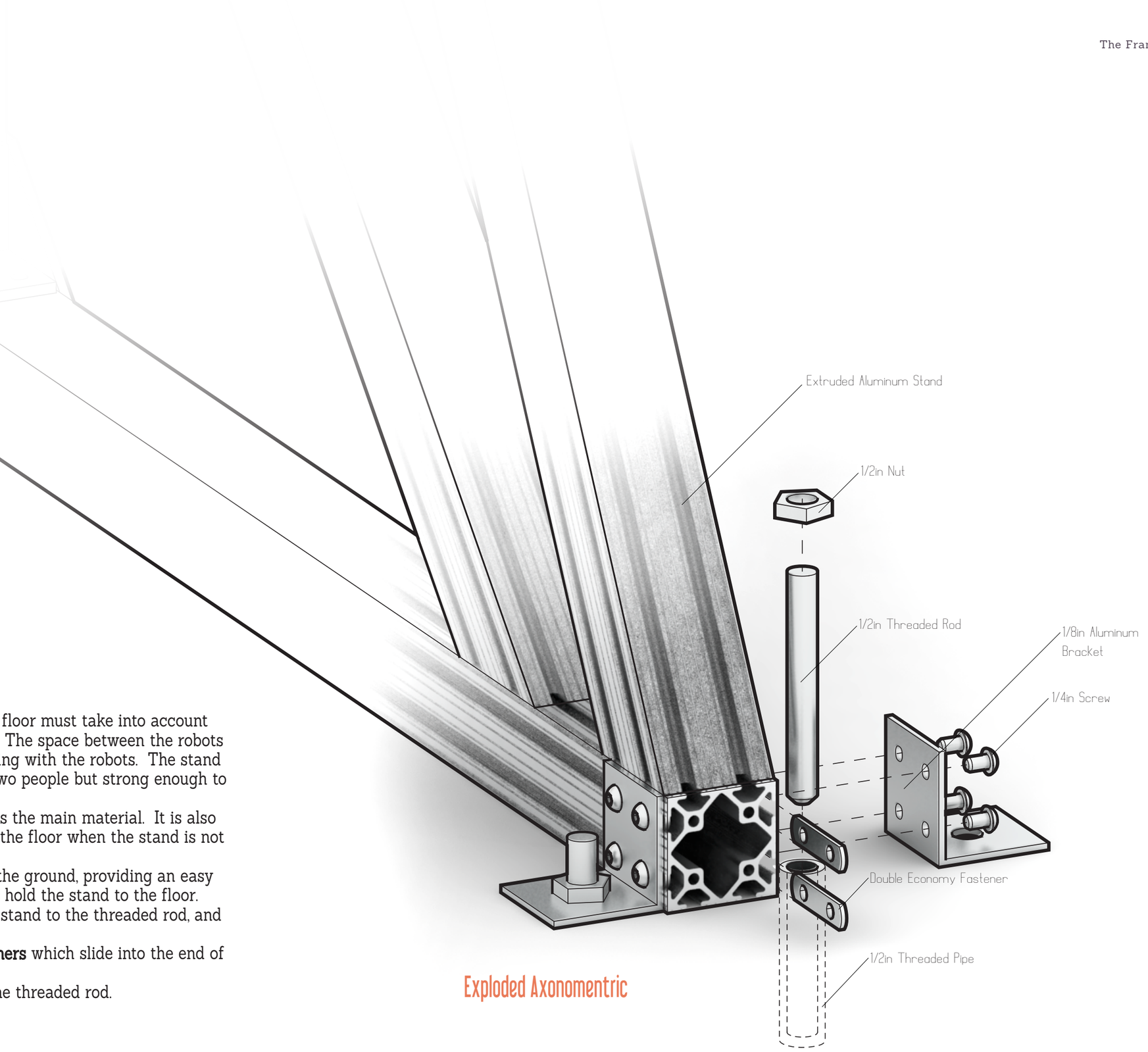
Secured Sheet

Exploded Axonomentric



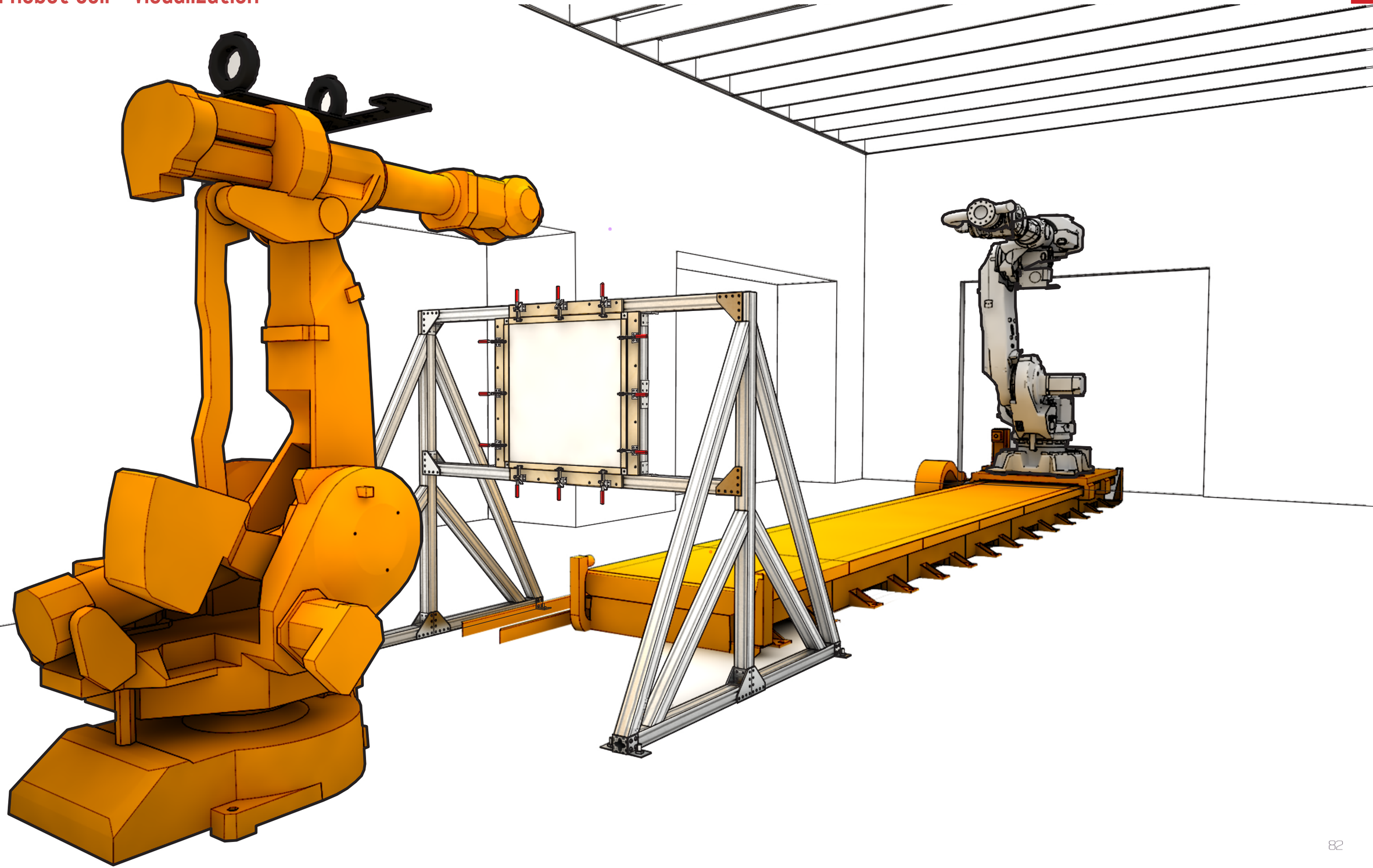
Floor Attachment Detail

The design of the attachment of the stand to the floor must take into account the need to move the stand when it is not in use. The space between the robots is valuable and important to other students working with the robots. The stand must be light enough to be movable by at most two people but strong enough to resist the forming forces.
For this reason **extruded aluminum** was chosen as the main material. It is also important to ensure that nothing protrudes from the floor when the stand is not in place.
To ensure this, a **threaded pipe** will be sunk into the ground, providing an easy way to insert and remove a **1/2in threaded rod** to hold the stand to the floor.
An **1/8in aluminum bracket** is used to attach the stand to the threaded rod, and is held to the stand by 4 **1/4in screws**.
These screws screw into 2 **double economy fasteners** which slide into the end of the stand.
To top it all off, a **1/2in nut** is tightened around the threaded rod.



Exploded Axonomentric

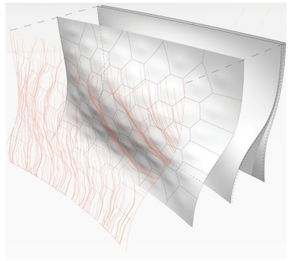
Frame in Robot Cell - Visualization

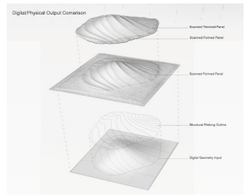


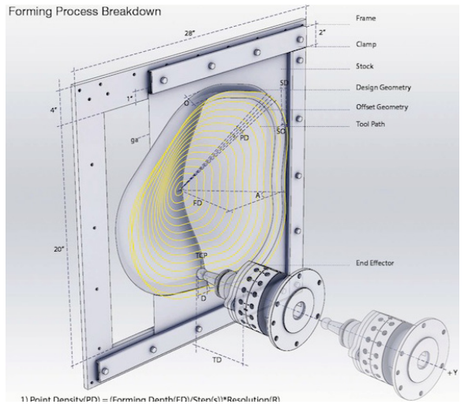
Precedents

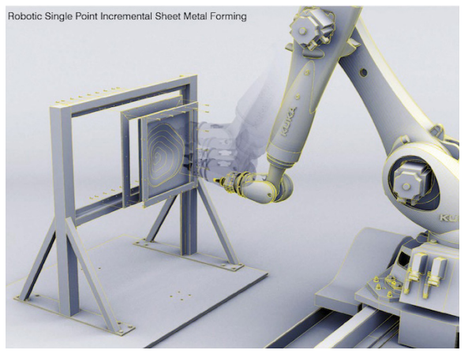
per-Forming - Incremental Sheet Formed Cladding

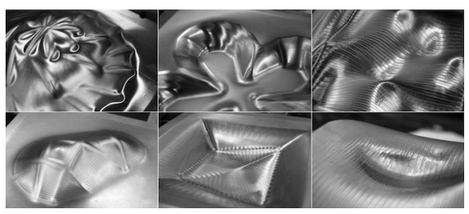
by Jake Newsume at the University of Michigan, 2013











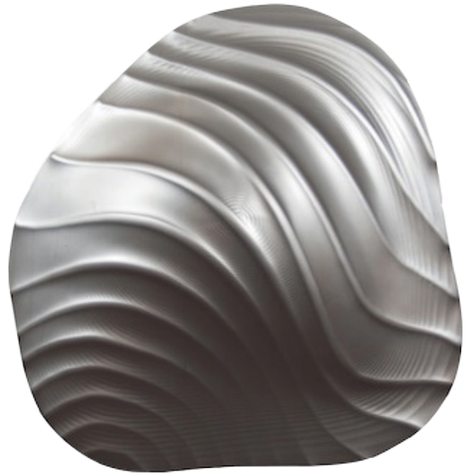
Submission for Tex-Fab's Skin Competition

Image credit: (Newsumme)

No Ribbing

Diagonal Ribbing

Edge to Center Ribbing



Global Linear Ribbing

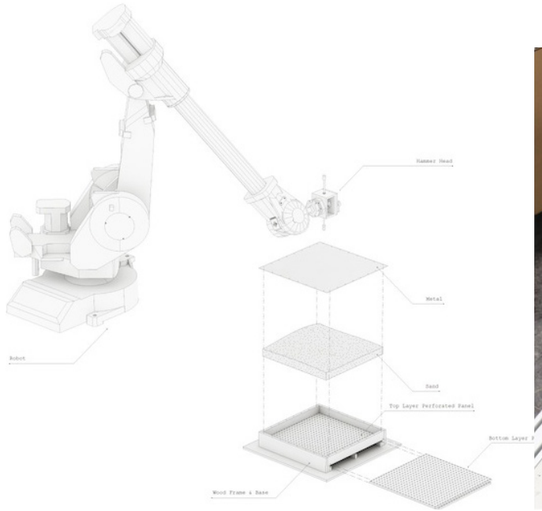
Ribbing Typologies

Image credit: (Newsumme)

