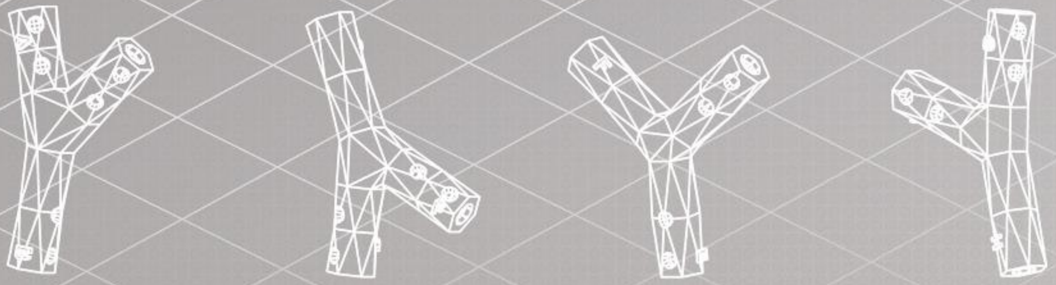
LIMB

PROCEDURALLY GENERATED MULTI-NODAL STRUT NODES



RESEARCH DOCUMENT

UNIVERSITY OF TEXAS ARLINGTON

table of contents

author and proeject data

TITLE

LIMB: Procedurally generated multi-nodal strut joints

AUTHOR

Josh Hallett - Graduate student studying architecture at the University of Texas at Arlington

PROJECT PARTICIPANTS

The work presented herein is a continuation of research objectives established in the Spring of 2014, and much credit is owed to Khang Nguyen and Tenaj Pinder, who were instrumental in reaching those objectives. Professor Brad Bell has provided indispensable guidance and resources, and he continues to furnish both valuable input and material assistance in meeting research outcomes. Collaboration with an Undergraduate researchers Halima Arevalo, Anruth Muthama, and David Garcia yielded excellent results, and my research outcomes were enriched, consequently.

CITY / COUNTRY

Dallas Texas
United States

TYPE

Research/Development of potential pre-cast concrete structural assembly techniques

CONSTRUCTION

The ultimate intent is to fabricate a pre-fabricated, concrete installation of an undetermined scale on campus or at another suitable location.

IMPORTANT TERMS AND DEFINITIONS

Strut – any linear segment of reinforcing material acting as the structural agent between their interconnecting centroids.

Node - component acting as a ‘slotted’ joint at the centroid of two or more intersecting struts.

Procedural Modeling - a form of generative modeling that takes advantage of pre-defined algorithms to propagate a specified series of ‘steps’ or changes to the geometry as a result of modifications to the input criteria.

Grasshopper (GA) - GUI based parametric coding plug-in for the NURBS based 3D modeling platform Rhinoceros.

project overview

01 INITIAL RESEARCH

LIMB constitutes a continuation of research initiated in the Spring of 2014 that sought to discover new and exploratory methods for reinforcing precast concrete panels with non-uniform porosity. It was determined during the course of this research that steel reinforcing rebar could be accurately and effectively located within a panel by utilizing a 3D printed nodes with interior “slots” whereby the rebar could be inserted and then fixed via wire-tie. These nodes were generated in the parametric plug-in tool Grasshopper, which enabled complete control over the nodes geometric properties.

This allowed not only for a vast amount of flexibility in terms of configuration and scale, but rapid alterations in the geometry as a response to changing design criteria and testing. The drawbacks of is particular GA (Grasshopper) definition was multi-fold, however. Where the node could be 3D printed reliably, and was somewhat sturdy, each node had to be manually adjusted to account for each nodal centroid. Furthermore, the node could only account for strut orientations in the X and Y Axis.

02 ESTABLISHING NEW CRITERIA

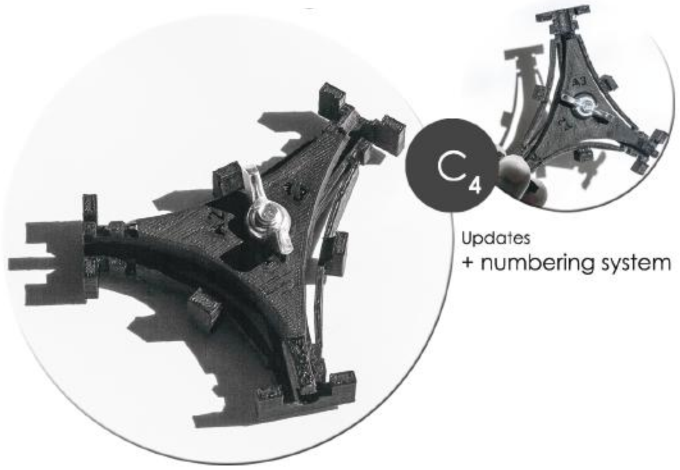
Given these limitations in this node, it became apparent that the initial GA code would need to be rebuilt from the ground up to accommodate new design criteria. This new code would need to enable the procedural generation of nodes at all necessary centroids within the panel/structure formwork, and orient the insertion point of the strut in a similarly procedural fashion. Thus, the intent was to create an adaptive strut node that could:

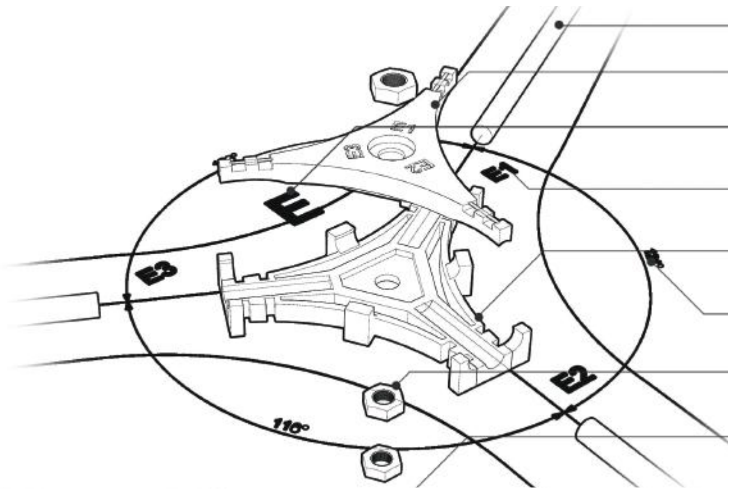
-Identify the requisite centroidal intersections in a porous panel or structure

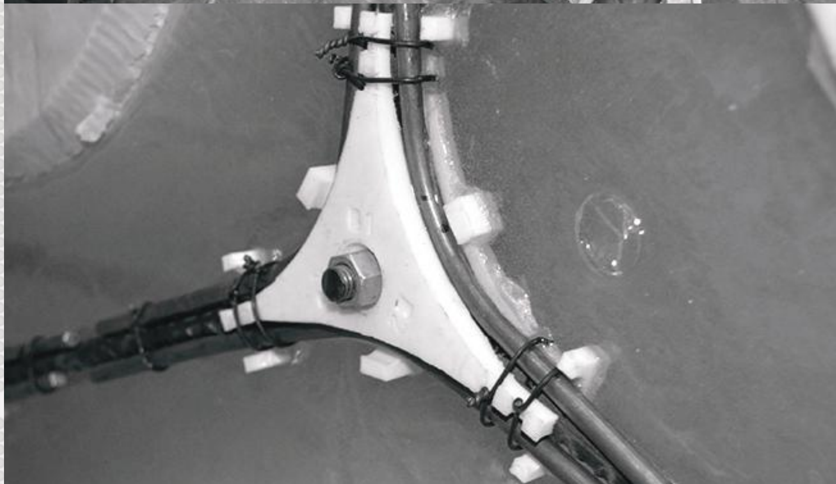
-Generate geometry at those centroids such that they corresponded to the necessary orientation of the strut or rebar reinforcement

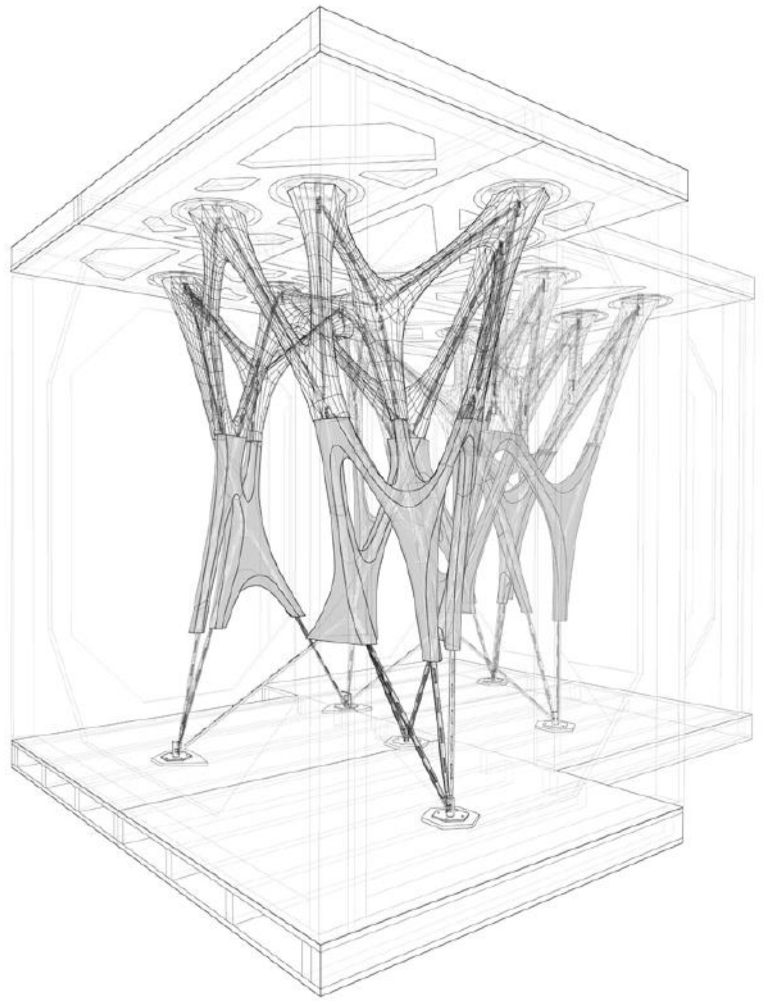
-Implement a cogent coding format that would allow for rapid, tooless on-site assembly.

-Coding operation that could adequate calculate the length of each strut in a structural system









03 ADDITIONAL OBJECTIVES

Several other project objectives were specified, such as exploring potential improvements to the tensile, plastic formwork utilized in the fabrication of Cast Thicket (TOPOCAST), and to utilize GA plug-ins such as Kangaroo, Millipede, and Karumba to explore further form optimization techniques. Secondly, it was also considered advantageous work cooperatively with an undergraduate team that, itself, would be experimenting with novel tensile-formwork casting techniques.

04 PROCESS

As noted previously, the strut-node would function best if propagated simultaneously at all centroidal node-points, and if their geometry could adapt procedurally to changes in the input geometry (in this case, 3D line segments representing the 'structure'). "Functionality", in this case, relating to both the inherent accuracy of the digital model, and the rapidity at which these models can be converted into a file format suitable for printing.

This criteria proved doubly challenging relative to past coding forays,as it would necessitate careful manipulation and organization of data within 'data trees' such that simultaneous manipulation of geometry at multiple locations could be accomplished. Geometry would need to conform to certain universal geometrical and performative criteria, yet posses morphological flexibility needed to integrate with stuts at a specific centroid.

In order to solve these two design problems, the GA code would need to properly interpolate the input geometry. The input geometry needed to be reducible to a series of intersecting line segments that could be 'exploded', such that the discontinuity between these line segments, and thusly the connective centroids or 'points', could be located and effectively ordered into a 'list' within a 'data tree'. Unnecessary points would then be culled from the data tree, leaving only those points with multiple intersecting line-segments.

Spheres with a pre-defined radius are then generated that these isolated points, and the intersection points between the line segments and the outside 'shell' of the sphere are detected.

Depicted top left: diagramtic representation of Cast Thicket

Depicted bottom right: Cast Thicket on display in the gallery space at UTA schoold Architecture.

04 PROCESS CONT.

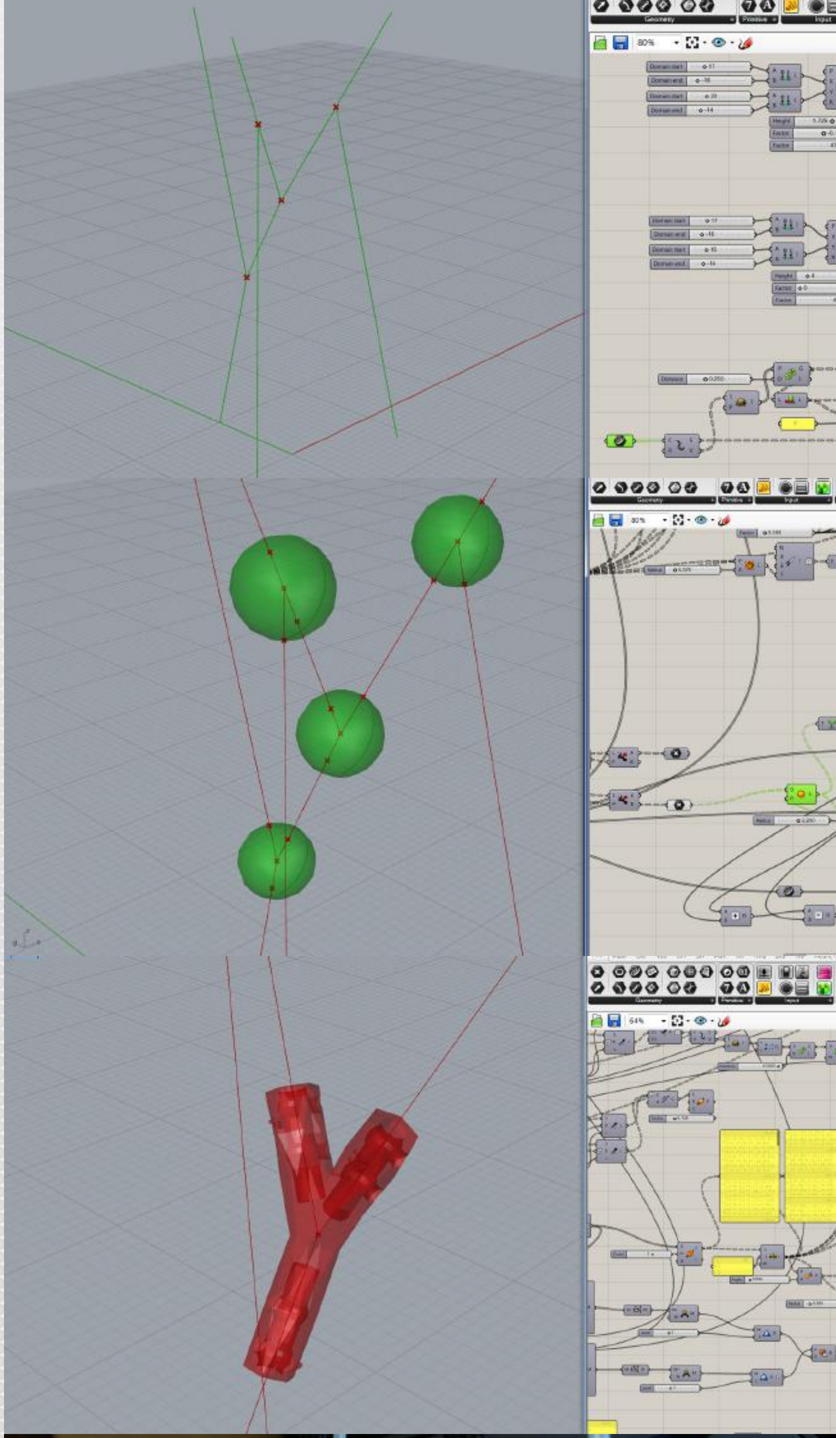
Furthermore, these spheres are used to trim the intersecting line segments at a certain length, allowing their utilization in a variety of ways, including the generation of Exoskeleton mesh elements and locating planes used to position additional assets.