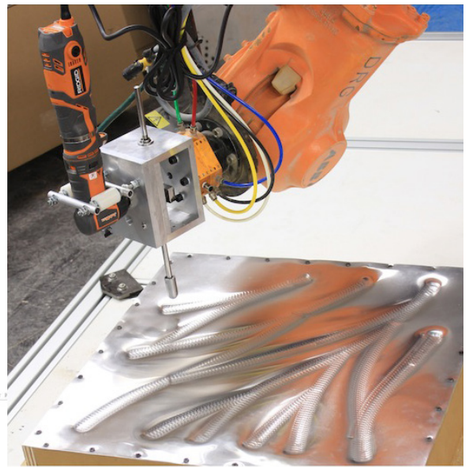
"Throughout the development of the physical and digital tools for this process, feedback has constantly been taken and given between design and fabrication. These parameters are being used in the design to influence the metal skin's global curvatures, local subdivision, and surface articulations to increase forming accuracy. The metal formed panels have been analyzed using 3D scanning technologies to understand where the structural ribs are needed for stabilization. The formed ribs are used as dynamic corrugations across the aggregation which makes a structured skin that identifies panel location." (Newsumme)

Robot Assisted Sheet Metal Shaping - Hammer-Formed Cladding

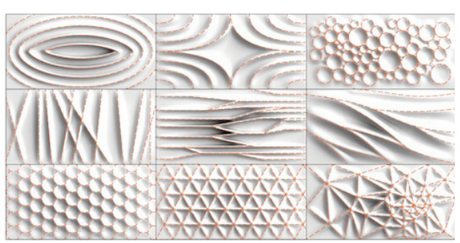
by Lik Hang Gu at the Harvard GSD, 2013



Robot working senario



Scotch York Mechanism for the Hammer



Patterns

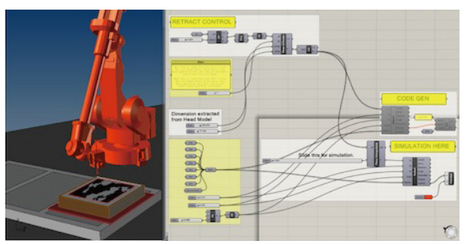
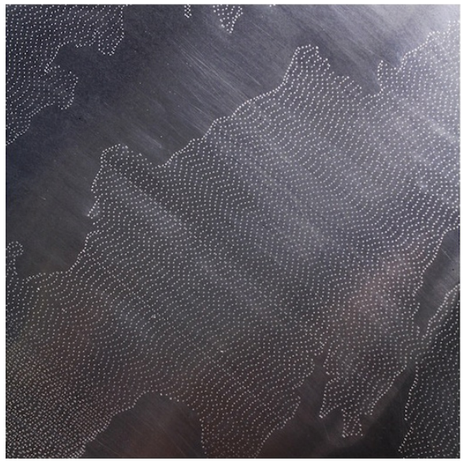


Image credit: (Gu)

Submission for Tex-Fab's Skin Competition

Pattern 1

Pattern 2



Pattern 3

Pattern 4

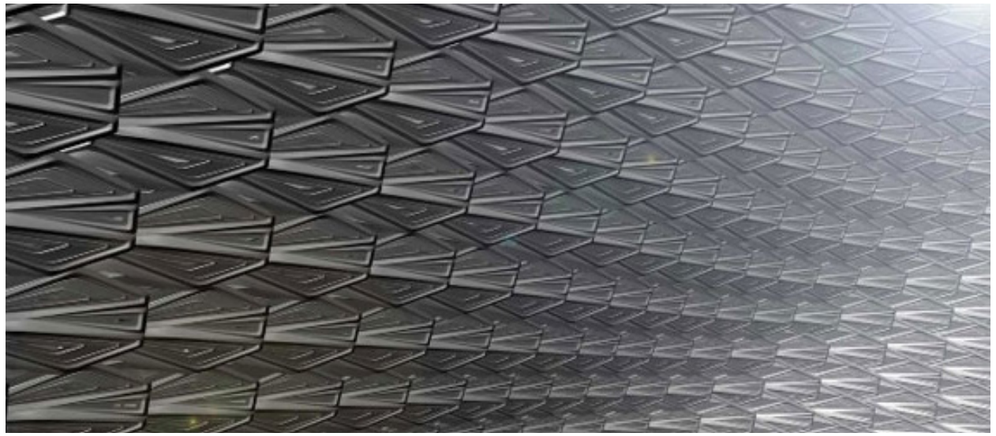
Image credit: (Gu)

Patterning Typologies

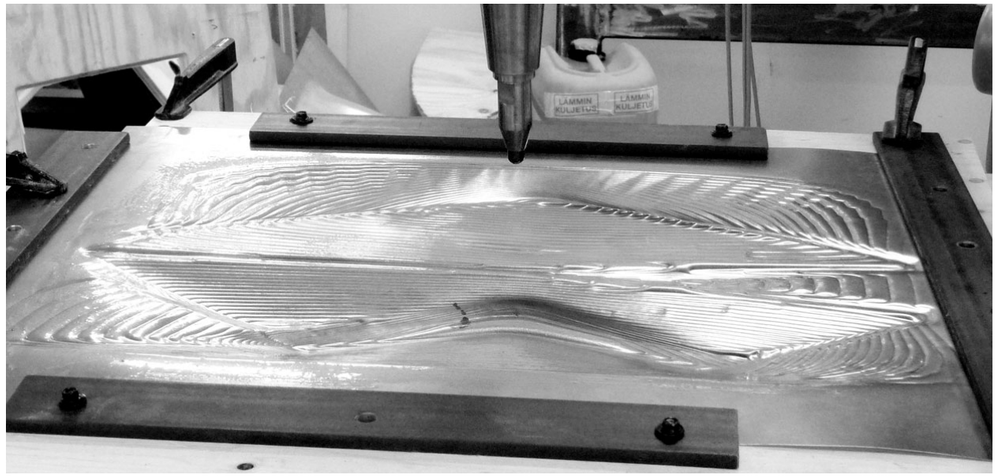
The hammering of sheet metal is much the the process of pointilism in drawing. One by one a robotic hammer punches the sheet at inputed points. There are many variations of this method which lead to decidedly different tacticle qualities. The result looks like a stiched cusion and the tool-path creates an ornamental aspect to each piece. "The sand base is designed to be able to adjust the hollow space under the sheet metal in order to allow further punching after sands are fully packed. Pattern followed by the curves generated in modeling software and translated into RAPID code for ABB robotic arm." (Gu)

Responsive Skin - Incremental Sheet Formed Molds

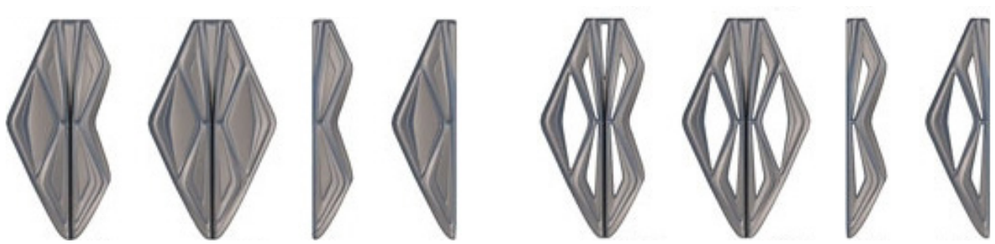
by Brian Cadiz, Gabriel Huerta, and Joseph Mathias at UCLA and Aalto University



Visualization



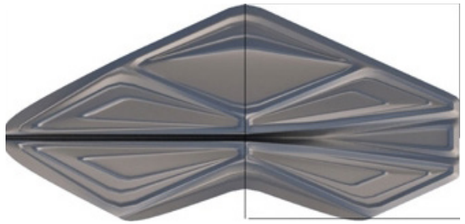
Fabrication



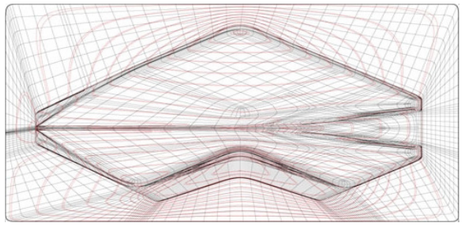
Shingle Typologies

Image credit: (Cadiz et al)

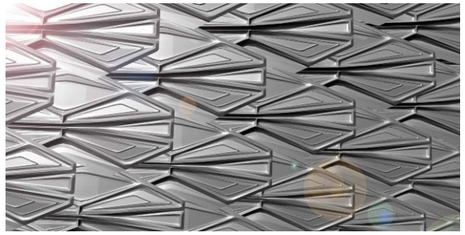
Shingles



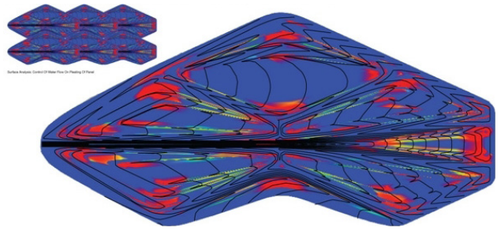
Shingle



Tool-path



Overlapping Shingles



Pleating based on stiffness and environment

Image credit: (Cadiz et al)

Shingles Parameters

"The lack of cost-efficiant manufacturing techniques has been an obstacle for avant-garde architecture. Digital design and production is changing that, and increasing the potential to realize more avant-garde architectural designs for today's needs. The project looks at the digital fabrication process of incremental sheet forming and its potential architectural application in the development of responsive building skins. The process, developed by the Aalto University Department of Material Technology, utilizes an industrial robot to form a sheet of metal against a computer guided piston-field as a means to create a low-cost reusable mould. The prototype shingle unit is designed within the constraints of the fabrication process and the maximization of the material properties of the recycled paper composite, UPM Profi, that it is to be injected molded from. Pleating on the shingle panel provides performative ornament that allows for both material stiffness and environmental performance. The ornamental patterning of the pleats, inspired by ancient armor, is contoured to control the flow of water from panel to panel. The variable thickness of the pleating further allows for a reduction in the overall panel thickness and weight by concentrating material in critical zones. Versatility is embedded into the form of the unit through the shaped grooves that allow for the seamless interconnection between shingles in the panelization of a rainscreen. Through the development of a series of scripts, the panelization of the unit was biased to be responsive to the environment and the minimization of unique pieces. Porosity and surface coverage of the building skin was optimized with data inputs from solar analysis and surface curvature." (Cadiz et al)

Folded Plate Structures - Two Point Incremental Formed Panels

by Univ.-Prof. Dr.-Ing Martin Trautz, Dipl.-Ing. Ralf Herkrath at RWTH Aachen University, 2009