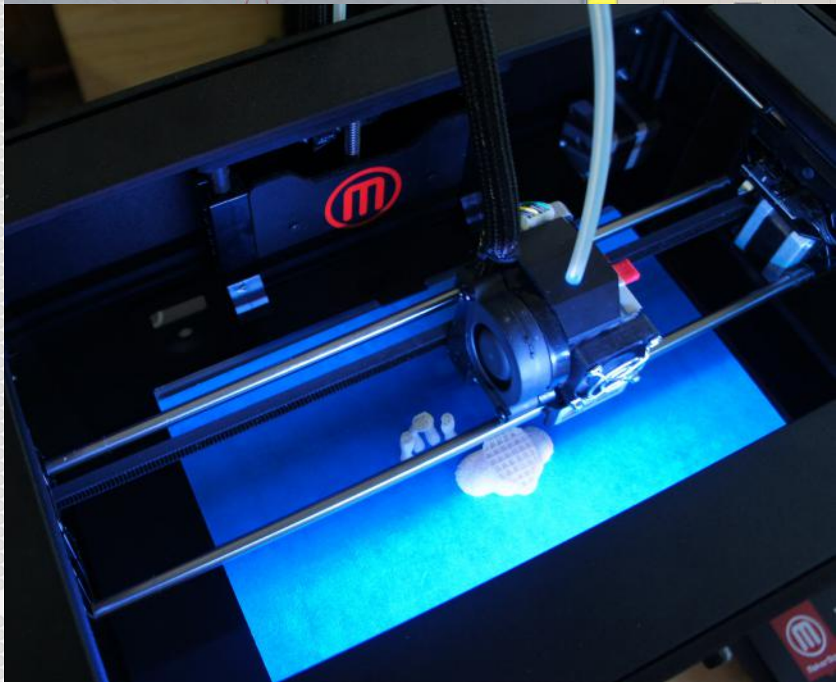
Exoskeleton, a plug-in for GA, “thickens line/node into watertight meshes. It solves the nodes using a convex hull and stitches the hulls together with polygonal struts.” Subsequently, one can begin to generate surprisingly flexible meshes around these isolated point/line combinations. If the input geometry (line segments are altered, the exoskeleton mesh will update correspondingly. Further exoskeleton segments are generated, and utilized in ‘Boolean Difference” operations that produce slots in the mesh, into which the struts will be inserted.

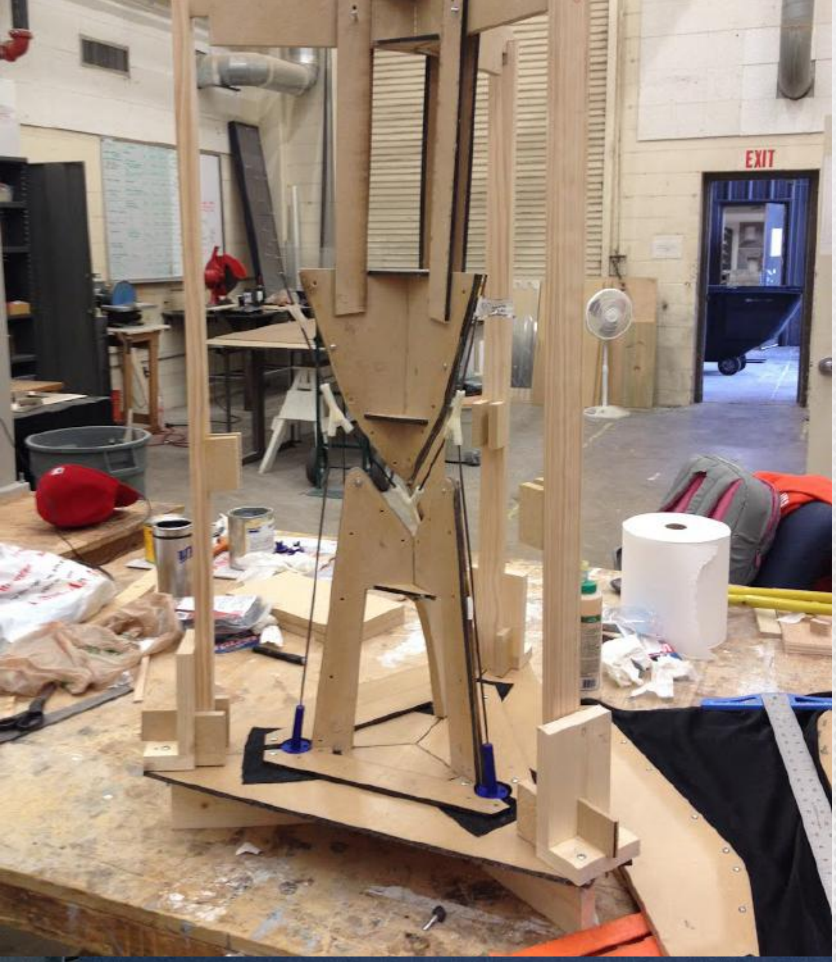
Integrated into the code are several auxiliary functions, such as a labeling/tagging system and a definition that would relocate the final, generated geometry to C-Plane where they could be quickly exported for 3D printing. Consequently, the overall GA definition had to be rebuilt many times in order to properly integrate changing research criteria and to make certain functions more efficient.

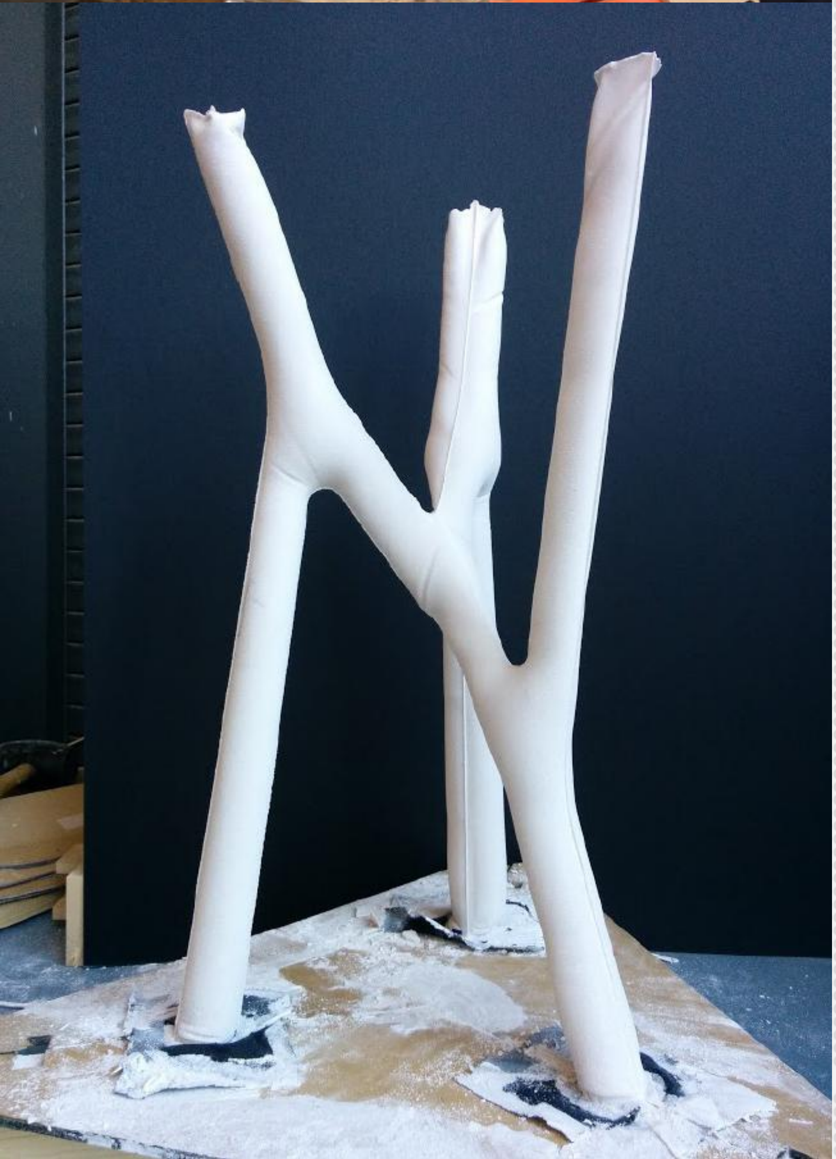
Apart from the challenge of generating the geometry, it was also essential that each node could be feasibly fabricated via 3D printing (Fused Deposition Modeling). Difficulty in achieving function 3D prints varied considerably depending on the overall complexity of the geometry, and the quality of the printer being utilized. Earlier node prototypes could be printed reliably on conventional, consumer grade printers. As the input geometry changed, however, from simple branching structures, to more complex geometries (such as cast thicket), the degree of complexity in the nodes required a printer capable of generating elaborate supports. Luckily, access to a Stratasys Systems uPrint SE Plus was provided. This printer could print models at incredibly high resolution, ensuring near one to one correspondence between the 3D geometry and the final, fabricated node.

Research vectors ultimately resulted in a reduced-scale production of a proposed skeletal framework for Cast Thicket, and a full-scale skeletal framework developed for an undergraduate research team developing fabrication techniques for concrete branching structures.









05 COLLABORATION

Once the GA definition for the node had reached a functional state, and it was demonstrated that node prototypes could be successfully printed, direct cooperation with Brad Bell's undergraduate team became advantageous. The undergraduate team was currently exploring the extended application of bulge casting (fabric casting) techniques originally pioneered at the University of Manitoba by Mark West. The teams intent was to explore casting methodologies and novel formwork innovations that would allow them to cast multi-directional branching structures. As my research dealt with the structural reinforcement of non-uniform, concrete branching structures, there were natural synthesis between research objectives. This collaboration would provide me with the opportunity to apply my research practically and direct my focus singularly to the functionality of the node.

In addition to the GA work concerning the node, generative modeling techniques making use of L-Systems were explored, and a large number of small-scale models were produced. It was later deemed that the dendriform geometry was structurally problematic, and a tripod like morphology was devised in Rhino and then formally optimized in Kangaroo through Mesh Relaxation.

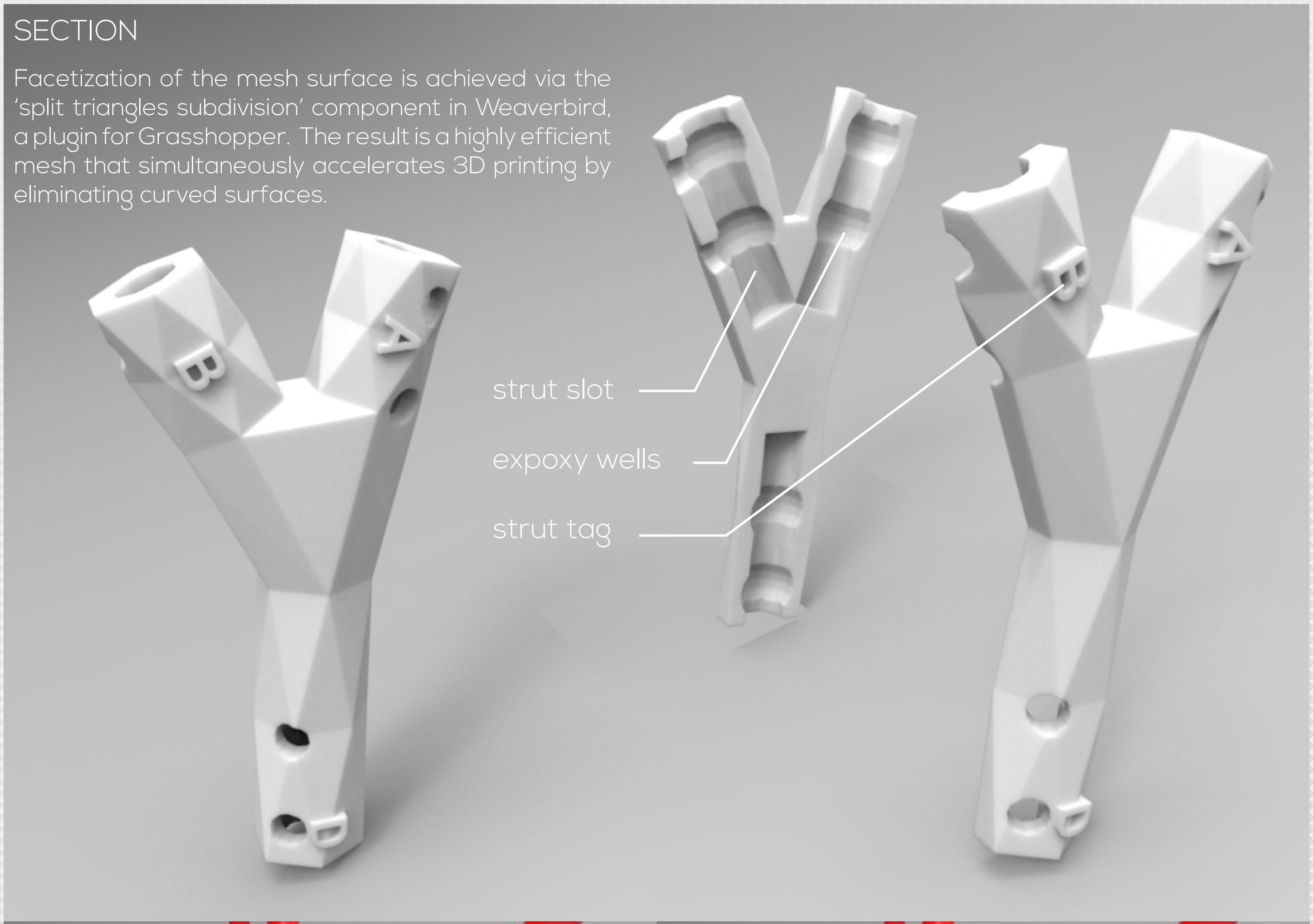
It quickly became apparent that several different types of nodes would need to be developed. Otherwise, the interior reinforcement could not be properly integrated within the overall formwork. Top and base nodes would enable the undergraduate team to fix the steel reinforcement to MDF formwork during casting, and subsequently fixing the steel reinforcement.

While it sometimes proved problematic ensuring that the internal, skeletal structure properly fit within the formwork, the exercise demonstrated that, in practice, the 3D nodes facilitated quick, accurate, structural reinforcement.

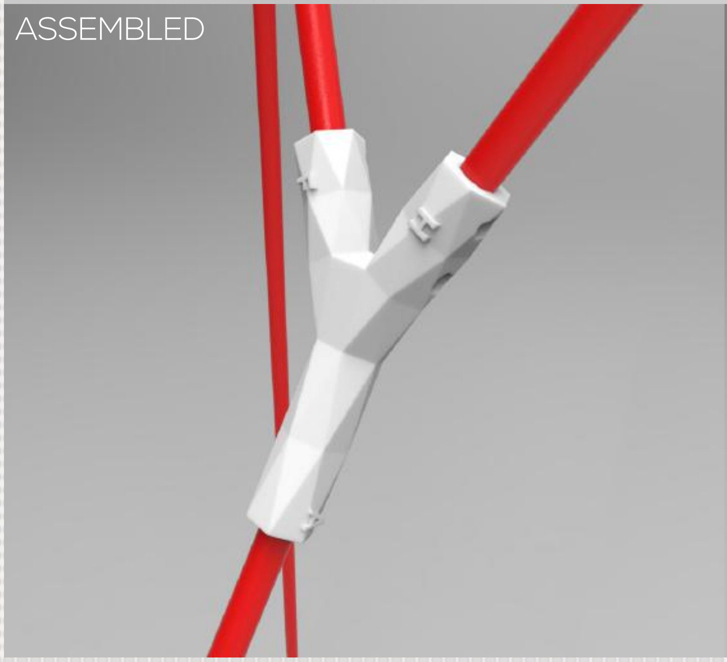
Depicted top left: MDF forwork assembly with rebar cage and interconnected nodes. Seven nodes in total were printed for this prototype.