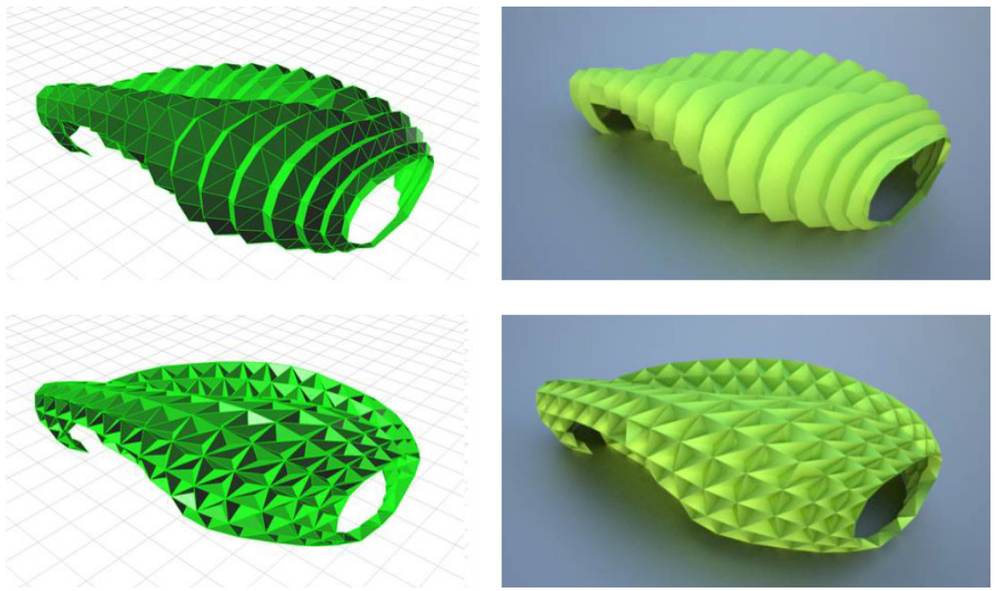


Image crecit: (Trautz and Herkath, 10)

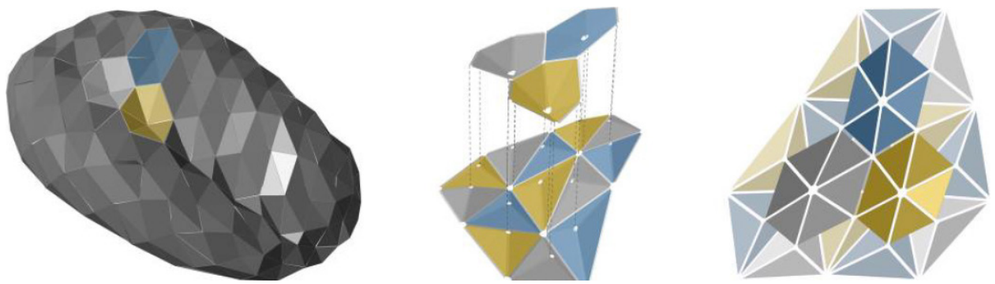


Image crecit: (Trautz and Herkath, 10)

Double-Layered Facet-Like Folding Structure

The pre-elementing happens according to the chosen folding structure (longitudinal or facet-like). Metal pyramids with a hexagonal base area on the outer and a triangular base area on the inner site evolve from a double-layered facet-like folding. The softwaretool processes the individual metal pyramids for the CNC controlled sheet metal forming with the help of software that translates the geometry of the free-form into the data to control the CAM production (computer aided manufacturing). All different parts are labeled to assure a correct typological erection." (Trautz and Herkath, 10)

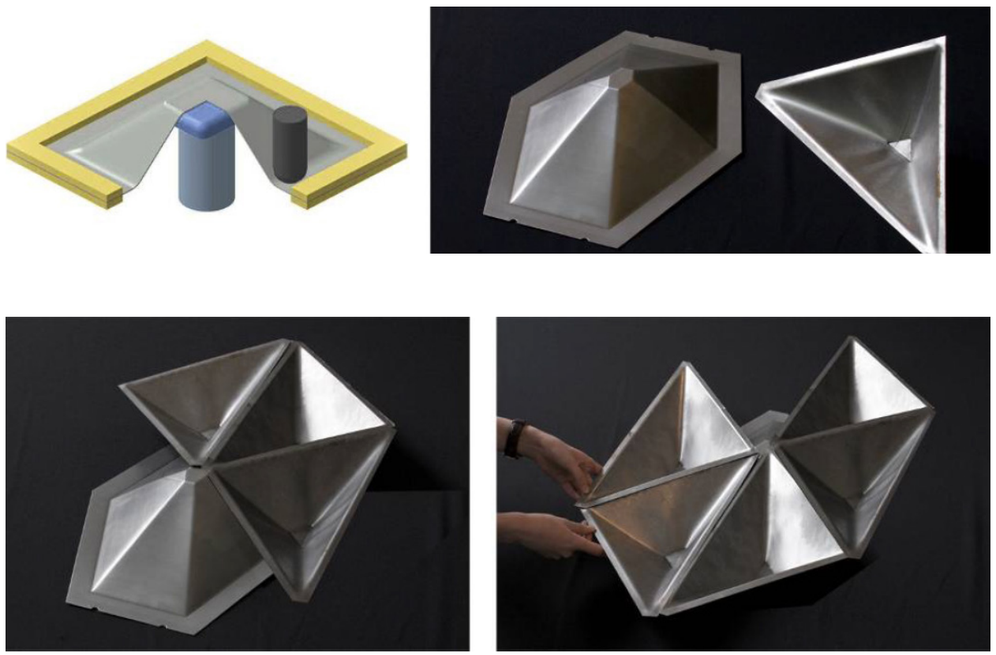


Image crecit: (Trautz and Herkath, 12)

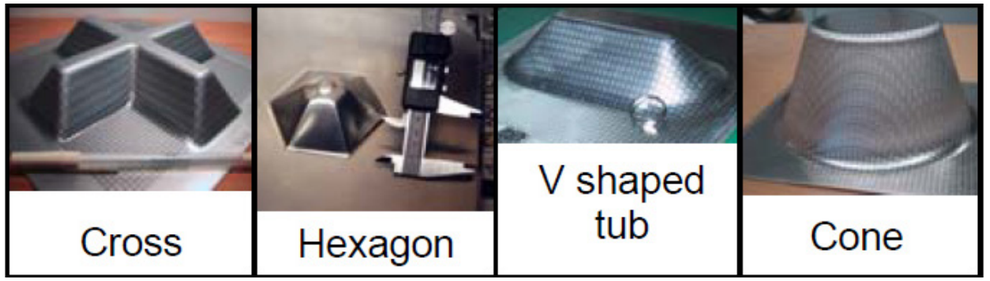
Assembly Process

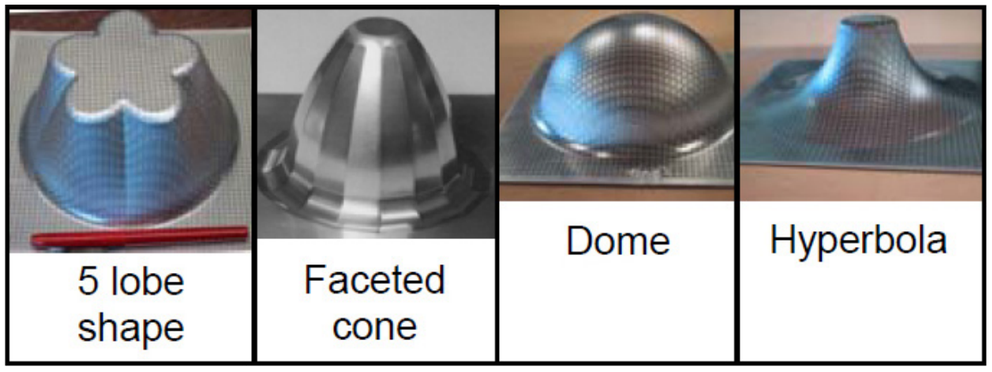
"The different metal pyramids are assembled to buildings elements which are to be erected on site. The assembling of the elements is guaranteed by overlapping of the different metal sheets." (Trautz and Herkath, 11)

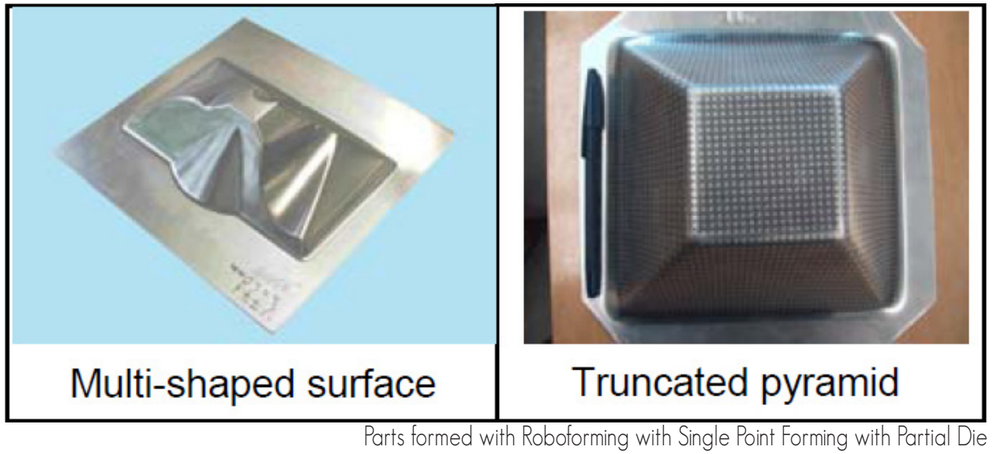
"The principle of folding structures is an established principle of construction in nature whose potential has rarely been used in architecture. Based on folding structures, high-stressed and wide-spanning light weight structures can be realised. More than the complex requirements for their constructive detailing, the limited possibilities to describe them geometrically with mathematical geometrical functions have constraint their realisation. Numerical digital methods annul this limitation by making the tessellation and triangulation of arbitrary shapes possible. Due to this method, almost any folding structure can be constructed (Trautz [7]).
After the development of the software tool and after the first successful shaping, the research has now the aim to approach the application of folding structures in context of the sustainable use of material and to gain new impulses for this principle of structural shaping. Additionally, new ideas of the building process for free-forms should be generated which could also serve as helpful suggestions for other engineering disciplines." (Trautz and Herkath, 12)

Examples - Single Point Incremental Formed Parts

With Partial Die







Parts formed with Roboforming with Single Point Forming with Partial Die

Image crecit: (Jeswiet, SPIF, 20)

Comparison of SPIF vs Roboforming with Local Support

Part formed with Single Point Incremental Forming

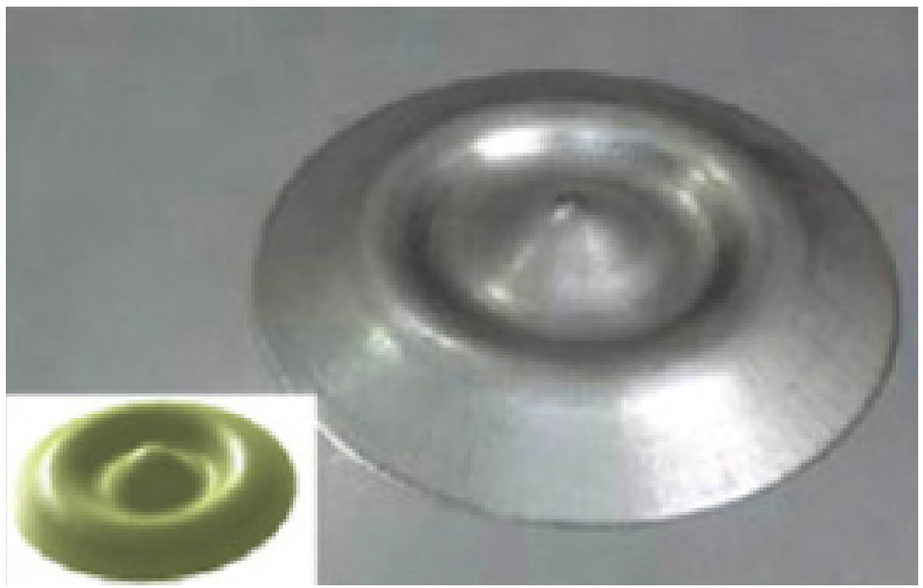
Part formed with Roboforming with Local Support

Image crecit: (Buff et al., Accuracy, 153)