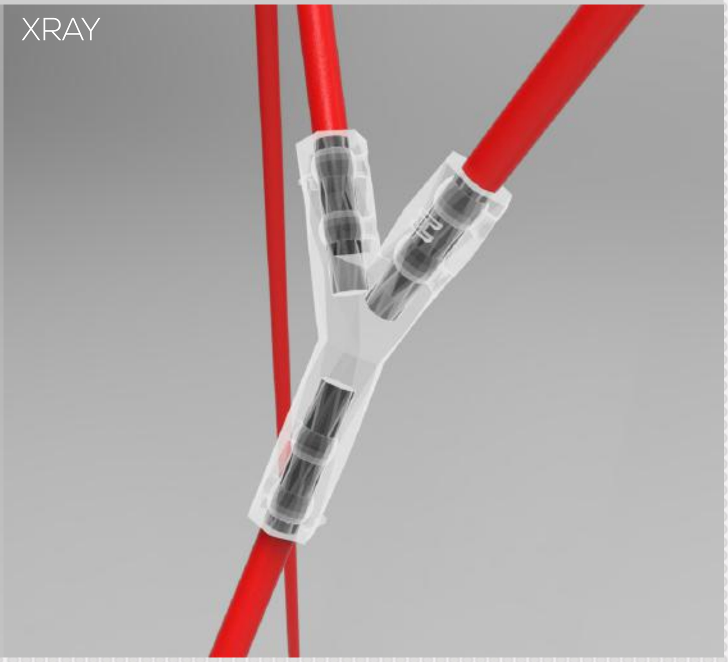
Depicted bottom left: The final hydrostone prototype with reinforcing cage cast internally.

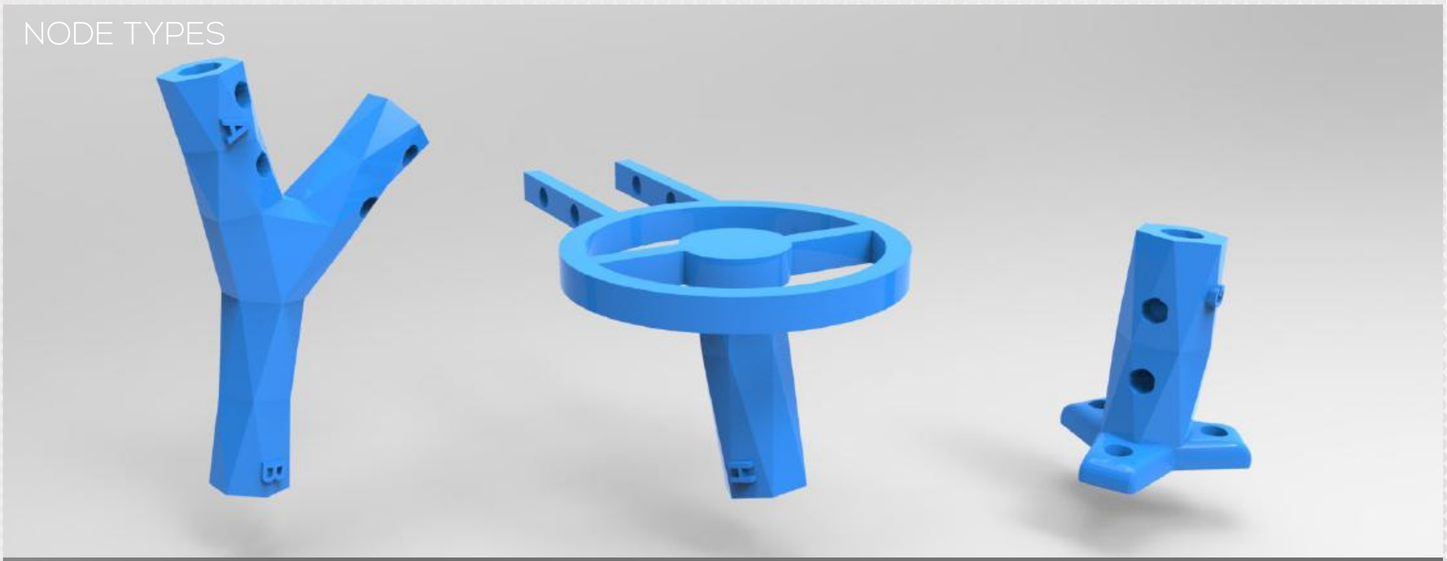
diagrams

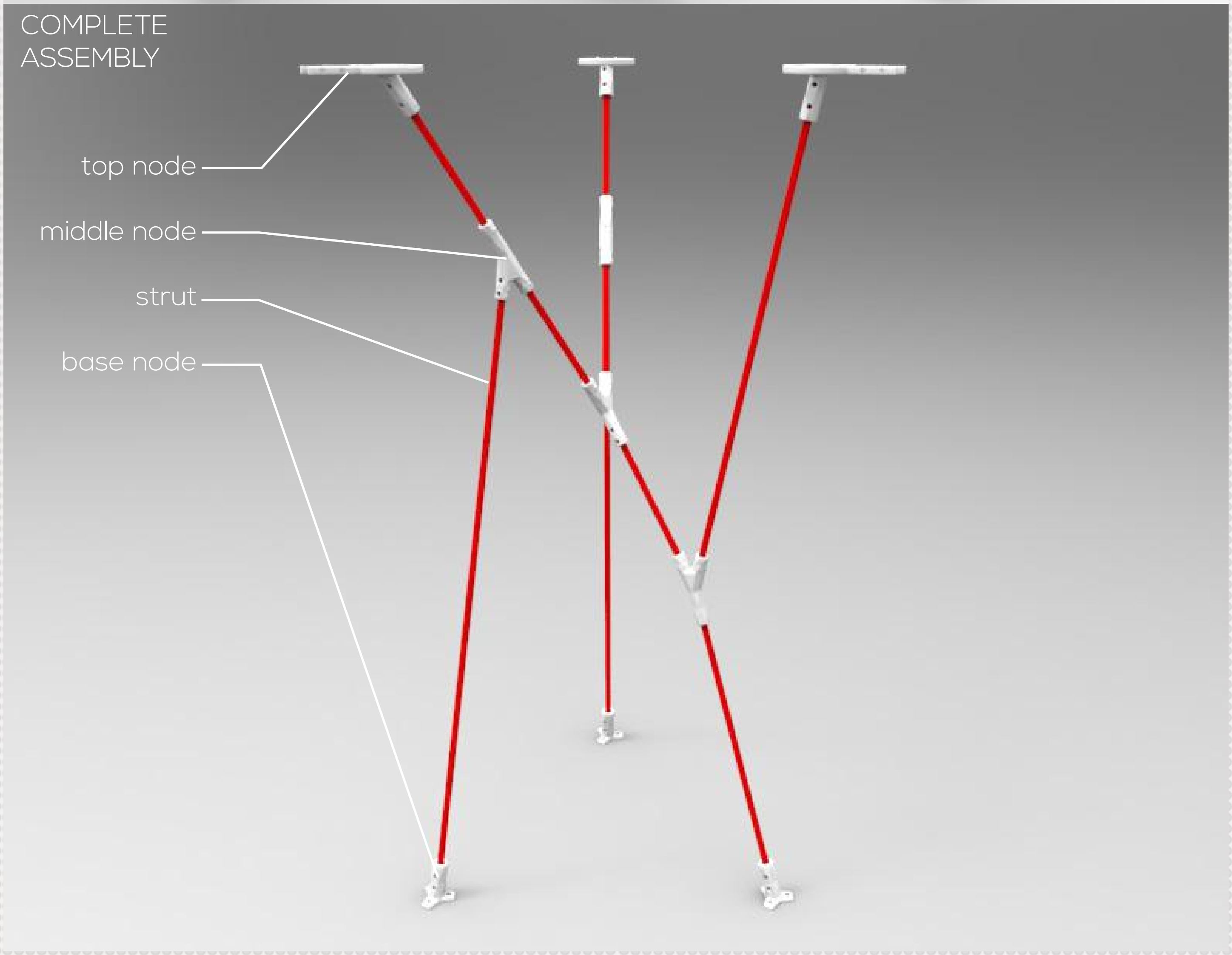


Facetization of the mesh surface is achieved via the 'split triangles subdivision' component in Weaverbird, a plugin for Grasshopper. The result is a highly efficient mesh that simultaneously accelerates 3D printing by eliminating curved surfaces.






NODE TYPES


COMPLETE ASSEMBLY
top node
middle node
strut
base node

design challenges & prospects

01 PREFACE

The purpose of this section is two-fold: to elucidate difficulties that arose during all aspects of the research process, and to prime the next research time that carries this work forward. Subsequently, future teams will be fully aware of the challenges they will likely encounter, and have a good baseline for surmounting those problems.

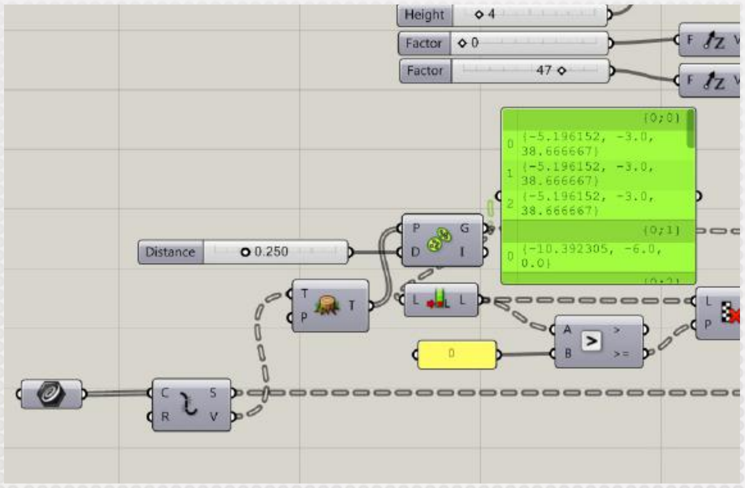
01 GRASSHOPPER

As stated earlier, the most difficult challenge to be faced when coding multiple nodes is ensuring that geometry propagates correctly at all desired locations. Confusion regarding data-trees, and how they function, can easily result in hours of wasted effort (as I learned firsthand). Once definitions reach a certain scale - dozens of components - attempting to resolve particular issues can easily confound and frustrate. Given one's general proficiency with Grasshopper, it may be advisable to break coding problems into manageable chunks.

If one is not well versed in Grasshopper, the nexus for understanding the research presented here is to practice deconstructing different types of geometry into line/point, and then carefully clean up and structure the data.

While the current version of Exoskeleton 2 allows for a wide range of deformation, there are limitations to keep in mind. If the angle between two line segments is too acute, and the radius/thickness of the mesh to large, an overlap in mesh geometry will occur. This will prevent the convex hull from being properly 'drawn', and the coder must either reduce the acuteness of the angle, or reduce the radius of the tubular geometry being generated by the Exoskeleton component. One must carefully evaluate the input geometry and ensure that the angles between struts are not too acute, or take steps to adjust the node geometry.

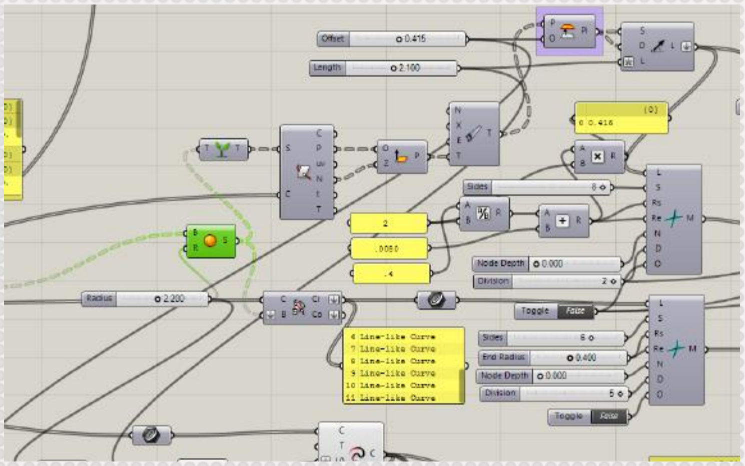
It is also crucial that one is able to calculate the exact dimensions of the struts that will be utilized. Though a methodology for accomplishing as much has been provided, new research could explore alternative strategies.



Height 4
Factor 0
Factor 47
Distance 0.250

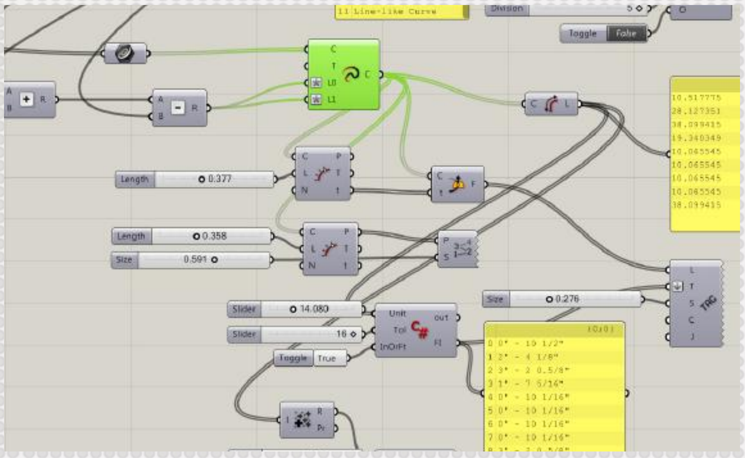
```
{0;0}
0 {-5.196152, -3.0, 38.666667}
1 {-5.196152, -3.0, 38.666667}
2 {-5.196152, -3.0, 38.666667}
{0;1}
0 {-10.392305, -6.0, 0.0}
```

Seen here, the input geometry is 'exploded' into line and point, vertices flattened, and then duplicate vertices grouped so that non-duplicates can be culled via index masking/dispatch operation

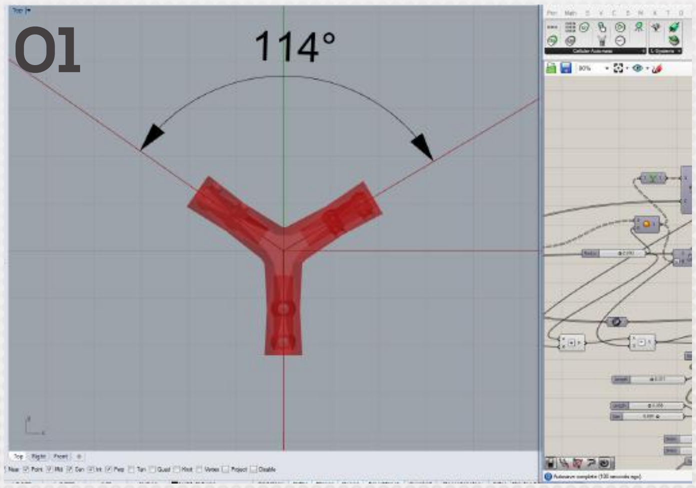

Offset 0.415
Length 2.100
Sides 8
Node Depth 0.000
Division 2
Radius 2.200
Toggle False
Sides 6
End Radius 0.400
Node Depth 0.000
Division 5
Toggle False

```
4 Line-like Curve
7 Line-like Curve
8 Line-like Curve
9 Line-like Curve
10 Line-like Curve
11 Line-like Curve
```

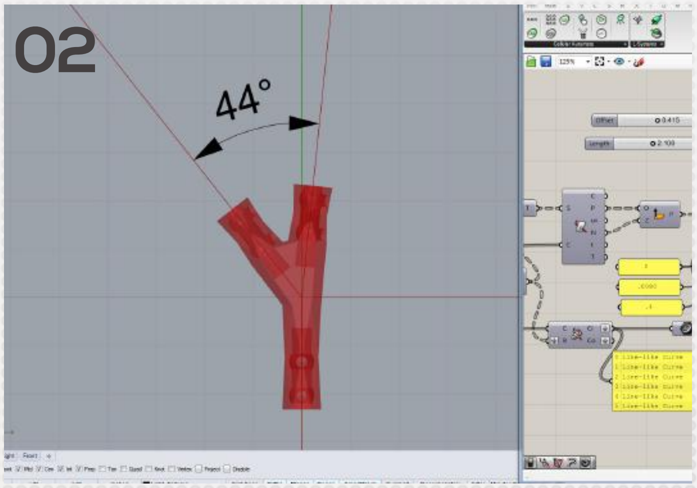
Shown here is the routine for successfully generating the convex hulls that will serve as the base geometry for the node. The overall length of each node slot is dependent on the predefined radius of the spheres/breps being used to trim the intersecting lines


Length 0.377
Length 0.358
Size 0.591
Slider 14.080
Slider 18
Toggle True
Size 0.276

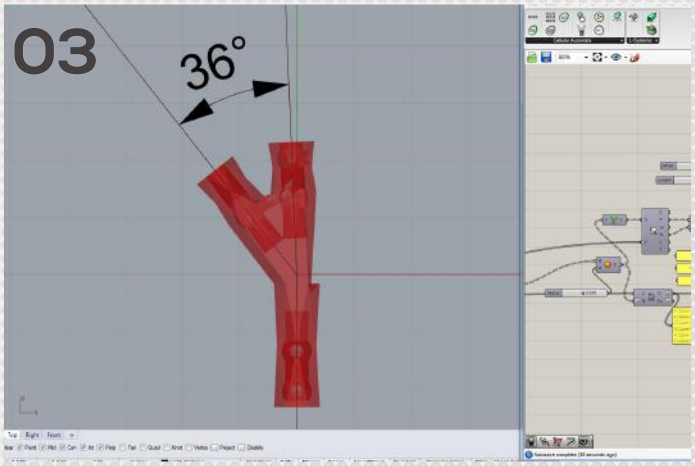
03 NODE DEFORMATIONS AND LIMITATIONS



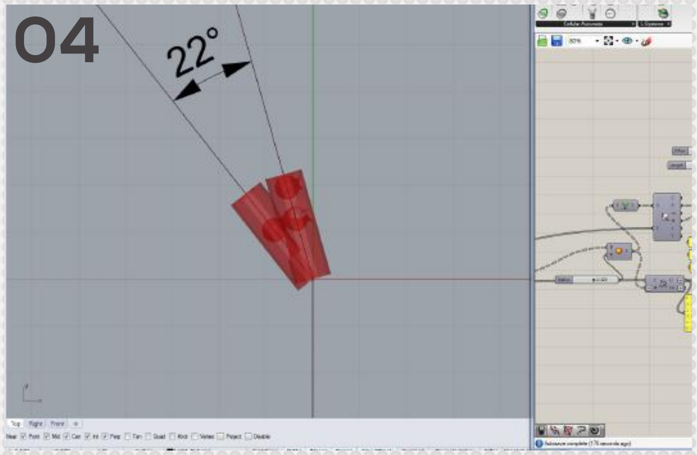
Obtuse angles yield no erratic or unwated outcomes.



Even at 44 degrees, the definition continues to function with no erros, though one should note that as the gap is reduced, the chance of overlap between the strut-slots increases



At 36 degrees, minor drawing errors may develop in a small number of nodes. These issues can usually be resolved either by making slight alterations to geometry parameters.



Given the parameters of this node, failure occurs at 22 degrees. The only options available are to reduce the radius of the Exoskeleton 'strut', or increase the length of the strut-slot so that the strut and node no longer overlap.

04 IMPROVING CODE

The most important aspect of the node is procedural deformation with respect to input geometry. It's performance and, indeed, its utilization becomes questionable if geometry cannot be developed around line segments with acute angles. Continued research should explore methodologies for developing convex hulls that are not susceptible to overlap, and properly adapt to either extremely acute/obtuse angles.

This goes without saying, but any modifications that reduce the overall scale of the definition are ideas, as certain operations (Mesh Difference, for example) can hog system resources and changes to parameters can take excruciatingly long to complete (or even crash the definition!).

Lastly, it would also be advantageous to write a 'better' definition capable of relocating each node in the system to a predefined location with an optimized orientation to expedite 3D printing.

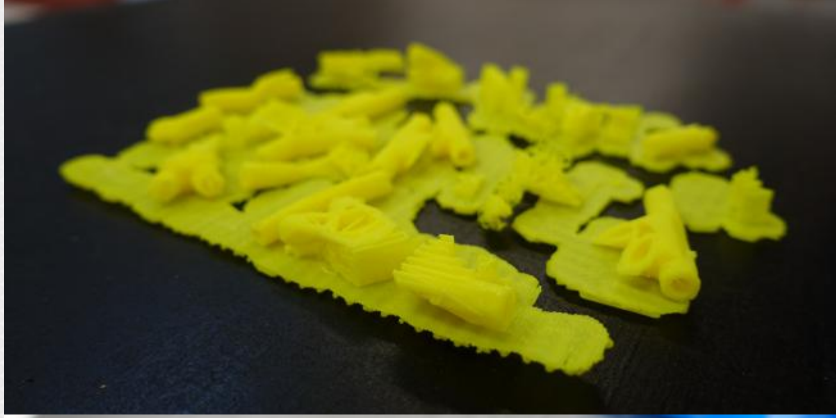
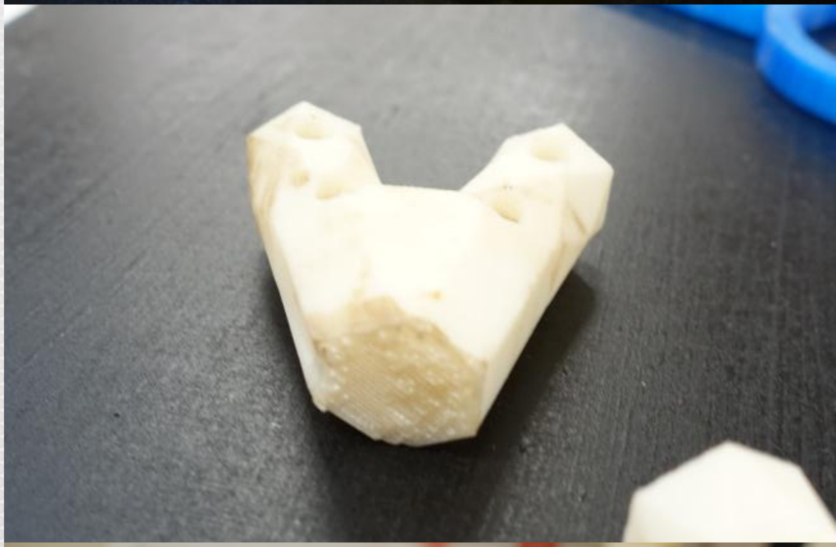
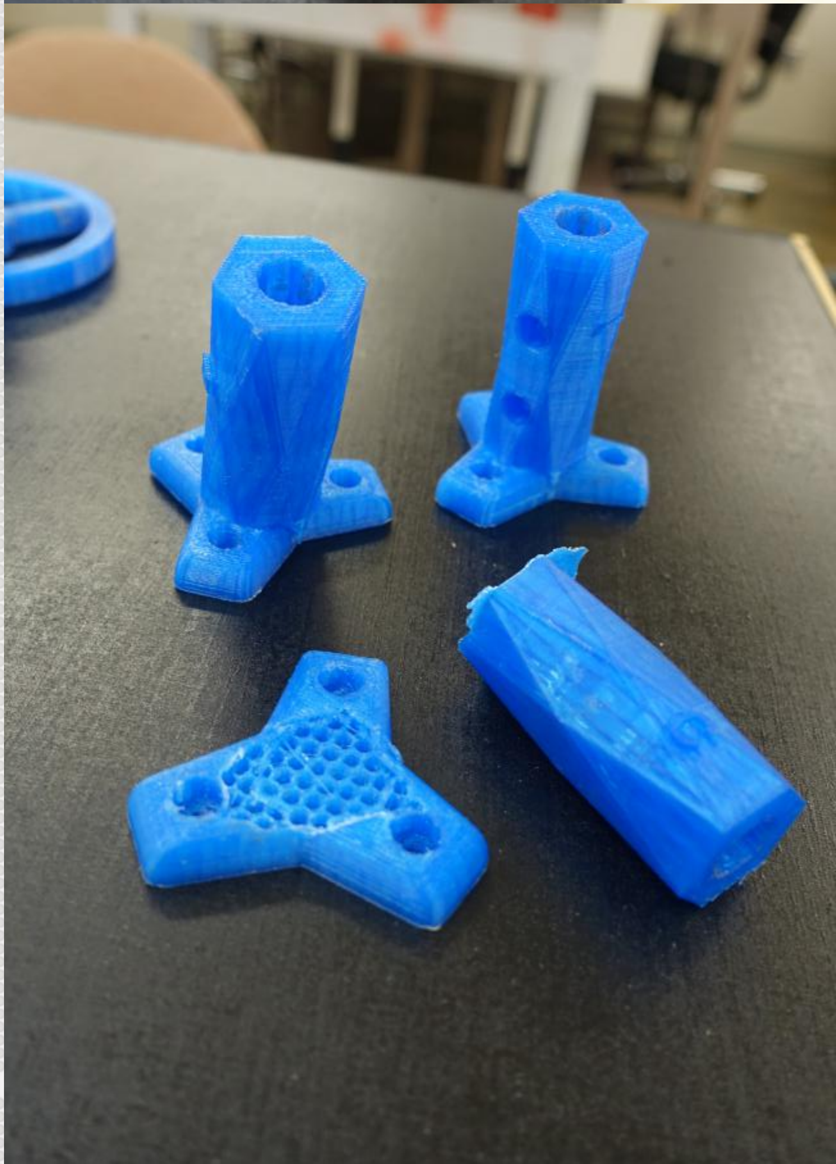
05 FABRICATION

Difficulties faced during fabrication dealt almost exclusively with 3D printing and the strength of the printing material (ABS plastic, in this case). Scale was a contributing factor to many failures, as many consumer printers were unable to print nodes at a small scale. Printing with rafts, or reducing the number of nodes being printed can mitigate some of these issues at scale, however, it proved to be much more efficient to simply print smaller nodes on a higher quality printer.

As mentioned previously, the sometimes unwieldy geometry of the nodes will require 3D printed supports in order for the print to be successful. Supports generated by Makerware were deemed unsatisfactory on every occasion. The uPrint SE Plus proved to be the most reliable option, but at a higher cost per print.

Even when printed at high density, ABS plastic will delaminate quite easily given even a small amount of force. Great care should be taken during assembly to avoid putting too much pressure on the protruding strut-slots, as these carry the most risk of being cleaved from the core centroid of the node. While improvements to geometry may yield improved rigidity, the best solution would be to change the filament material used during printing.

The success of any digital fabrication project requires a close correlation between the final, fabricated prototype, and its accuracy relative to the digital model. Knowing whether or not there have been slight deformations in the ABS during printing, for instance, may be difficult to discern, and could have considerable impact upon the final assembly. If the material being utilized as the structural strut is uneven or crooked, the final results may be even more askew. On some occasions, the strut-slots of several nodes had to be milled out with a larger diameter bore so that the desired strut could be properly fit to the node. The failure to account for the variability of your building materials can result in a series of compounding errors that further reduces conformity to the digital mode.

Top right: An attempt to print 24 small, individual nodes in one attempt ends in failure.

Middle & lower right: full-scale nodes broken during assembly.

06 PROSPECTS

While initial research has proved promising, the intent is to produce a variable, procedurally generated node suitable for industry utilization. Currently, the nodes lack the material strength necessary to withstand the abuse of a conventional construction setting. Furthermore, the question of whether or not the nodes meet industry 'best-practices' for assembling rebar has yet to be satisfactorily answered. Thus any future research team should explore the following:

-*Material strength.* Nylon is a much stronger material with high resistance to delamination when compared to ABS or PLA plastics. The need for highly variable deformations in the node geometry necessities a material sympathetic to mass custimization.

-*Fabrication accuracy*: Develop methodologies that ensure simultaneity with the digital model. Much like Cast Thicket, the use of plum lines hanging from each node, corresponding to a corresponding location on the ground plane, would enable fabricators to make minor adjustments as needed during assembly.

-*Geometric optimization.* Grasshopper plugins such as Kangaroo or Millipede could refine the node mesh, making the node both more structurally sound and materially efficient.

-*Multi-Discipline Integration*. Consultation with structural engineers (especially those experienced with rebar) would be critical for determining whether the fastening strategies presented here would be ideal for industry implementation.

-*Large-scale installation.* Proof of concept has been roughly demonstrated, but the development of an installation combining both the flexibility of reinforcing system composed of strut nodes, and tensile formwork would present a unique opportunity to combine a number of different research agendas currently being pursued by Brad Bell and the Digital Fabrication Consortium at UTA.

Top left: some unfortunate casualties during assembly of the Cast Thicket mockup

Bottom left: the ends of each rebar segment needed to be ground down in order for them to fit cleanly.

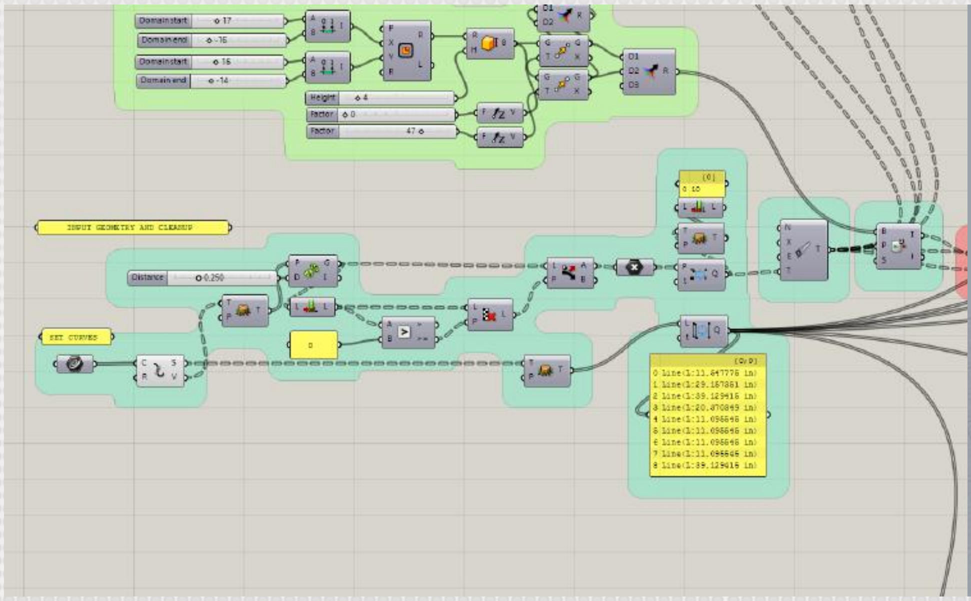
grasshopper definition

01 BREAKDOWN

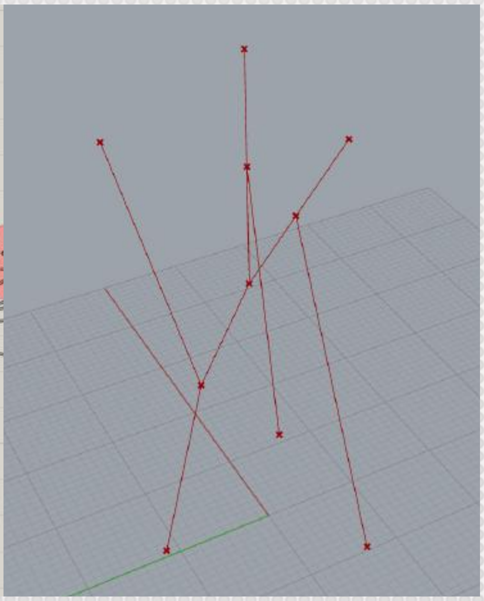
The purpose of this section is to, hopefully, discombobulate the GA definition for the uninitiated, or for anyone who happens to take this research forward. Unfortunately, and regardless of how detailed this guide might be, Grasshopper definitions are innately idiosyncratic in nature, each one representing the personal preferences and workflows of the individuals who built them. There are no work "standards" when it comes to Grasshopper, unfortunately.