Examples - Roboformed Parts - Local Support

Concave and Convex

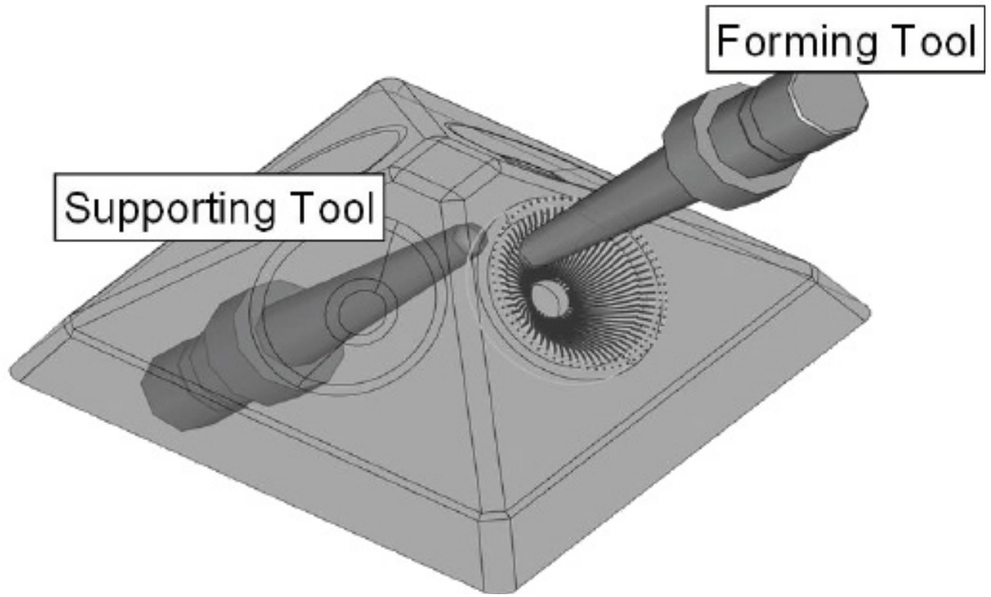
Parallel Forming



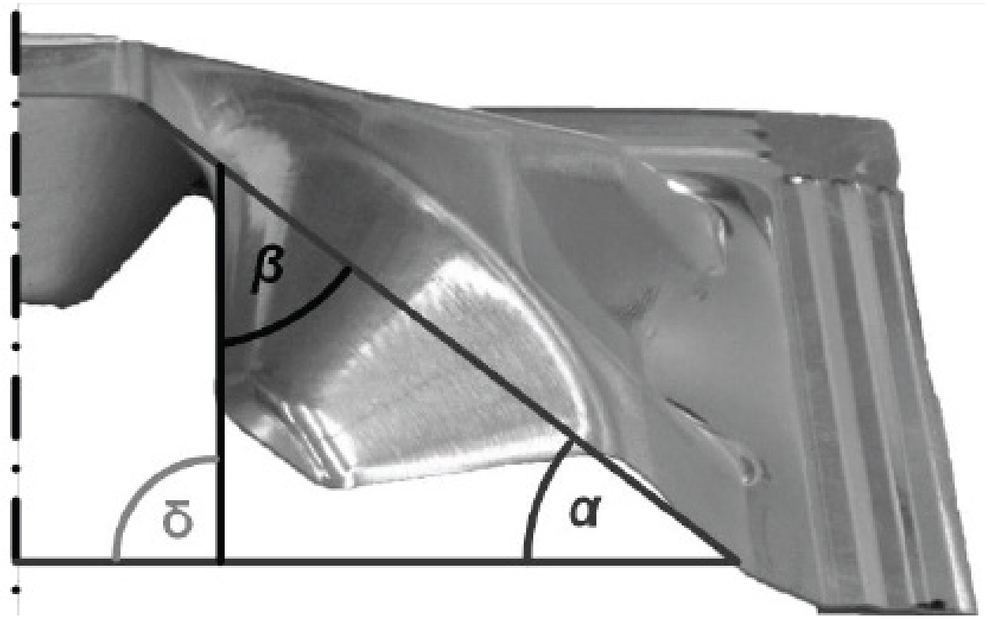
Sequential Forming

Image crecit: (Malhotra , Accumulative-DSIF, 253)

Subsequent Forming



Support Tool acts as Peripheral Support



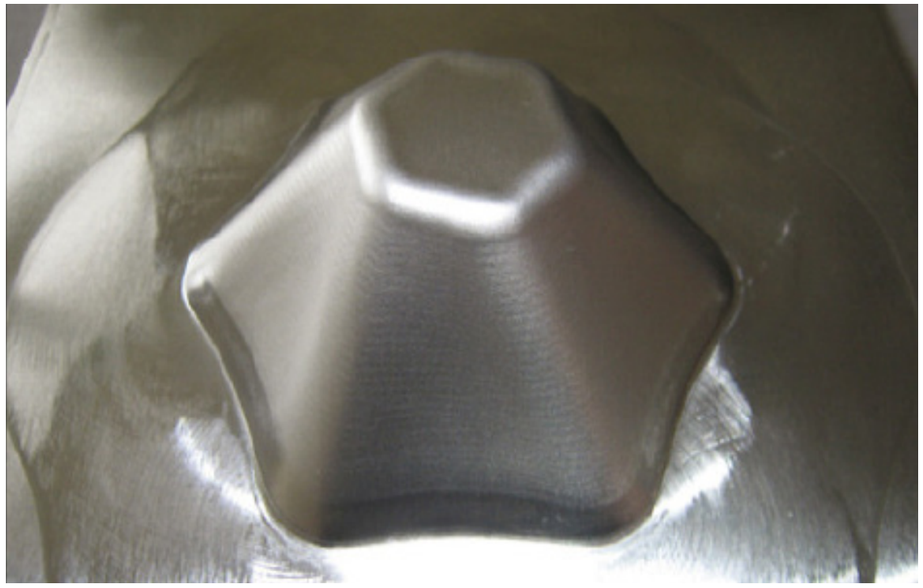
Maximum Draw Angle for Subsequent Forming

Image crecit: (Buff et al., Accuracy, 154)

Examples - Roboformed Parts - Peripheral Support

Peripheral Support Pieces

Wierd Bean Thing



Twisted Hexigon

Image crecit: (Meier et al., Tool Path, 149)

Comparison of Formability - Peripheral Support vs Local Support

Point of Failure with Peripheral Support

Point of Failure with Local Support

Image crecit: (Meier et al., DPIF, 330)

Examples - Roboformed Parts - Multiple Forming

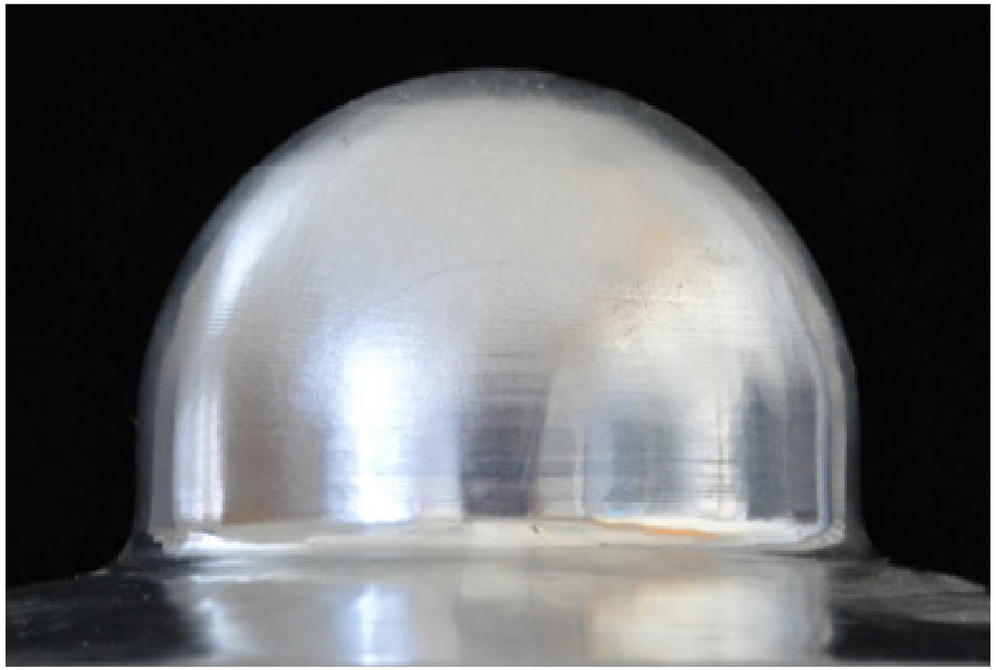
Multi-Pass Forming

Complex Automotive Part

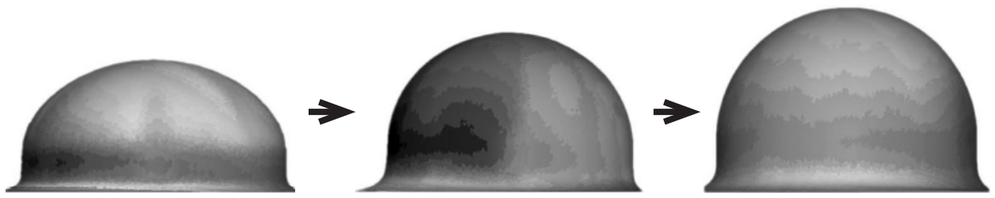
Cylinder with Undercut (97°)

Image crecit: (Meier et al., DPIF, 328)

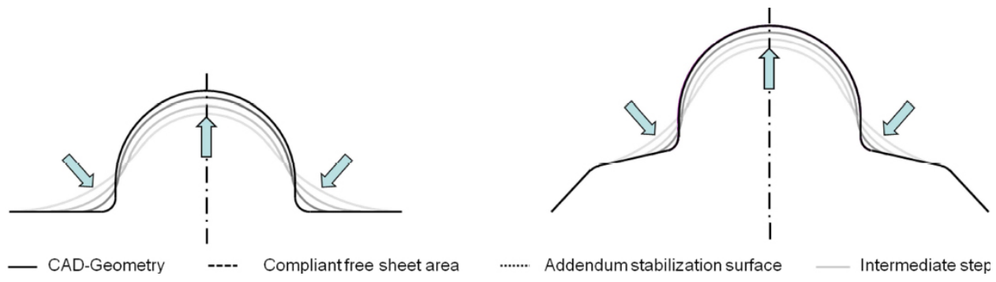
Multiple Forming with Stabilization Surface



Hemisphere



3 Step Forming Process

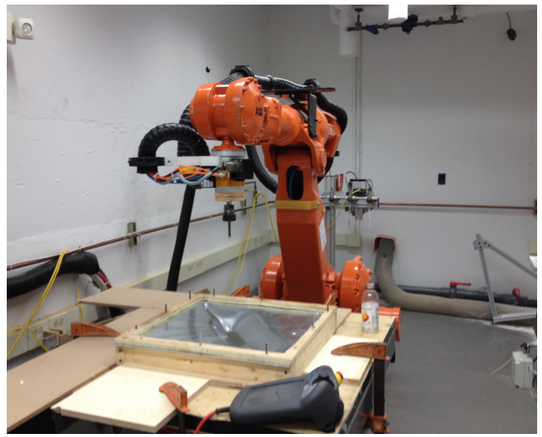


Stabilization Surface

Image crecit: (Kreimeier, Accuracy, 859)

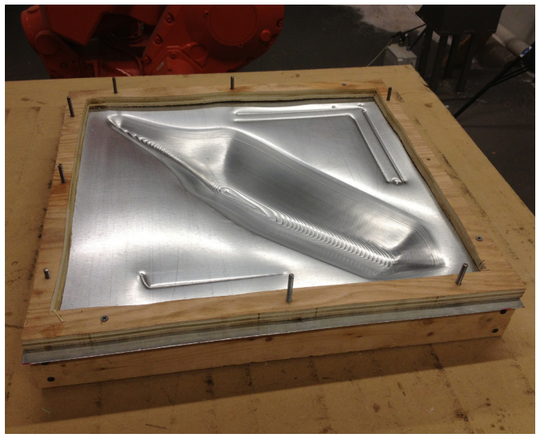
Incremental Sheet Formed Prototype - Process

By Alex Fischer and Matt Adler at Carnegie Mellon University, 2013



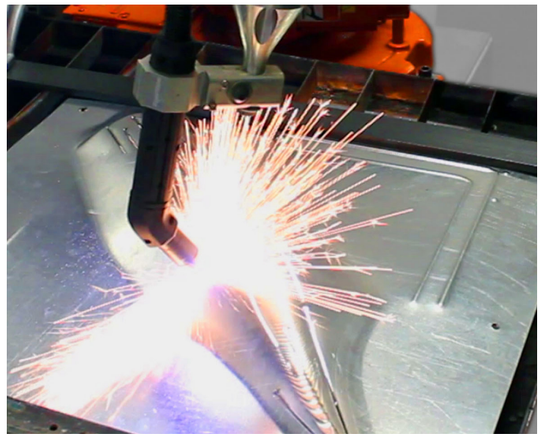
Incremental Sheet Forming

Step 1



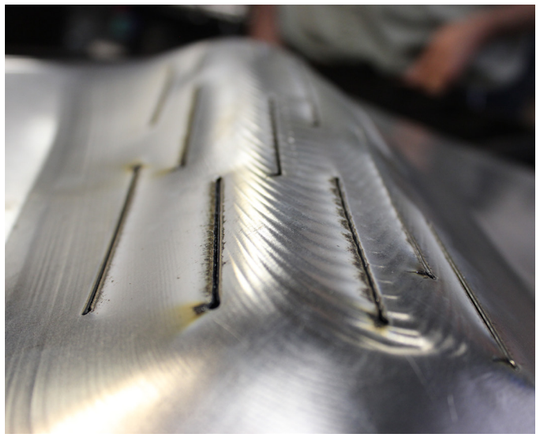
Incremental Sheet Forming - Result

Step 2



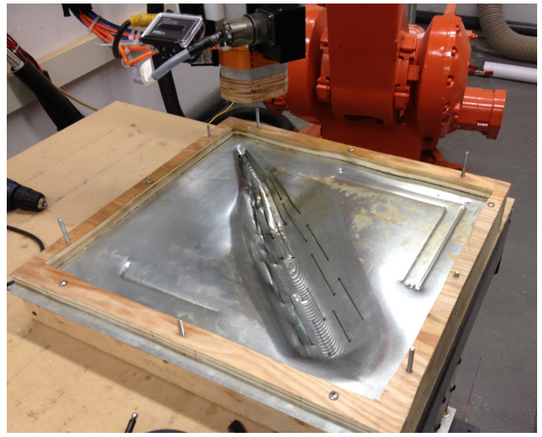
3D Plasma Cutting

Step 3



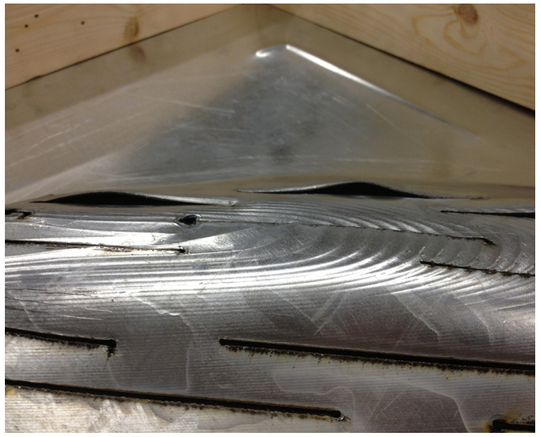
3D Plasma Cutting - Result

Step 4



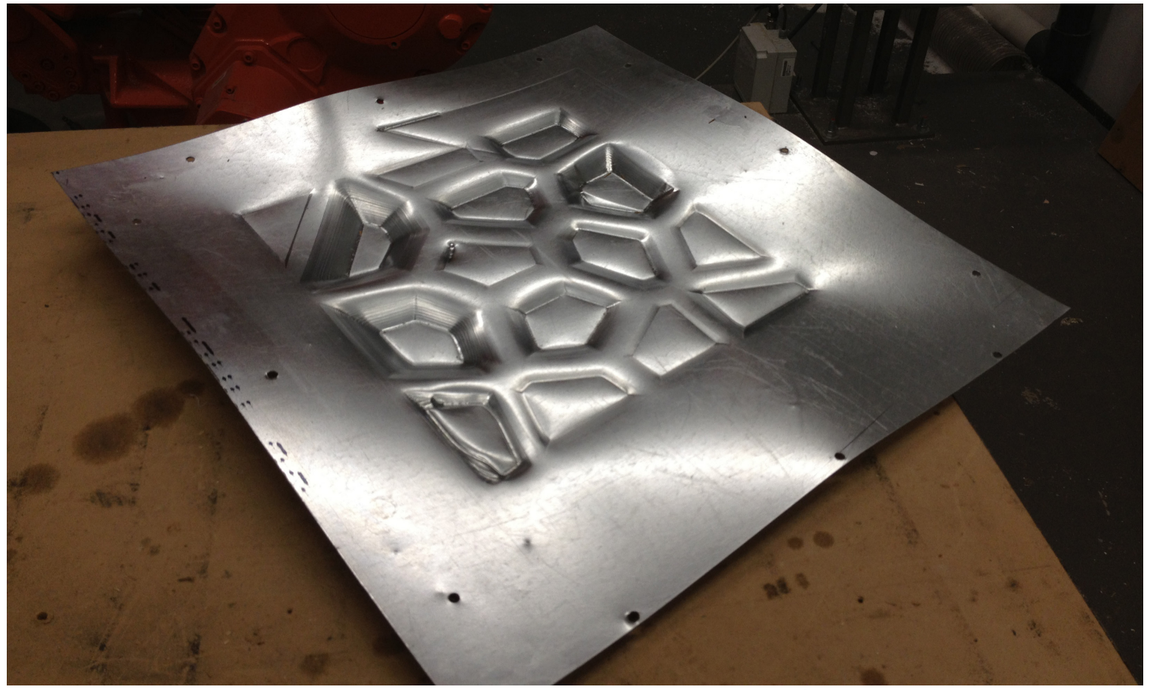
2nd Incremental Sheet Forming

Step 5



2nd Incremental Sheet Forming - Result

Step 6



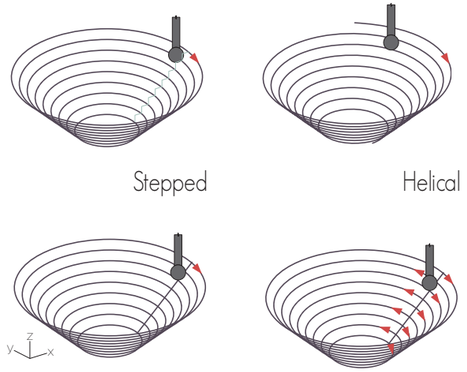
Local Deformations made without backing plate

Incremental Sheet Formed Prototype - Video

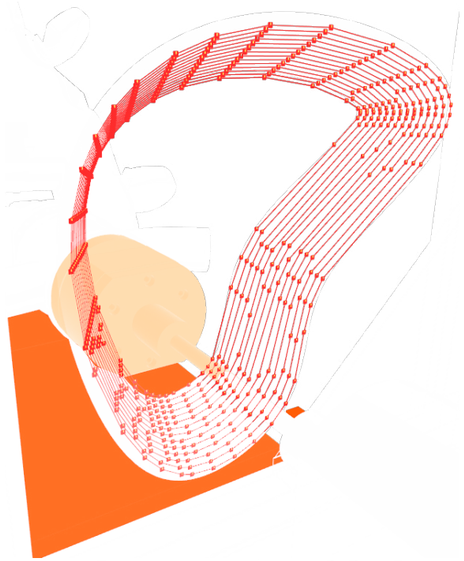
By Alex Fischer and Matt Adler at Carnegie Mellon University, 2013

Experiments

Tool-path Variation

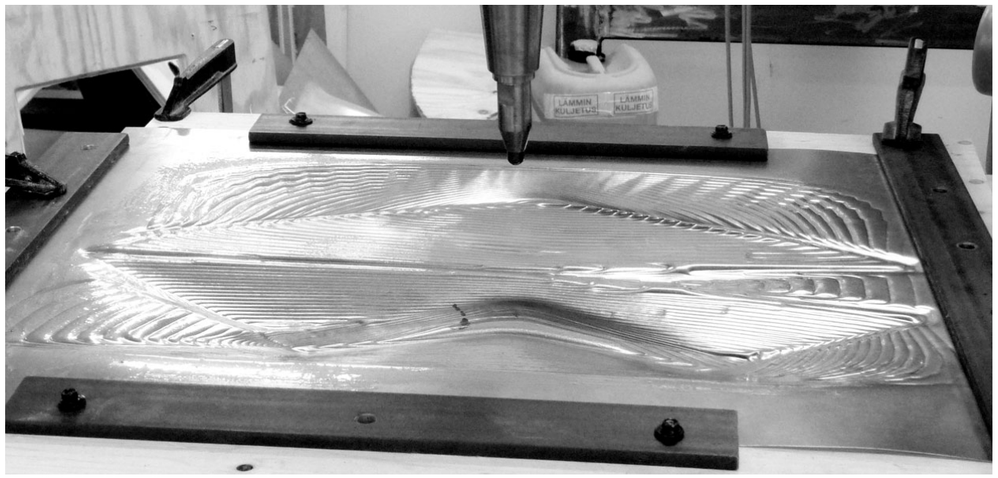


Custom Toolpath Types

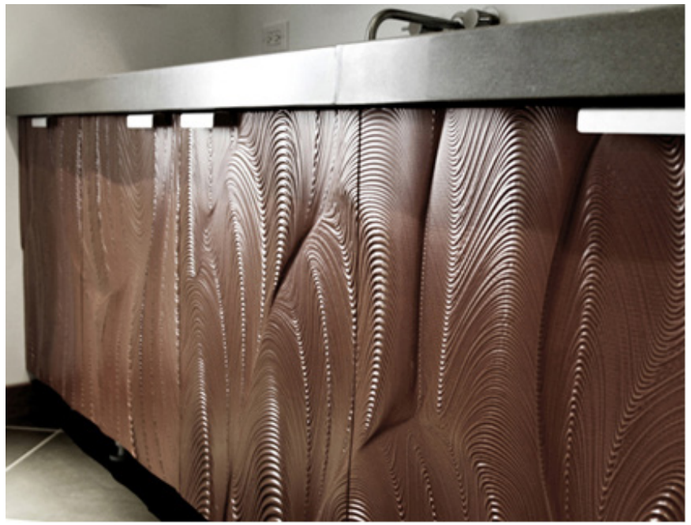


Toolpath resulting from divByDeltaTan

divByDeltaTan causes irregular spaced targets which influences the appearance of the formed part.