I have attempted to group 'component clusters' based on their function, but a good amount of reverse engineering may be required nonetheless. Each cluster will be diagramed, explained, and paired with some relevant imagery from Rhino depicting various aspects of what that cluster may generate or control. Access should also be made to earlier definitions, the study of which will give you insight into the developmental process of the node.

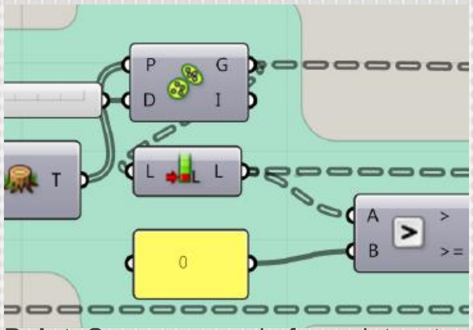
01 INPUT AND CLEANUP



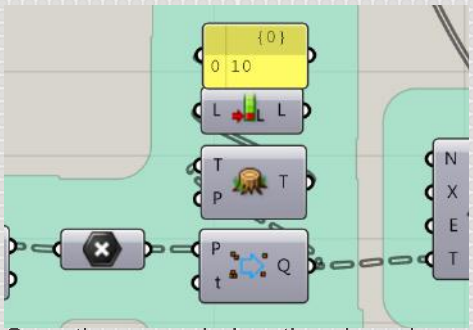
```
{0;0}
0 line(l:11.647776 in)
1 line(l:29.167861 in)
2 line(l:39.129416 in)
3 line(l:20.870849 in)
4 line(l:11.096646 in)
5 line(l:11.096646 in)
6 line(l:11.096646 in)
7 line(l:11.096646 in)
8 line(l:39.129416 in)
```



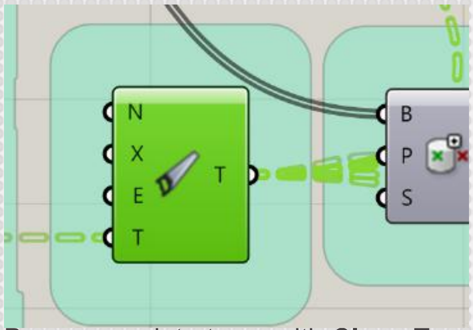
This is simple, yet crucial, portion of the definition. Depending on your input geometry, it ultimately needs to be reduced to a combination of verities and line. The input geometry should be flattened, unnecessary vertices culled, and duplicates deleted.



Point Groups search for points at a provided distance from an existing set of points.

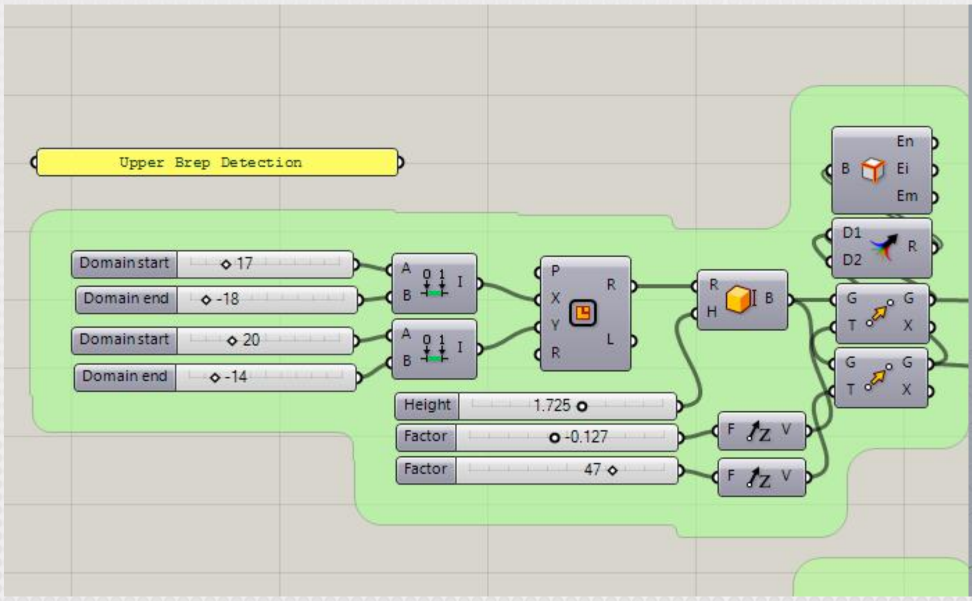


Once the unneeded vertices have been culled, use **Delete Duplicate Points** to remove overlapping vertices



Prune your data trees with **Clean Tree** and remove null values.

02 VERTICE DETECTION


Upper Brep Detection

Domain start 17

Domain end -18

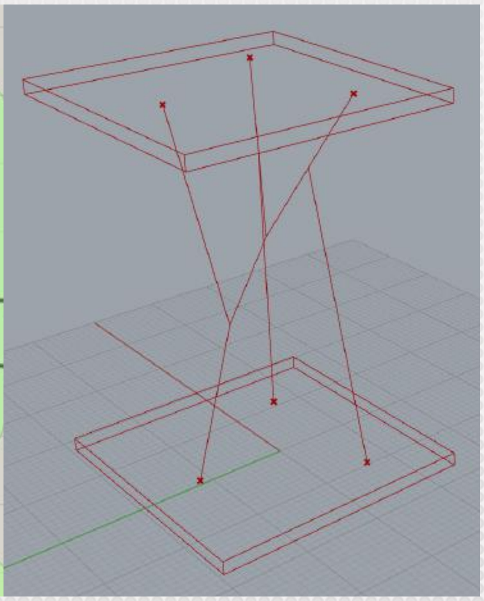
Domain start 20

Domain end -14

Height 1.725

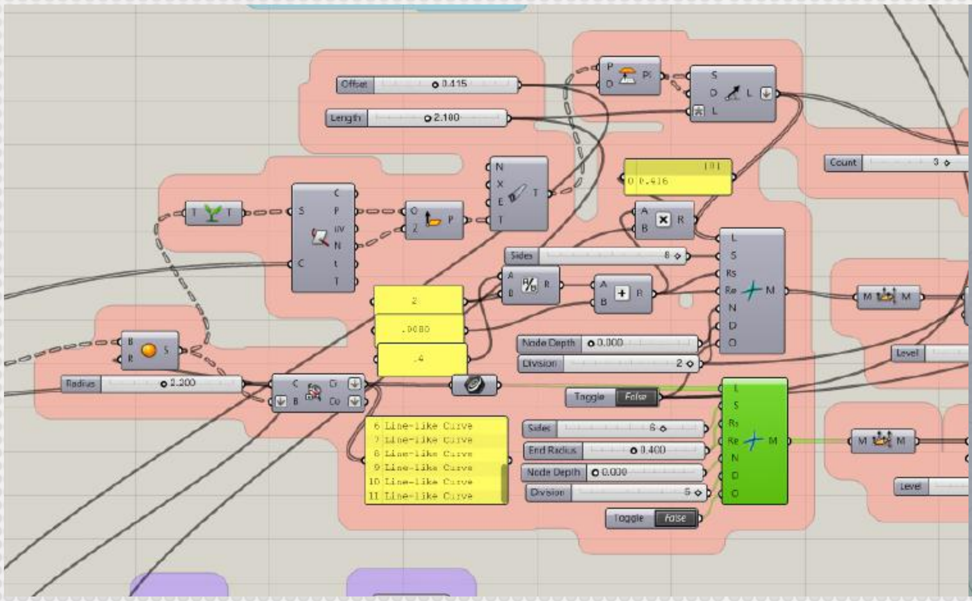
Factor -0.127

Factor 47



If necessary, you may need to break out the top and bottom level vertices so as to generate specialized nodes. This portion of the definition generates a scalable detection boundaries for both the top and bottom vertices.

03 CONVEX HULL GENERATION


Offset 0.415

Length 2.100

Count 3

Sides 8

Node Depth 0.000

Division 2

Toggle False

Radius 2.200

Sides 6

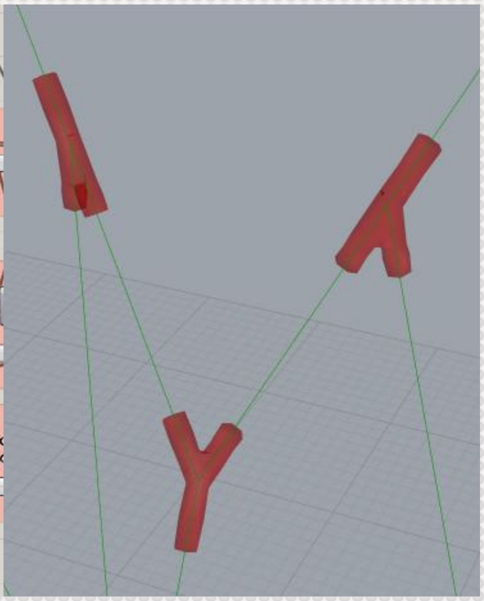
End Radius 0.400

Node Depth 0.000

Division 5

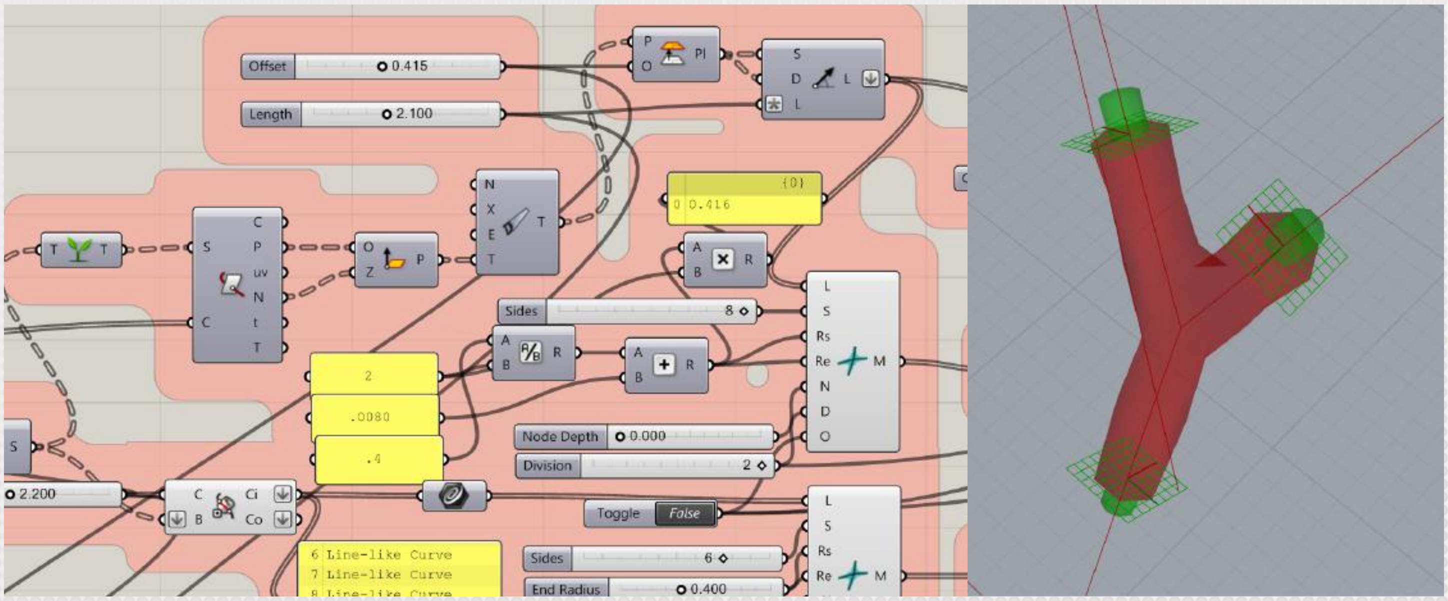
Toggle False

```
6 Line-like Curve
7 Line-like Curve
8 Line-like Curve
9 Line-like Curve
10 Line-like Curve
11 Line-like Curve
```

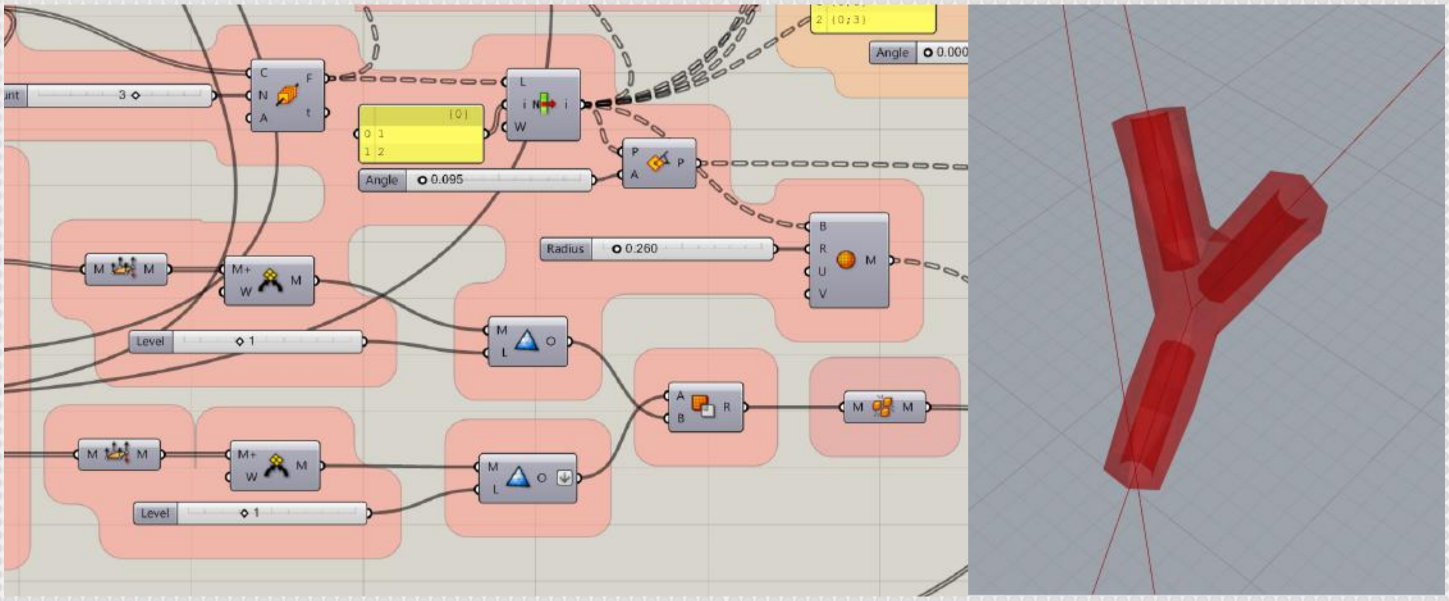


Once the desired vertices have been isolated, use them to generate spheres. These spheres will be used to trim the line segments that we've derived from the original input geometry. Use the **Trim With Breps** component and then pull out the **Inside List** of 'line-like curves'. These curves will be used to generate the convex hulls via the **Exoskeleton 2** component, so they'll need to be flattened if they haven't been already. The resulting meshes should be similar to those shown above. Exoskeleton 2 will also be needed to create the subtractive geometry required for the **Mesh Difference** component.

03 CONVEX HULL GENERATION CONT.

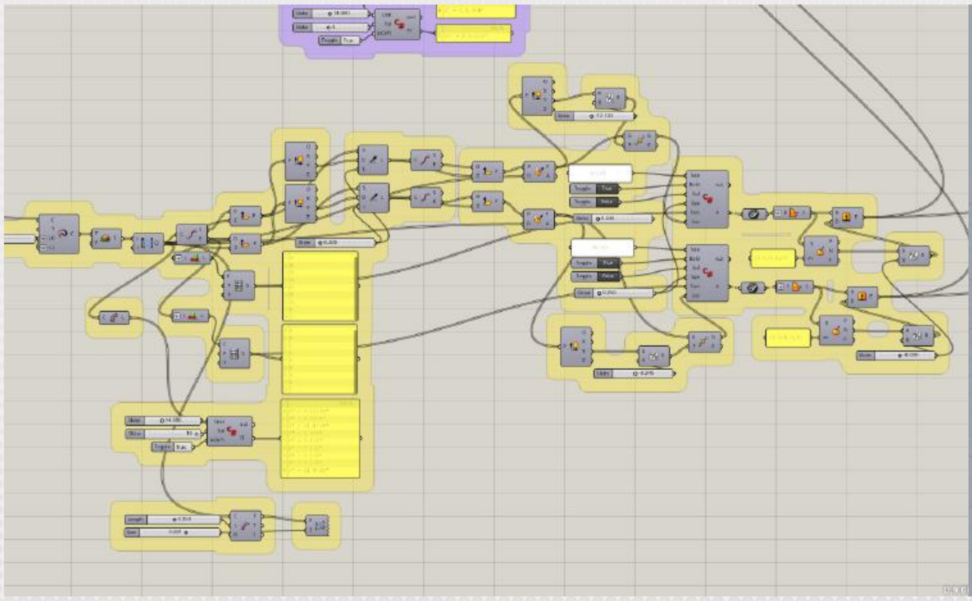
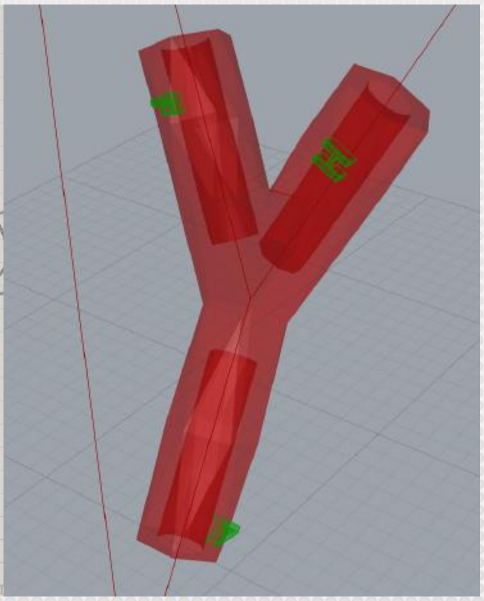


Connecting the cleaned up, input curve geometry, and spheres to a **Surface I Curve** component will provide points and normals from which **Plane Normals** will be located, and used to offset **SDL Curves** that then get piped into a second **Exoskeleton 2** component.



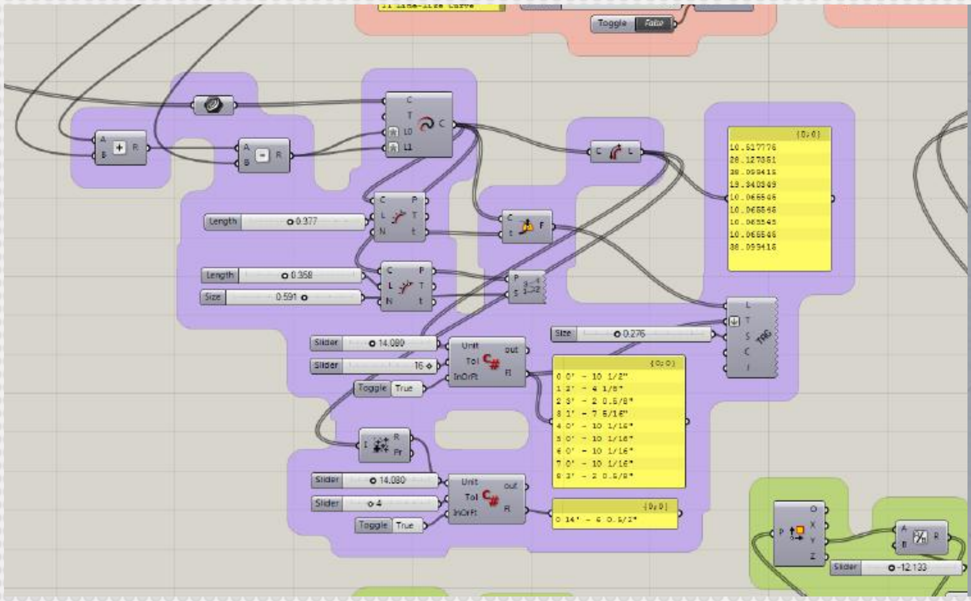
At this point, there should be two **Exoskeleton 2** components generating both additive and subtractive geometry: the first generating the 'limbs' for the strut node, and the second generating subtractive cylinders. It is generally advisable to **Unify Mesh Normals** and then join and weld these meshes to surface uniformity. **Weaverbird's Split Triangle Subdivision** is then used to facetize the surface, a treatment that will expedite 3D printing later on. Both meshes can now be connected to the **Mesh Difference** component.

04 EXTRUDED STRUT TAG

This portion of the code takes the end points and the start points on a series of curves and locates a series of planes that then allow for the orientation of 3D text. The 3D text is made possible by a custom script that converts text to curves. These curves are then extruded and capped.

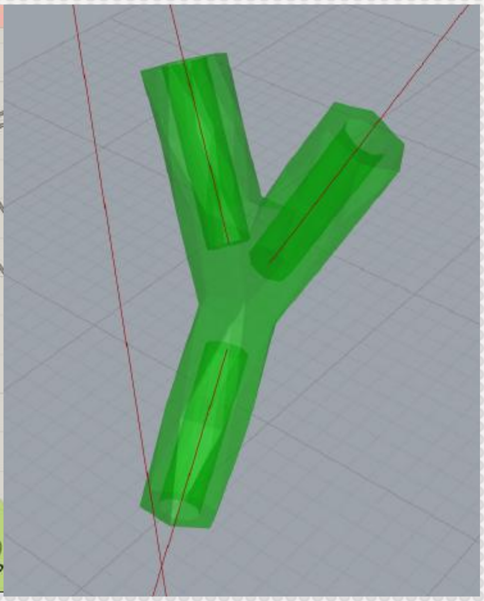
05 STRUT LENGTH



```
{0;0}
10.517776
28.127381
28.099416
19.340849
10.066546
10.066546
10.065545
10.066546
38.099416
```

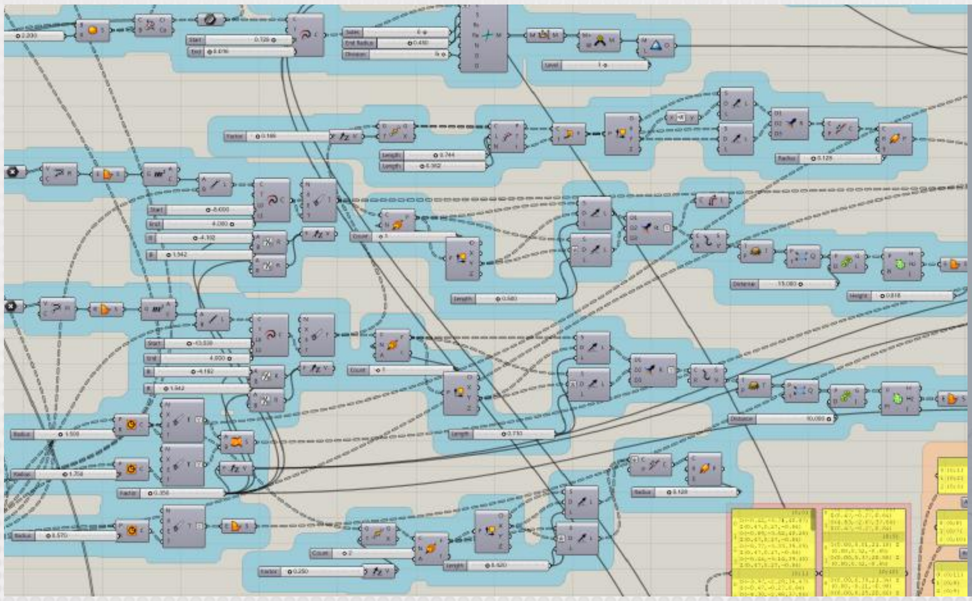
```
{0;0}
0 0' - 10 1/2"
1 2' - 4 1/8"
2 3' - 2 0.5/8"
3 1' - 7 5/16"
4 0' - 10 1/16"
5 0' - 10 1/16"
6 0' - 10 1/16"
7 0' - 10 1/16"
8 3' - 2 0.5/8"
```

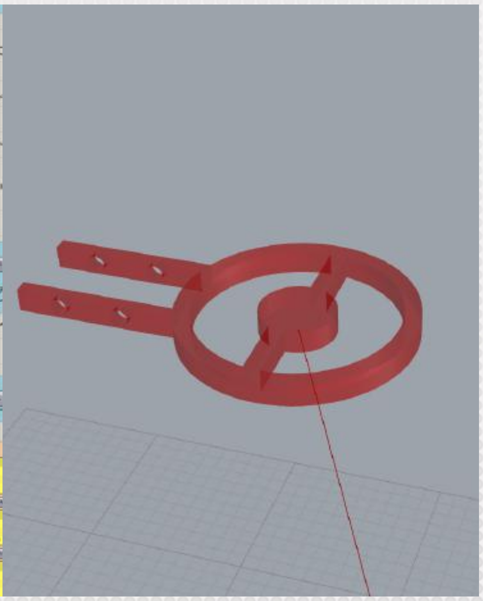
```
{0;0}
0 14' - 6 0.5/2"
```



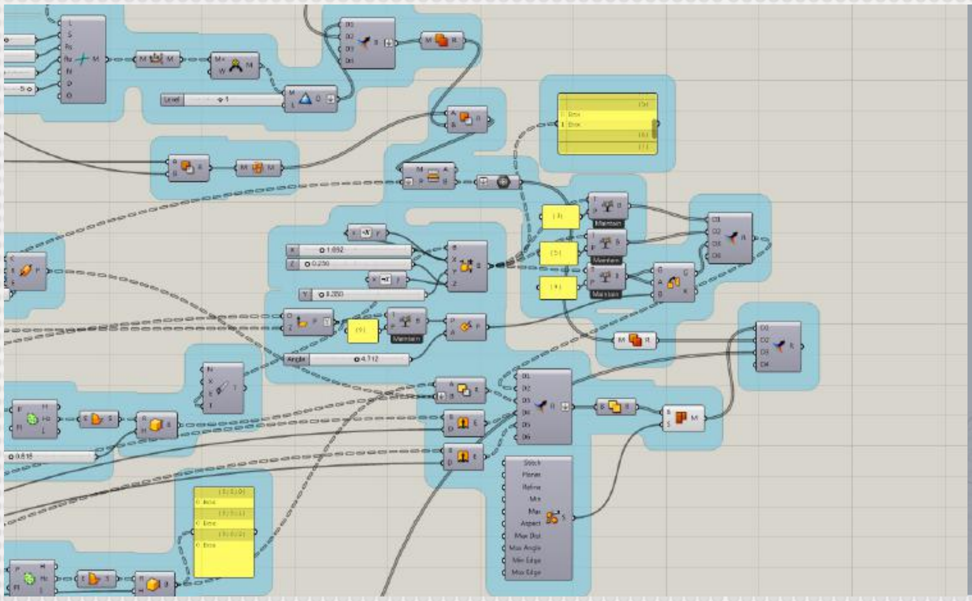
Each curve is reduced in length such that they terminate at the end of each strut-slot. A reliable series of dimensions can be derived from these curves. These dimensions correlate, in terms of their order on the list, to the sequence of tags generated in by the tagging operation described above, so curve “0” with length 10 - 1/2” will match curve A and vice-versa.

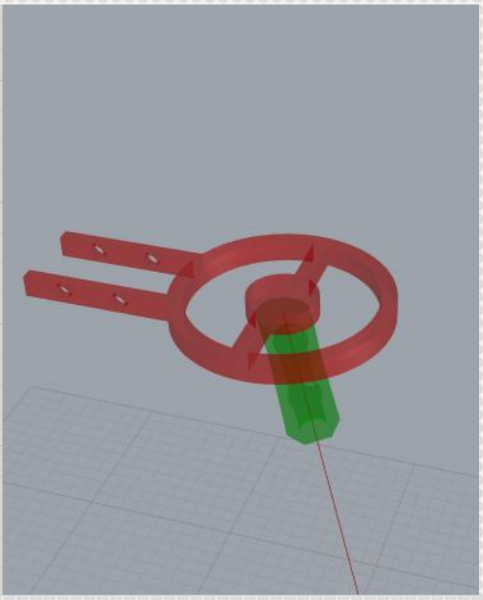
06 TOP PORTION STRUT NODES





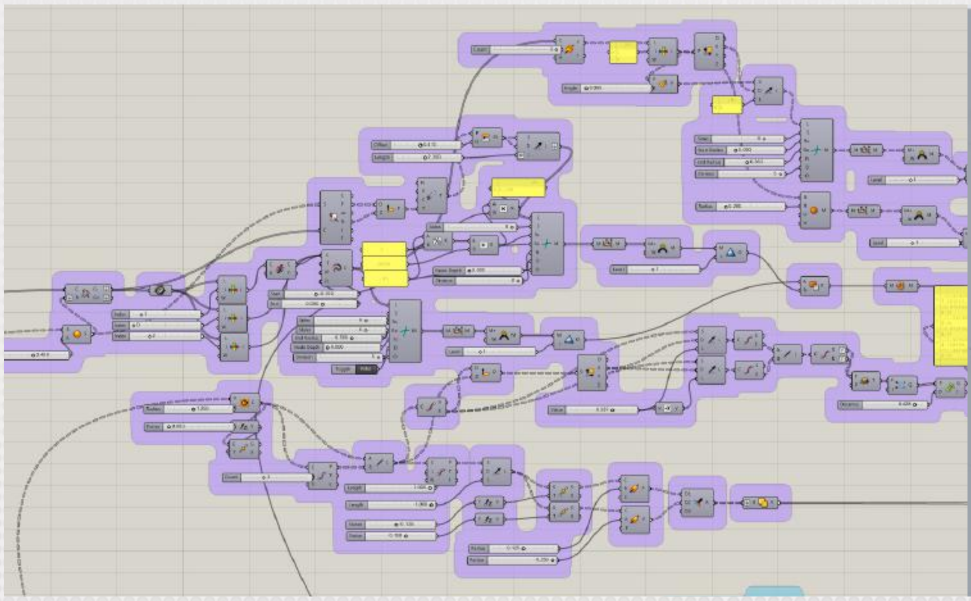
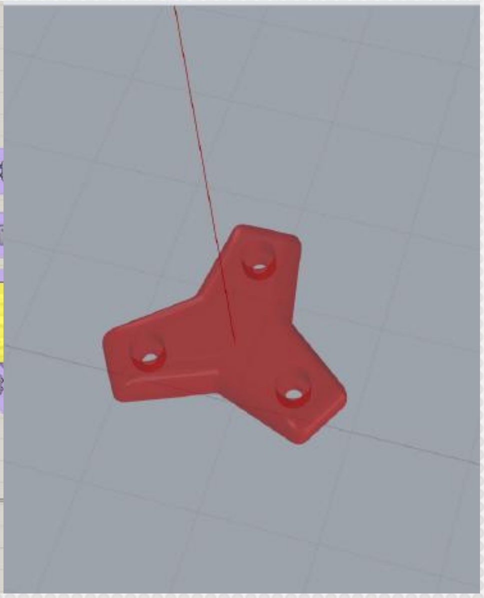
This portion of the code shares much in common with the Convex Hull generating procedure detailed earlier. The major difference is that these nodes, given that their centroids terminate at a set plane, are designed to mount directly to a formwork apparatus. The top-portion is left open so that a concrete (or some other substrate) can be freely poured through the opening and into the tensile formwork. The definition controls every geometric aspect of this node, such as the radius of the opening and the length and thickness of the mounting brackets (though the brackets proved to be fragile).