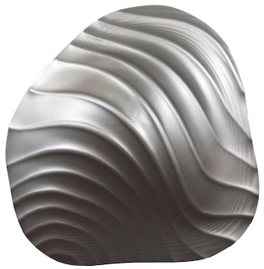


Fabrication
Image credit: (Cadiz et al)



Path Responsive Surface Milling - Leaves Style
Image credit: (Skylar Tibbits)

Edge to Center Ribbing



Global Linear Ribbing
Image credit: (Newsumme)

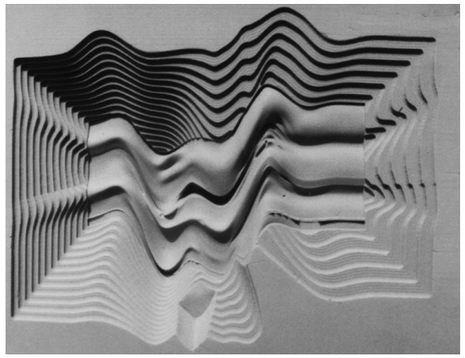


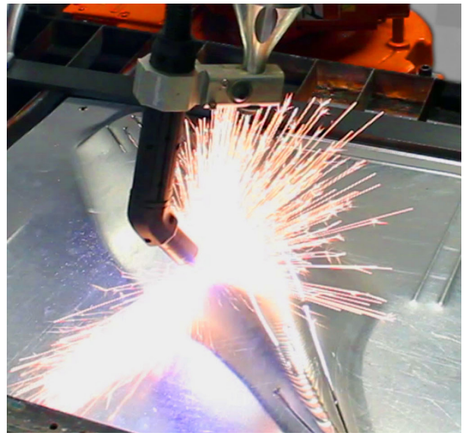
Figure Ground Mill File
Image credit: (subdv.com)

Ornamental Tracer Ribs
Image credit: (Newsumme)

Path Responsive Surface Milling - Cloth Style
Image credit: (Skylar Tibbits)

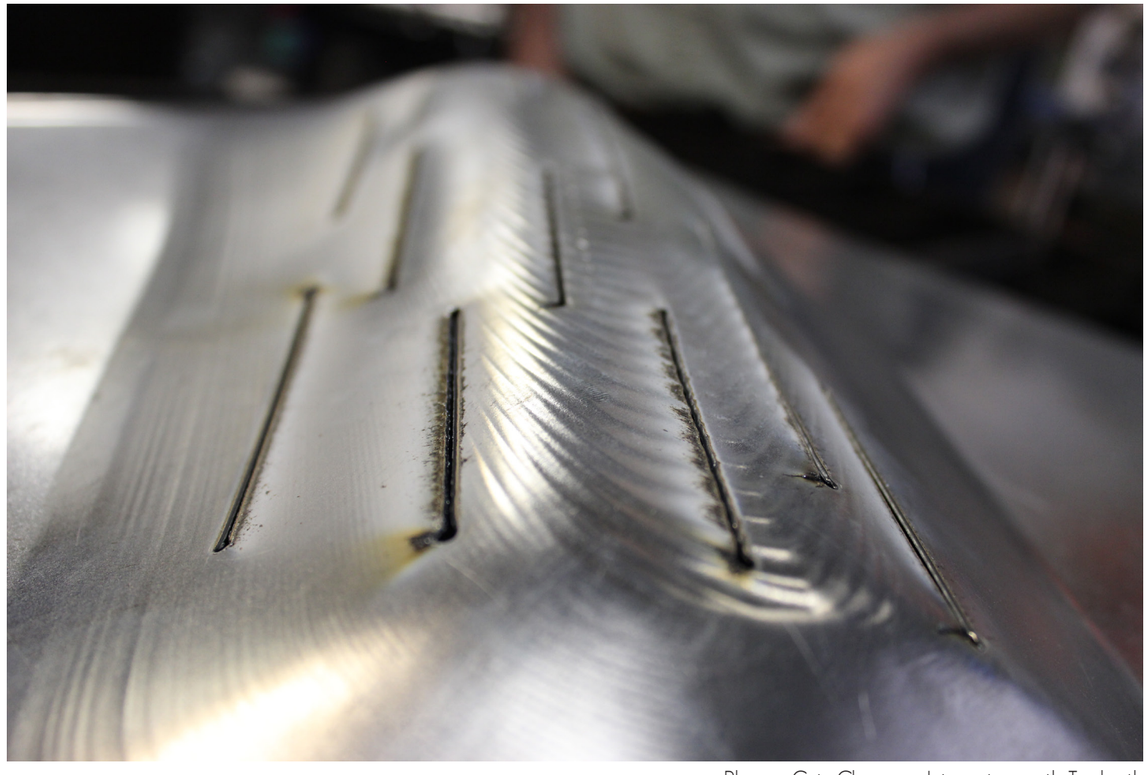
3D Plasma Cutting of Formed Parts

Prototype by Alex Fischer and Matt Adler, Carnegie Mellon University, 2013



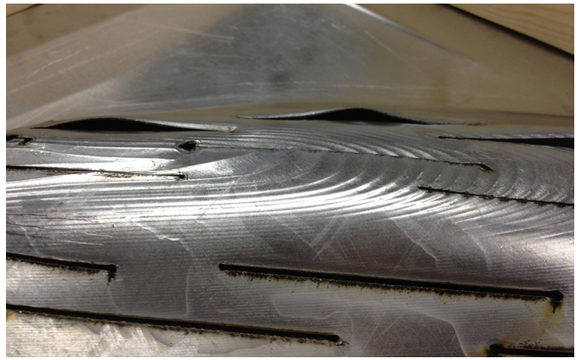
3D Plasma Cutting - Arc Firing

3D Plasma Cutting - Calibration

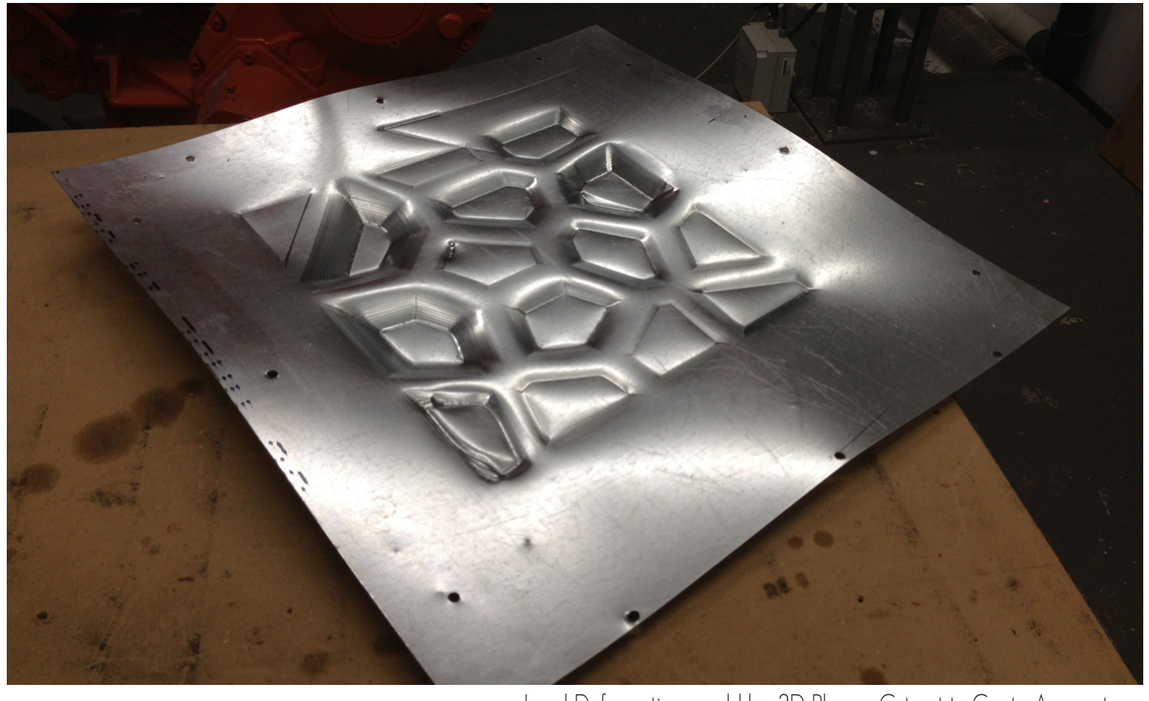


Plasma Cuts Closeup - Interaction with Toolpath

Incrementally Form -> 3D Plasma Cut -> Incrementally Form



Roboformed Slit Apperatures



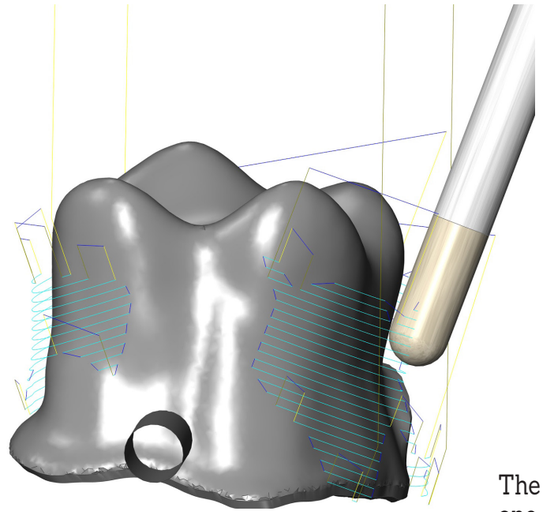
Local Deformations could be 3D Plasma Cut out to Ceate Apperatures

3D Plasma cutting of formed parts can be used to create appertures in panels. It would be interesting to see the result of multiple local deformations with their flat aread cut out. This could be used as a architectural cladding to control the passage of light across the facade

Undercuts via Multiple Forming

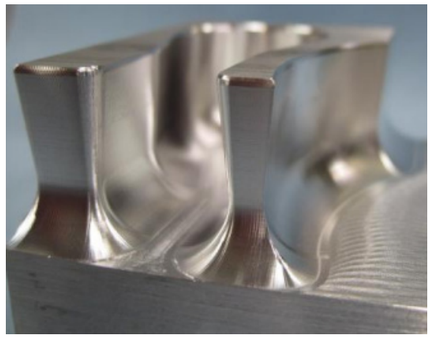
Cylinder with Undercut (97°)

Image crecit: (Meier et al., DPIF, 328)



Custom Dental Crown with Undercuts

Image crecit: moduleworks.com

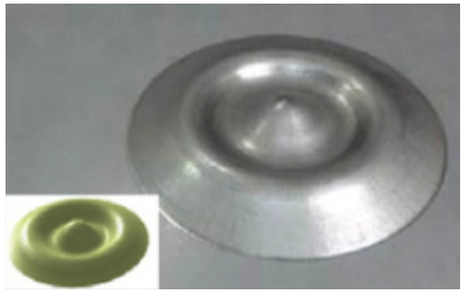


Undercuts made with a 5-axis Mill

Image crecit: onecnc.net/

The abilty to create undercut geometry is one of the unique benefits of Roboforming. It would be natural to find some architectural use for undercut panels. Interestingly, one proposed application of Microforming, essentially (Roboforming at a small scale) was bespoke dental crowns which have complex forms with undercut areas. (Jeswiet, SPIF, 23)

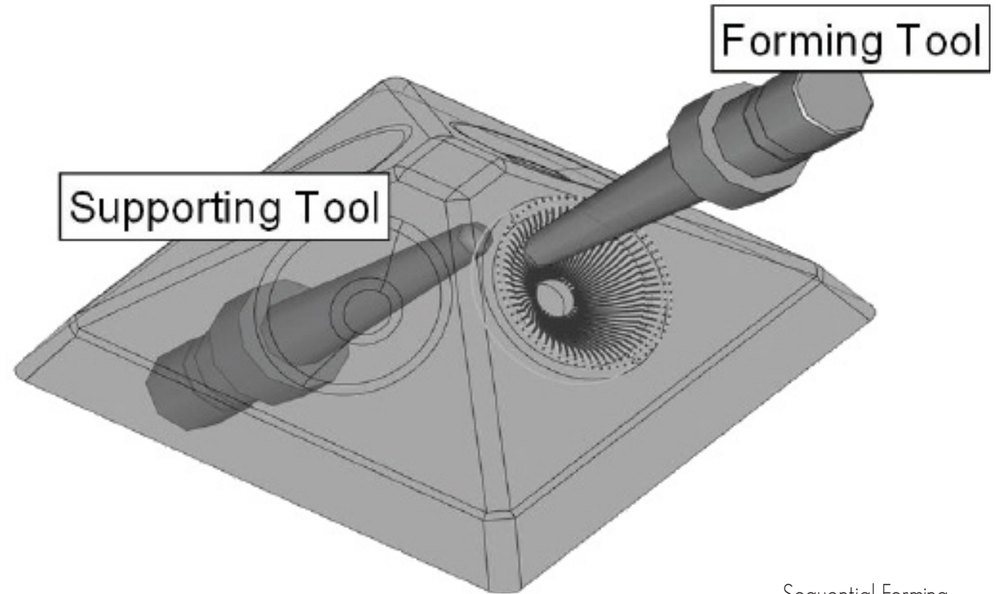
Concave and Convex Forms via Sequential and Parallel Forming



Sequential Forming

Parallel Forming

Image crecit: (Malhotra , Accumulative-DSIF, 253)



Sequential Forming

Image crecit: (Buff et al., Accuracy, 154)

An essential advantage of Roboforming is the ability to form on both sides of the sheet. By setting the robots to trade the roles of former and supporter in the middle of a forming process, it is possible to create concave and convex forms in a single run. Another method is to form a geometry and then form the result in select areas. In this case the support tool is physically a forming tool but behaves like a peripheral support tool, except in locations local to the forming tool.