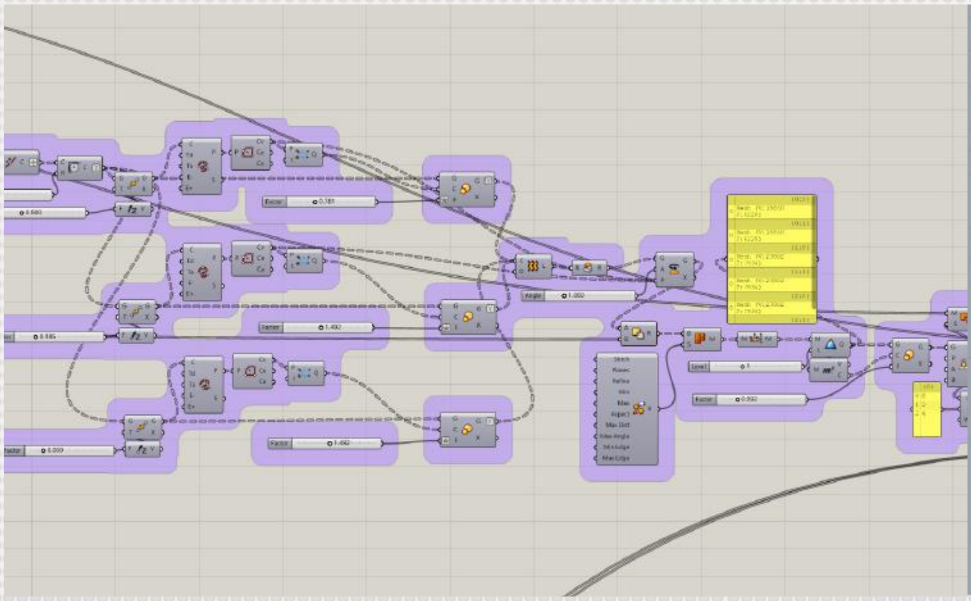
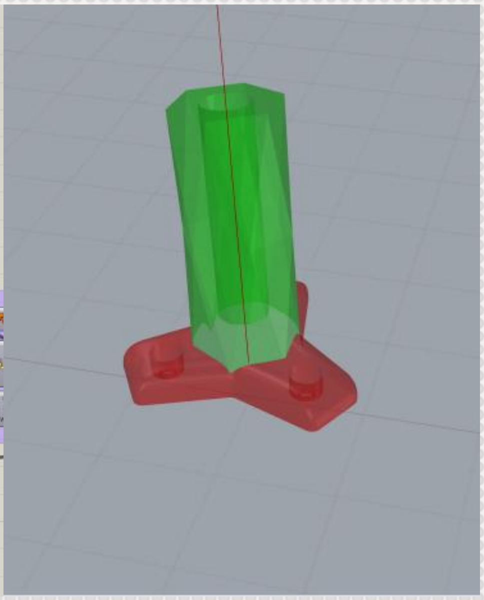


Mileage with this portion of the definition may vary, as this bracket configuration was designed to function with a very specific formwork. The code had to be developed very rapidly as well, resulting in a what is, animatedly, very complex and overwrought structure. Nonetheless, this code can be used as a basis for developing other fully parametric geometries, offering up, of course, the advantages of rapid alteration and experimentation.

07 LOWER PORTION STRUT NODES

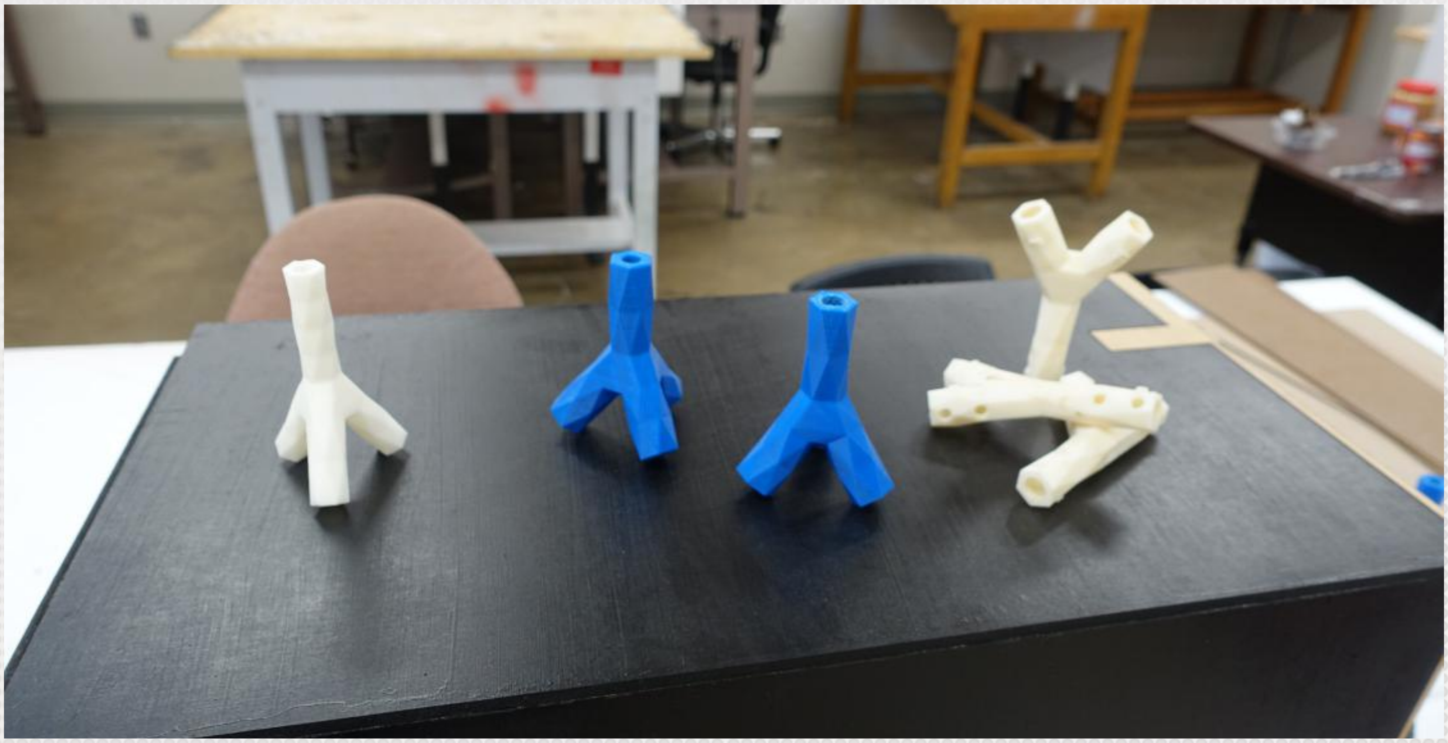



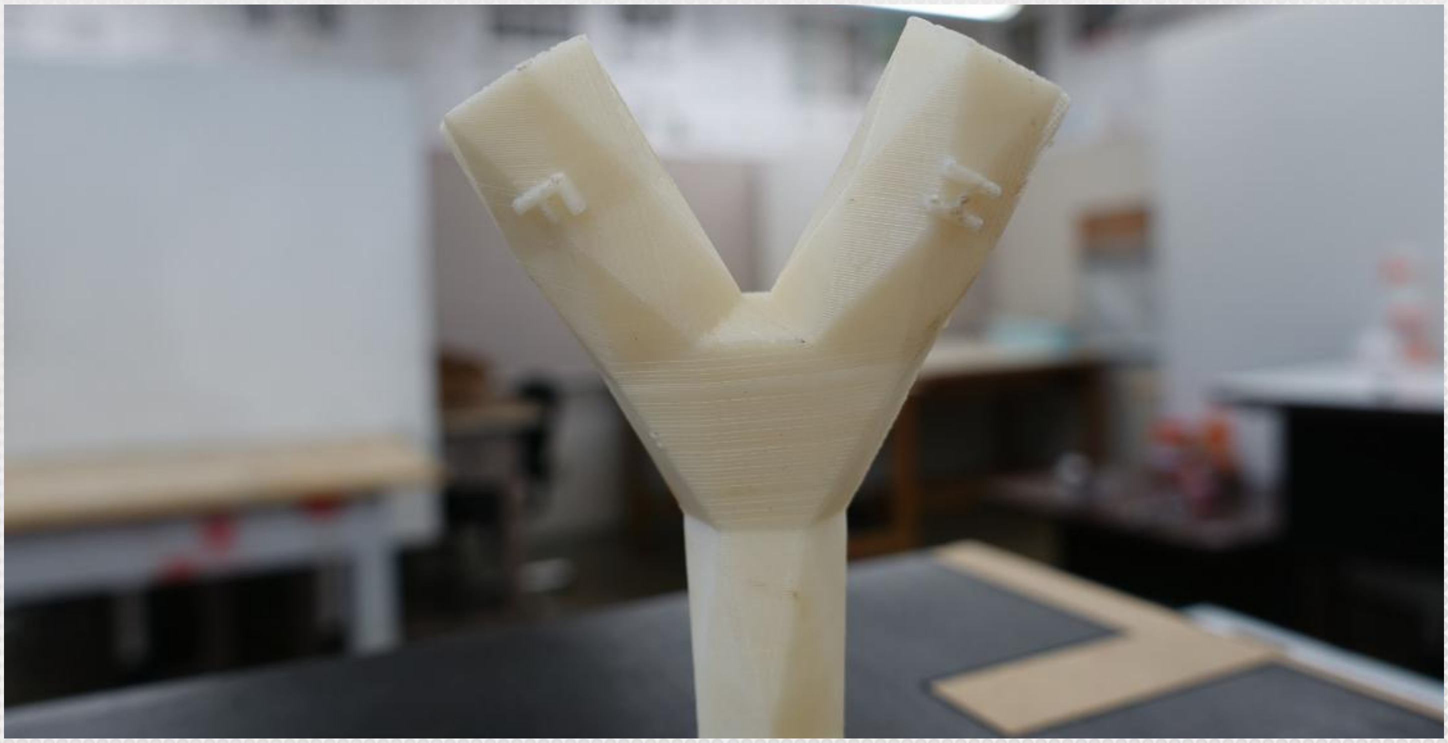
Much like the top nodes, this cluster utilizes much of the same componentry from the hull configuration portion of the definition. However, There is, again, a fairly elaborate methodology for generating the parametric geometry of the base portion of the node. The base node is intended to be fixed directly to the casting formwork, and it was important to provide openings for either bolts or screws. The initial geometry featured a simple, circular base, but it was thought advantageous to reduce printing time by decreasing the amount of surface area that needed to be printed.

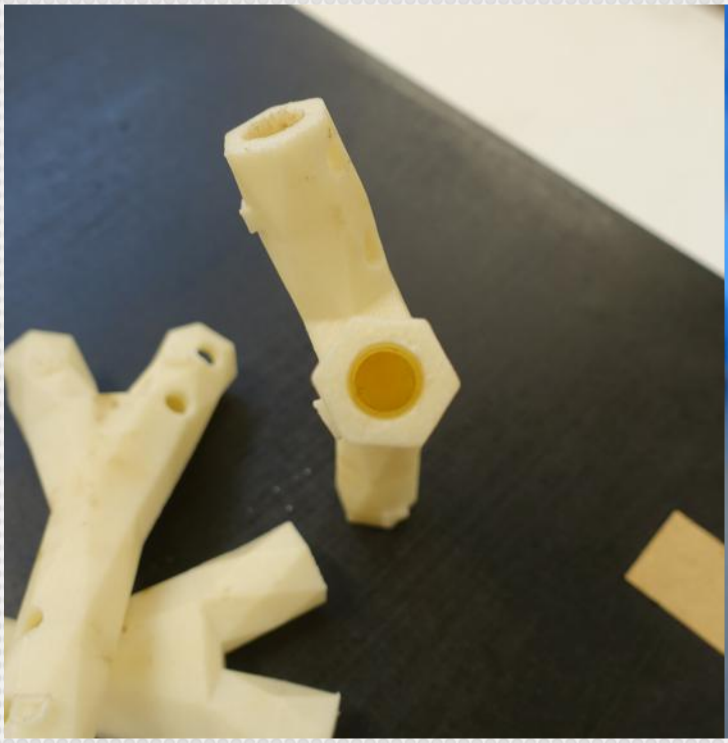



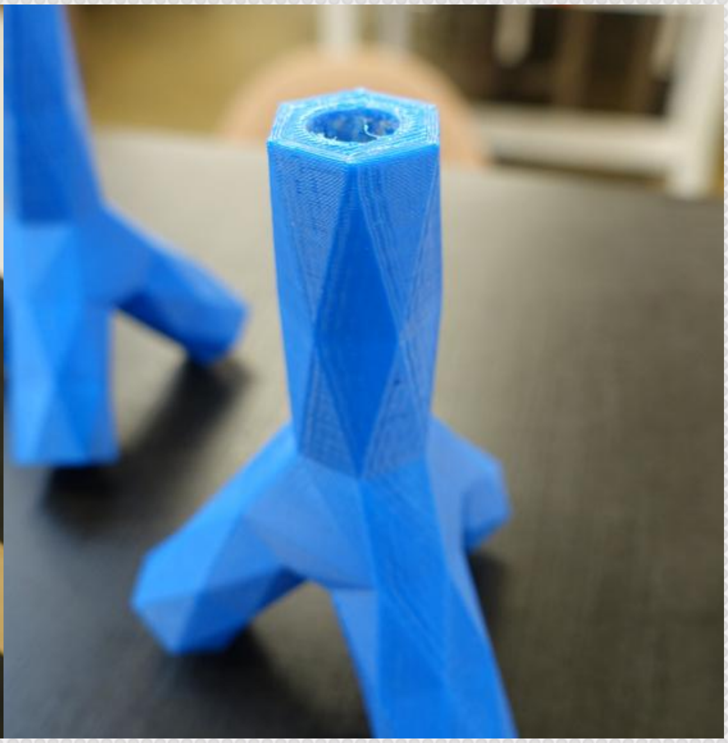
The base needs further development, and it became troublesome in some circumstances ensuring that the two geometries - the node slot and base - merge properly.