Speculation

Design Based on Competition Breif

TEX-FAB's SKIN Competition

BRIEF

The building envelope represents the most complex and fundamentally linguistic element of architecture today. Its formal development and performative capacity – what may be called the *activated envelope* – is foundational to its purpose and presents a dialogue the building has with itself and that of its context. We can understand this relationship in many ways, but ultimately it is one that mimics our own skin. Fundamental to this is an explicit or implicit *adaptability* found in its performance – how it functions and meets the needs of the building. In the preceding 100 years since the beginning of the 21st century the transformation from a static, heavy and obfuscating series of load bearing walls, to its current role of a communicative envelope, dynamical and exploratory, sets the stage for this competition and in what we believe is the most important area of research in architecture. It is within this framework that the international digital fabrication competition SKIN asks designers and researchers to speculate, or if they so choose – to present existing research - on the role of the building envelope by exploring new methods to enable the performative and aesthetic qualities of a façade.

Design submissions may develop any context they choose, real or virtual, at any scale and on any building type so as to present a complete thesis. Integrating structure, dynamical cladding or other system whether static or active may be submitted. We encourage the boldest visions and challenging technologies in the development of your proposal.
The competition will select four of the most robust and intriguing projects, that best rethink the building envelope, supporting those selections through prototypes developed to illustrate the potential of the competition submission.

CONTEXT

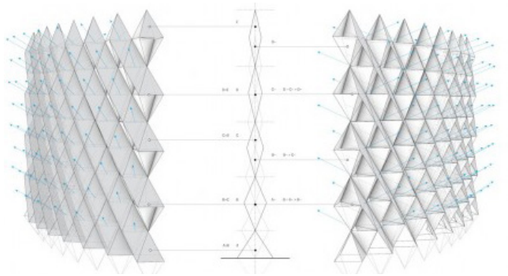
SKIN is a two-stage international design competition established to foster the deeper developments within the field of *computational fabrication*. We are soliciting design proposals that further existing research, by enabling prototyping at a larger scale or full scale, and proposals to jumpstart new research and design concepts into a first prototype. Choice of project location, contextual constraints, programmatic and functional requirements are open and should be freely interpreted to further the proposal's thesis.

MATERIALS AND FABRICATION

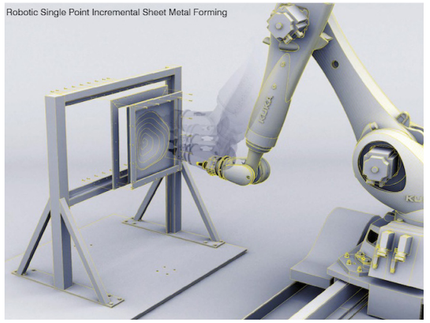
SKIN seeks proposals that specifically leverage the advancement of metal fabrication systems; however, we are open to hybridized material assemblies. As such it is not expected that metal is the only material system utilized in the design research, though we are seeking to collect and promote a concentrated body of work that can focus not only on metal application but also the methodology. Note, you must propose a specific fabrication method based on your research, thus the competition entry needs to specify techniques and materials.

SKIN Competion - Ended October 25, 2013

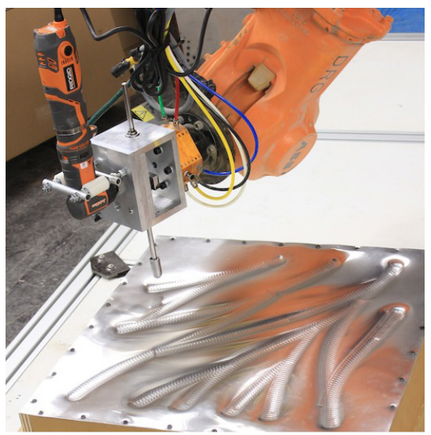
Entries

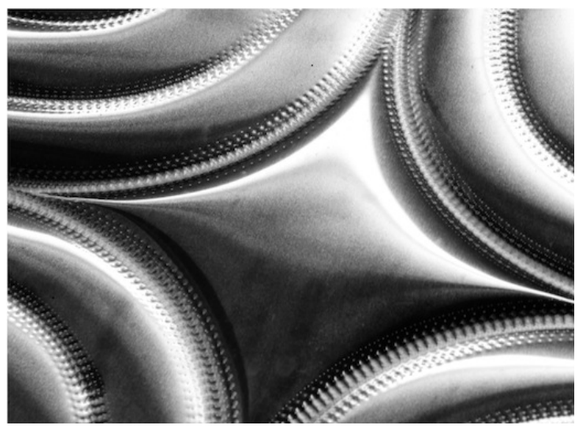


3xLP by Christopher Romano and Nicholas Bruscia



per-Forming by Jake Newsum

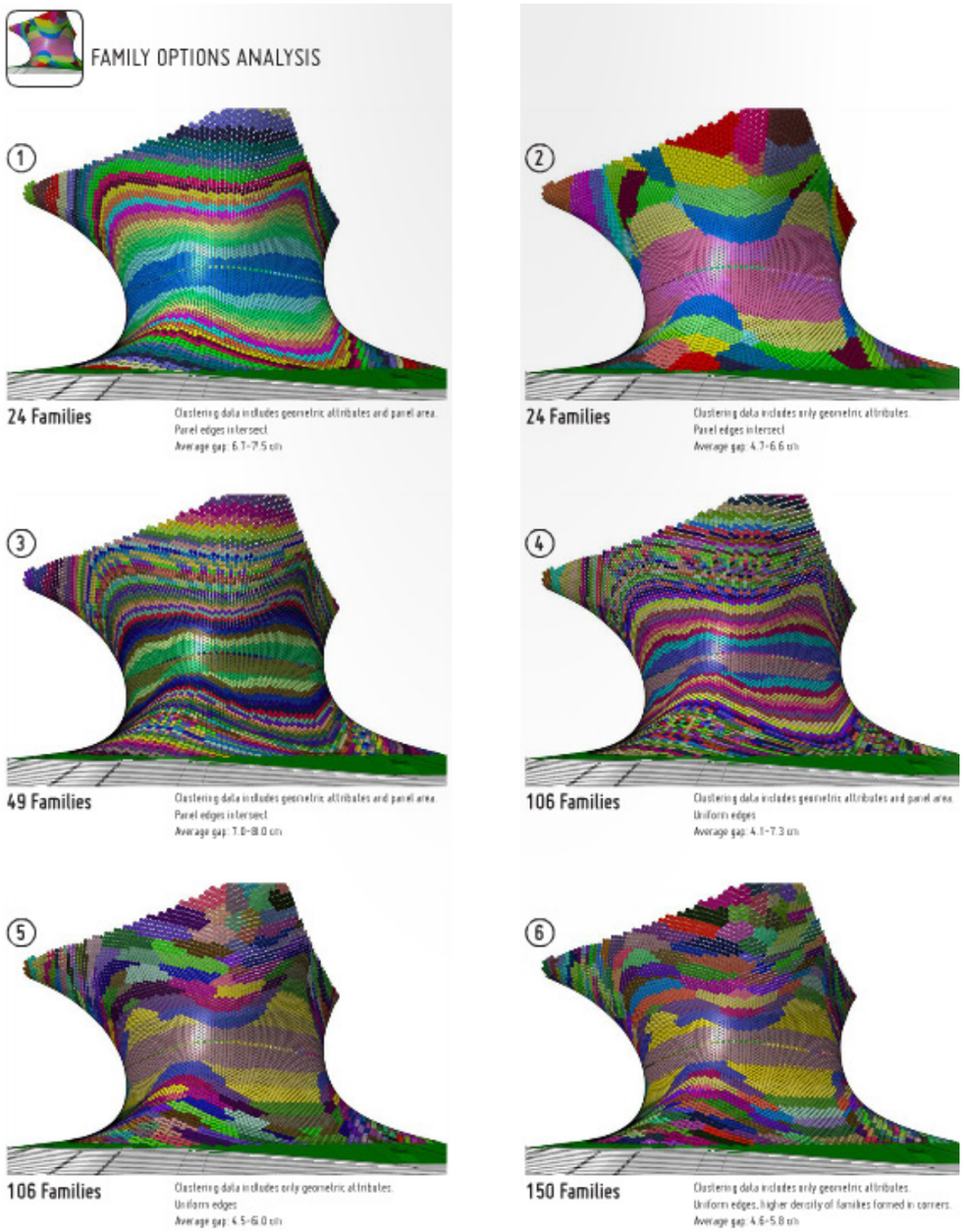


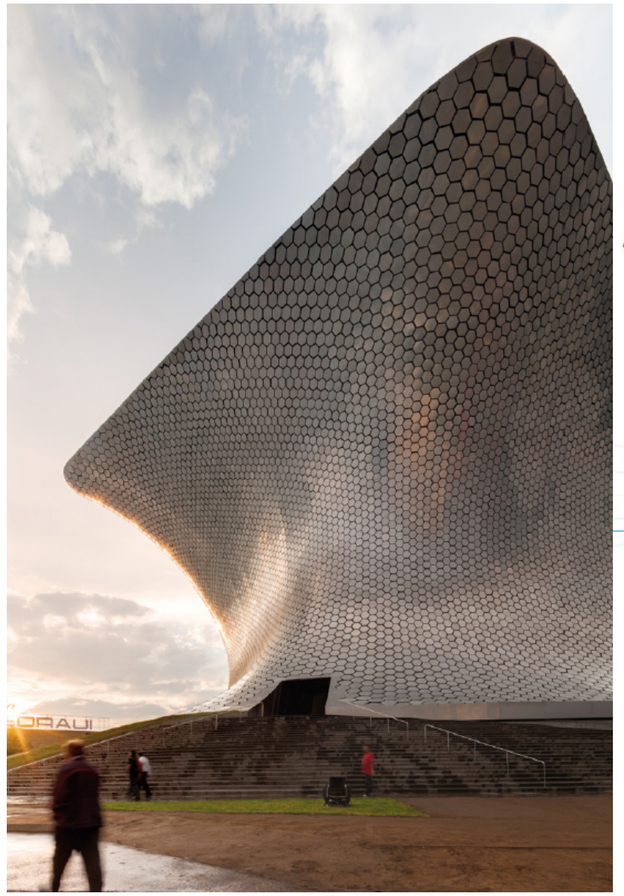


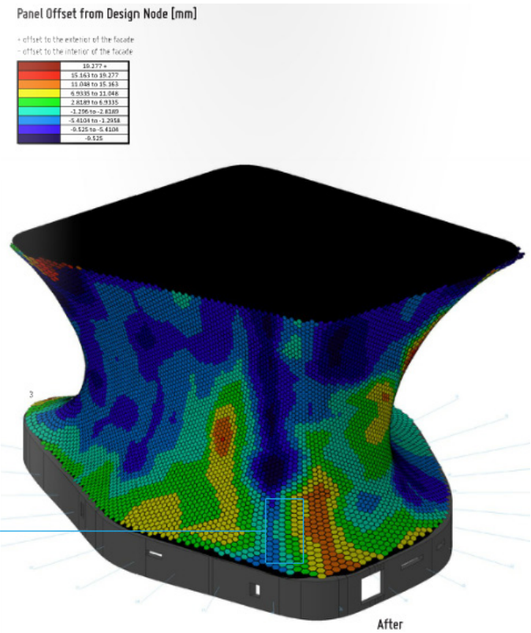
Hammer Forming by Lik Hang Gu

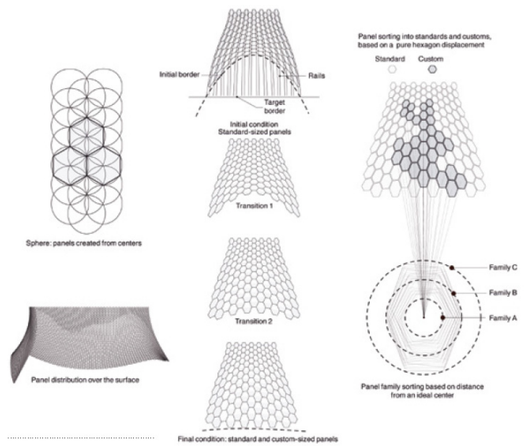
Design Based on Case Study of Existing Facade

Museo Soumaya by Free Architects









Design Based on Case Study of Existing Facade

Dongdaemun Design Plaza and Park by Zaha Hadid Architects

Image courtesy: failedarchitecture.com

Image courtesy: arabianindustry.com

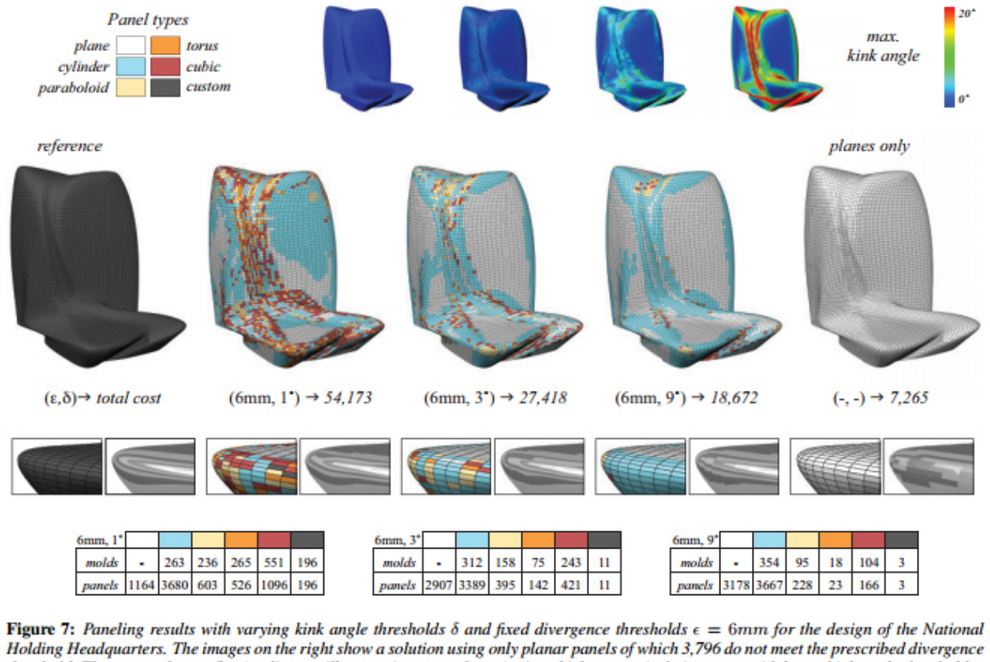


Figure 7: *Paneling results with varying kink angle thresholds δ and fixed divergence thresholds $\epsilon = 6mm$ for the design of the National Holding Headquarters. The images on the right show a solution using only planar panels of which 3,796 do not meet the prescribed divergence threshold. The zooms show reflection lines to illustrate inter-panel continuity which successively improves with lower kink angle thresholds.*

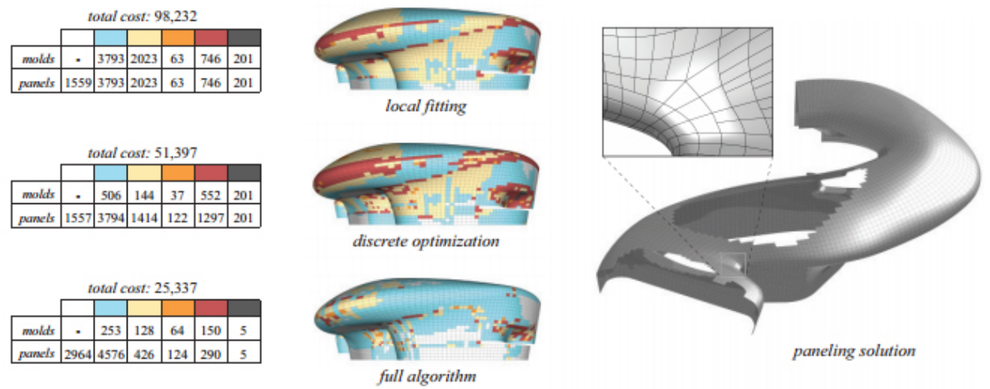
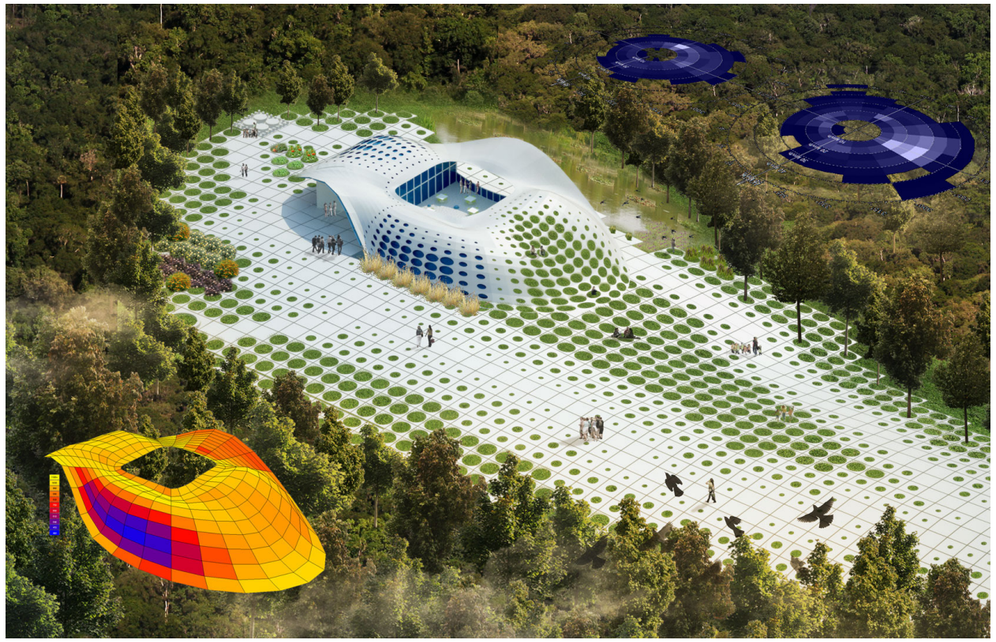


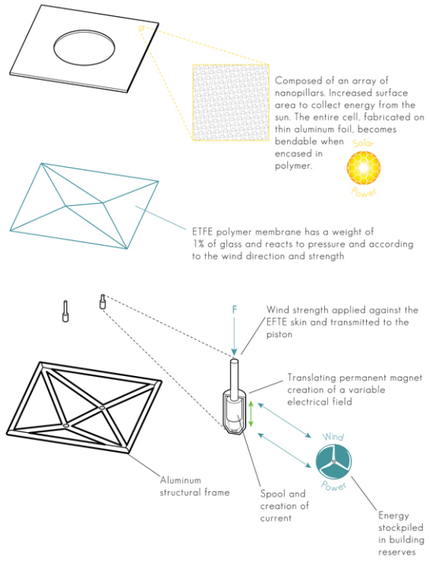
Figure 8: *Comparison of different methods for the same quality thresholds. State-of-the-art commercial tools only support a greedy panel assignment based on local fitting (top). Just one single application of our discrete optimization greatly reduces cost without loss in surface quality (middle). The full paneling algorithm interleaving discrete optimization with global continuous registration produces a high quality paneling (bottom). This solution contains 90% single curved panels and a very small number of custom molds, leading to a significantly reduced cost compared to greedy and local methods. The zoom on the right shows that our algorithm supports arbitrary curve network topology, including t-junctions. (Zaha Hadid Architects, Dongdaemun Design Plaza and Park, Seoul.)*

(Eigensatz, Paneling Freeform Surfaces, 1)

Design Based on Performative Quality

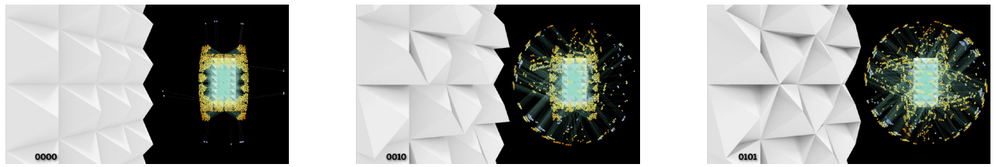
Environmental and Programattical Factors

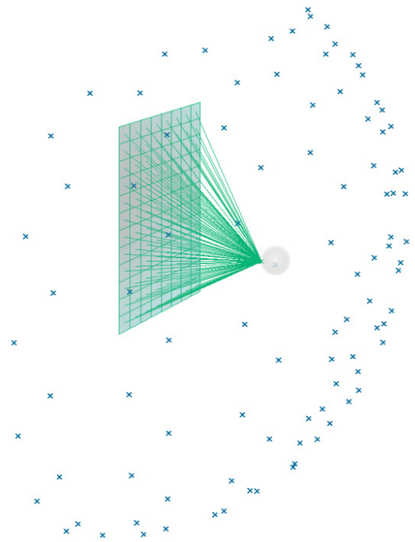


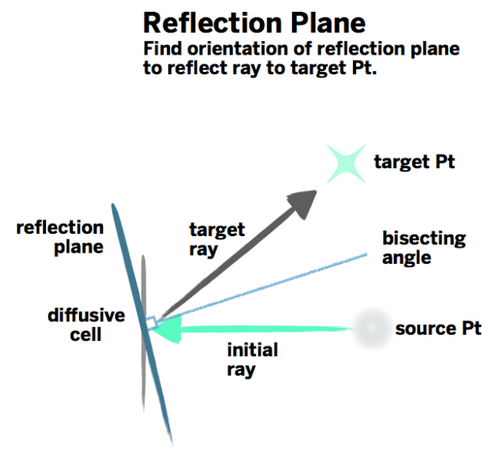


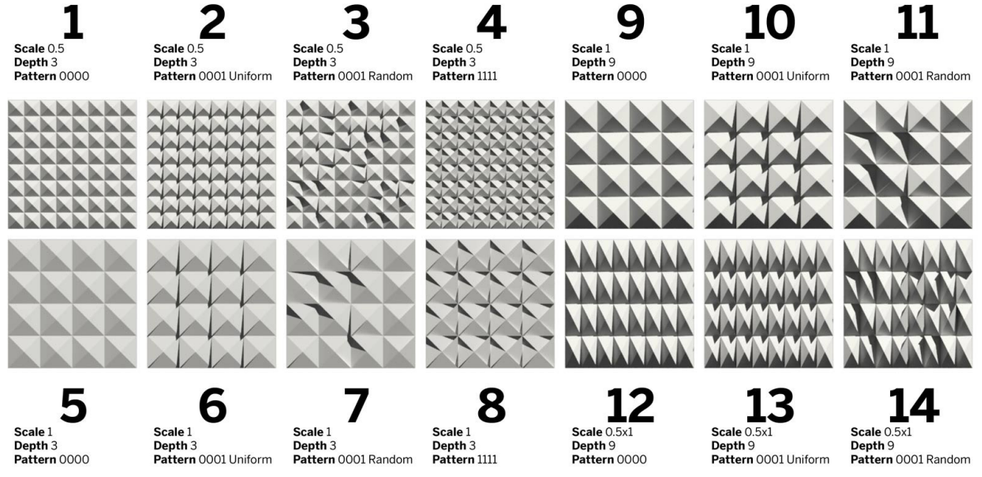
Panelization based on enironmental factors such as sun exposure, wind direction and temperature. These enironmental parameters are then overridden by programatic factors such as program type and occupancy. The double-layer panel consists of a wind harvesting EFTE layer and a new nano-solar energy layer on top. In areas with high winds the aperture on the solar layer gets wider or smaller to allow more or less wind flow based on the season. Conversely, the areas that recieve the most sunlight have panels with no aperature to maximize solar gain.

Acoustic Panel









Images courtesy: LMNTS.lmnarchitects.com

What I need advice on:

- What did I not do that I need to do before moving on?
- How should I structure my experiments?
- Should I be designing experiments with a final output in mind?
- What are some design options that I left out?