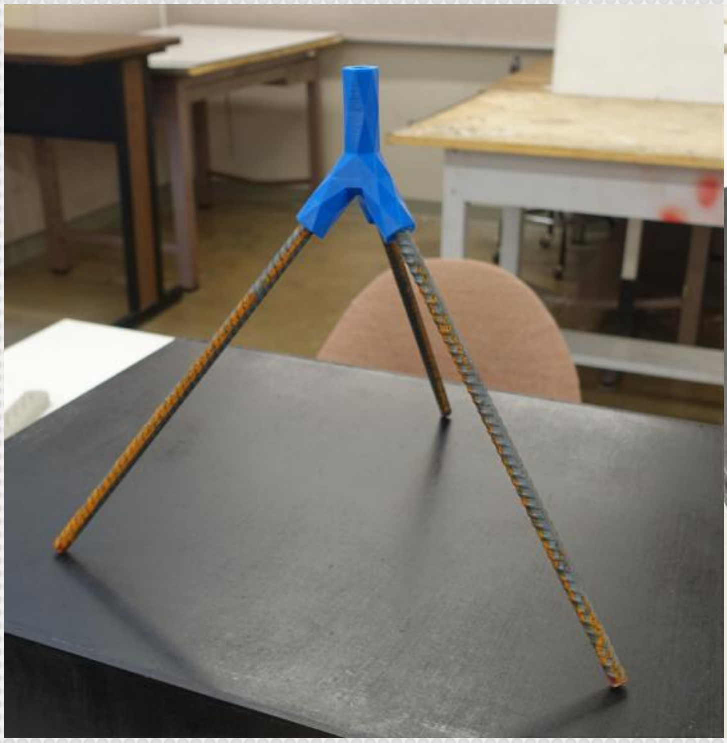
photographic documentation

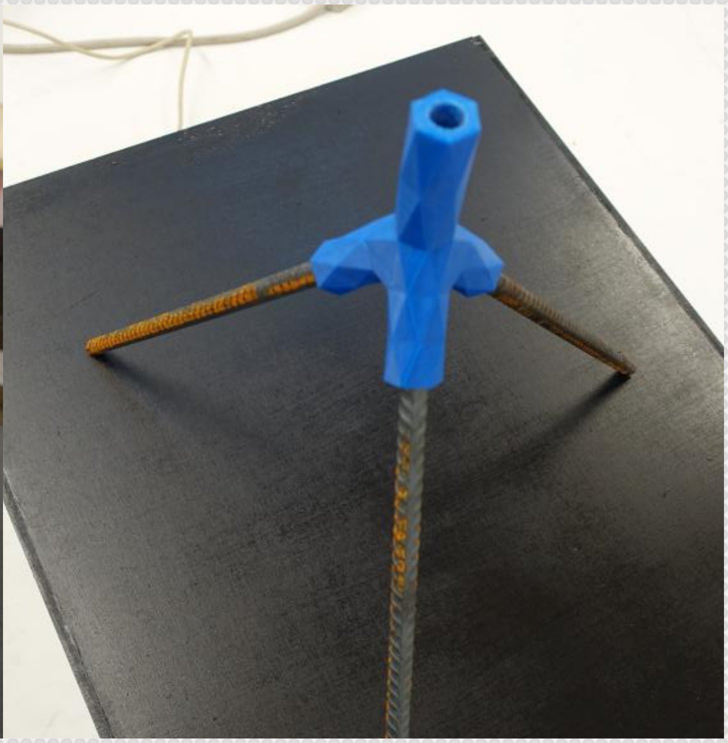


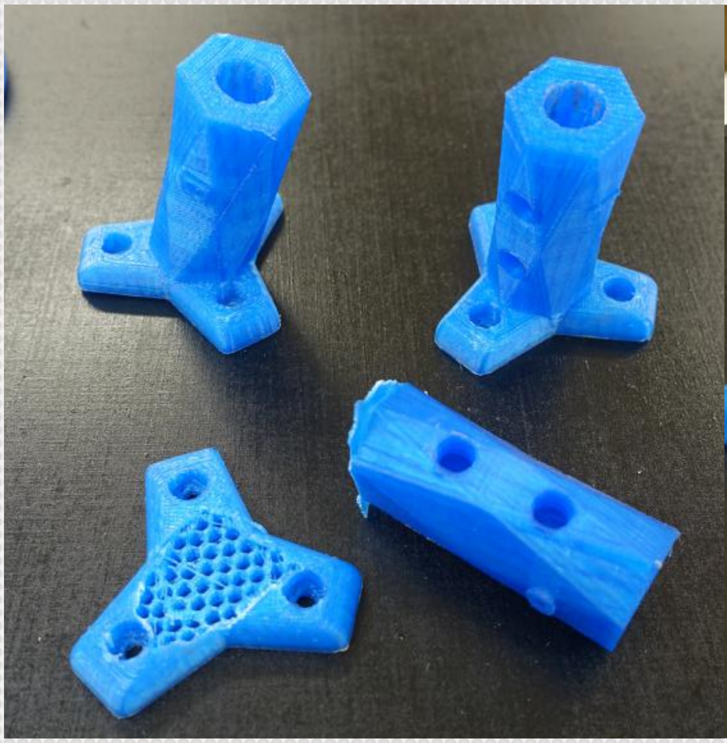


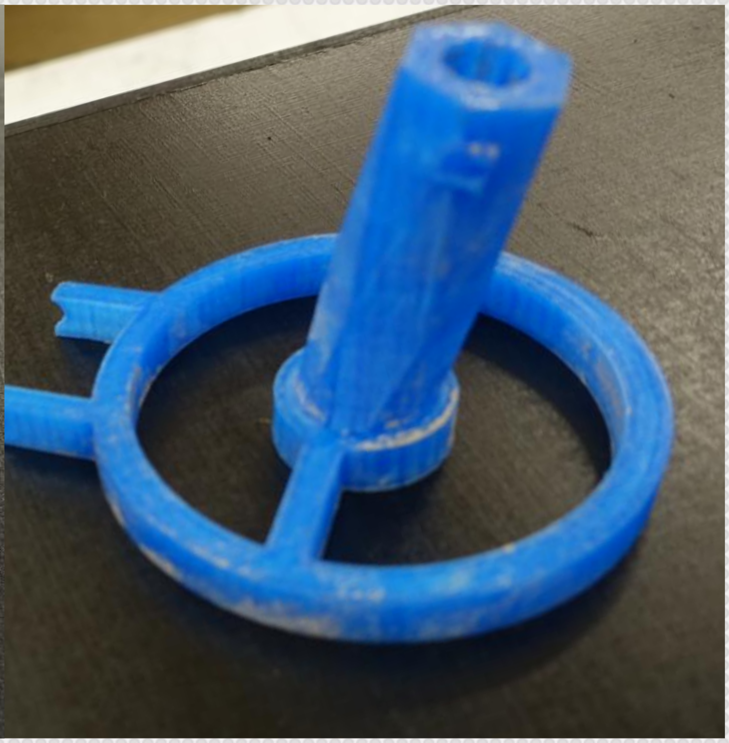


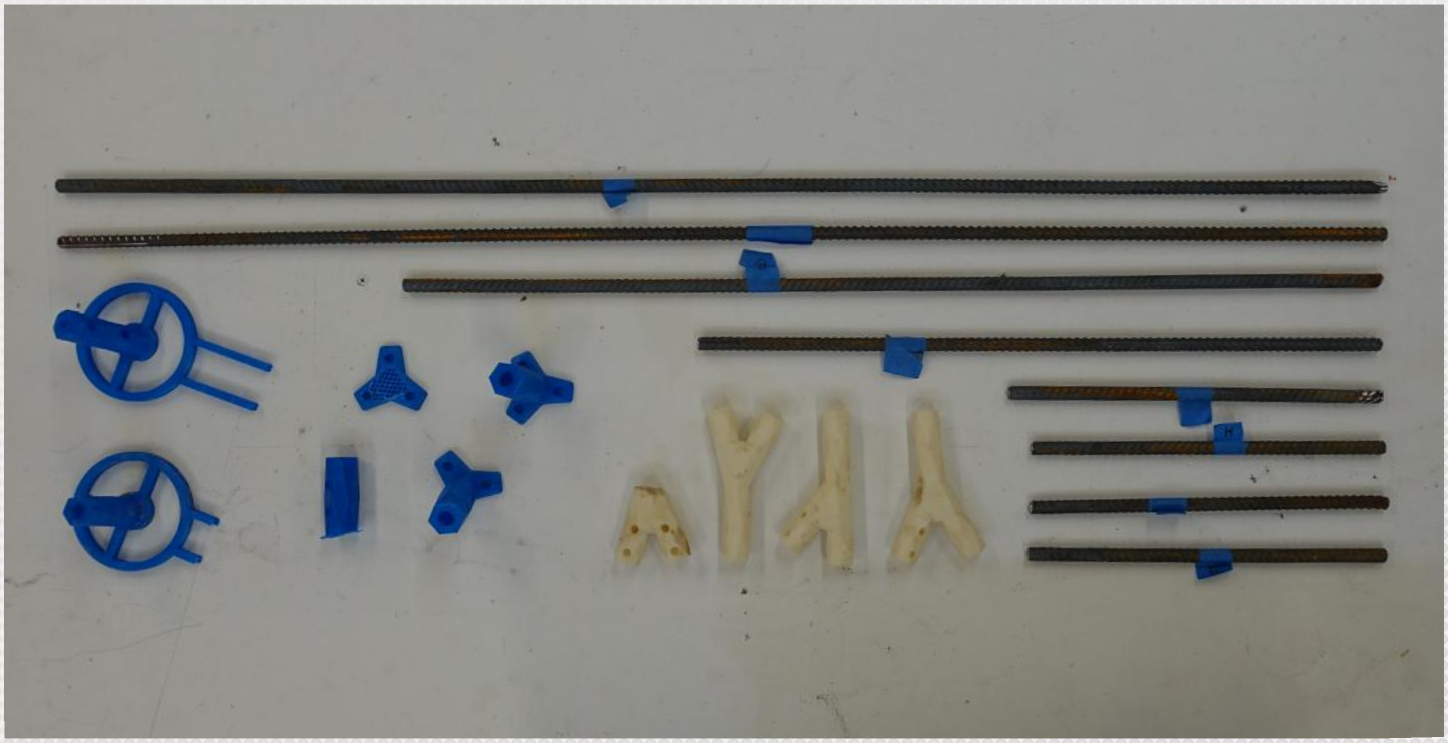


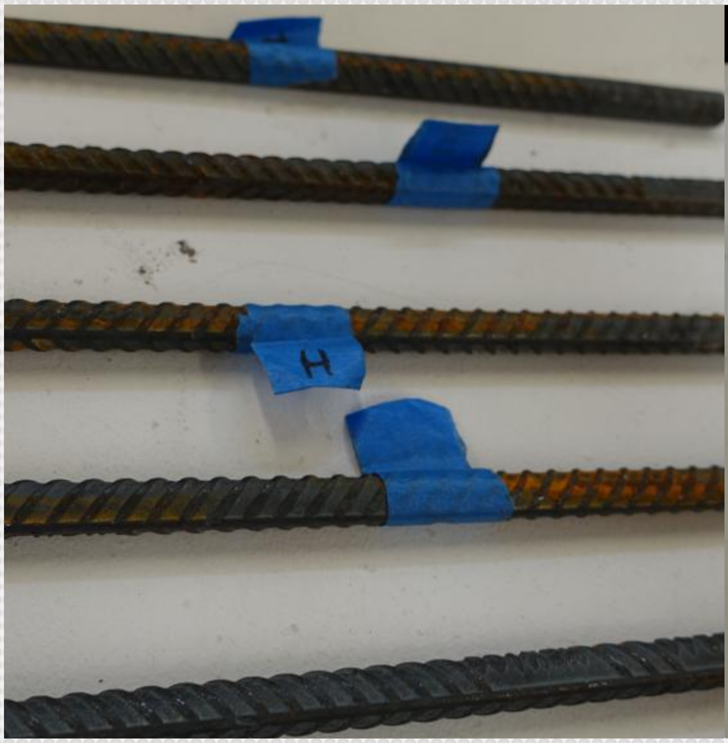


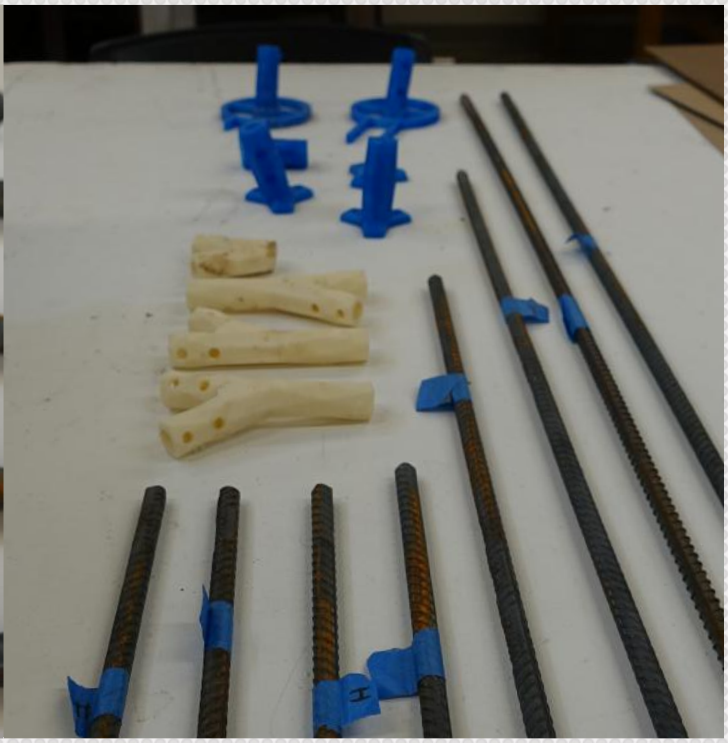


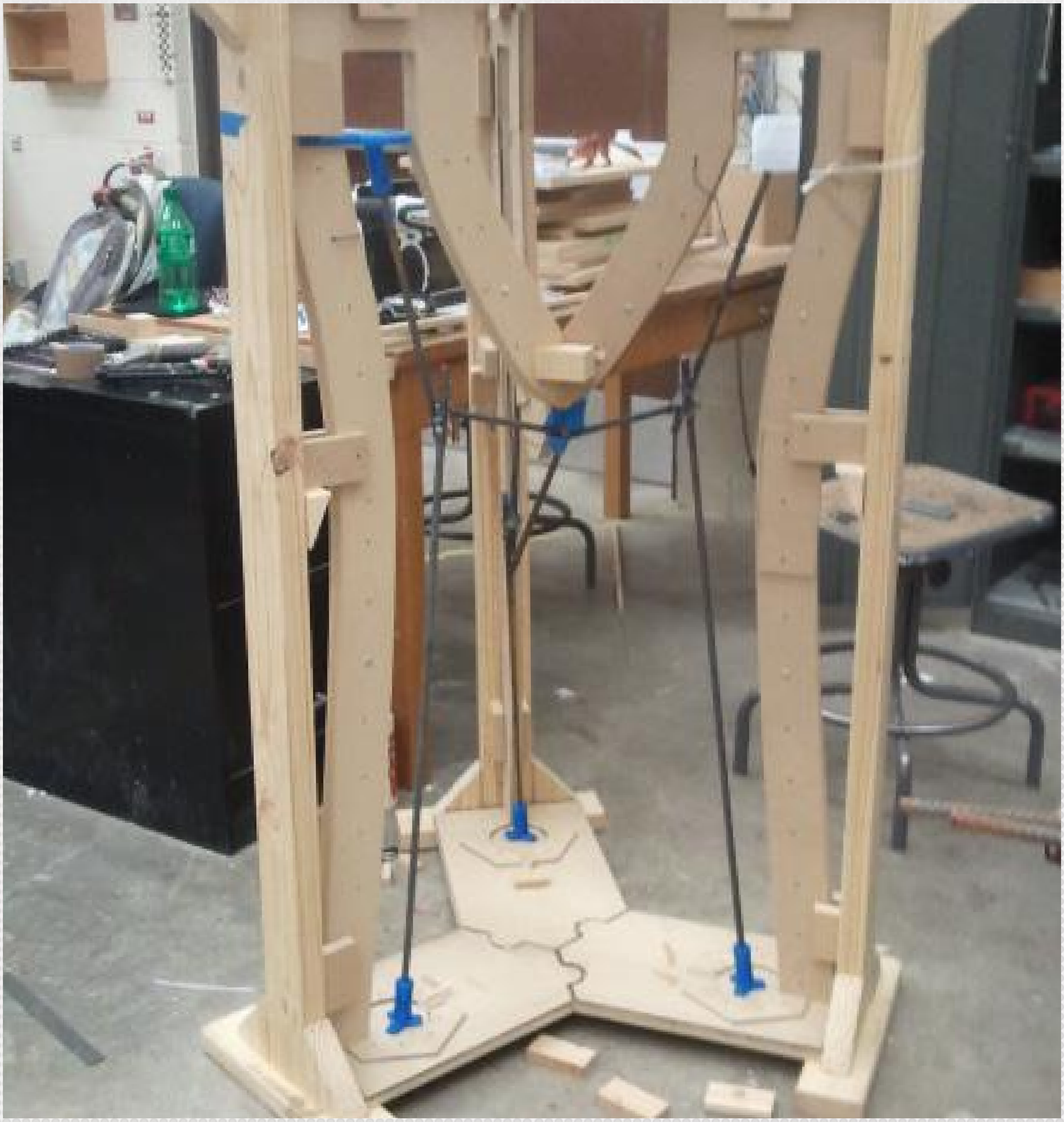


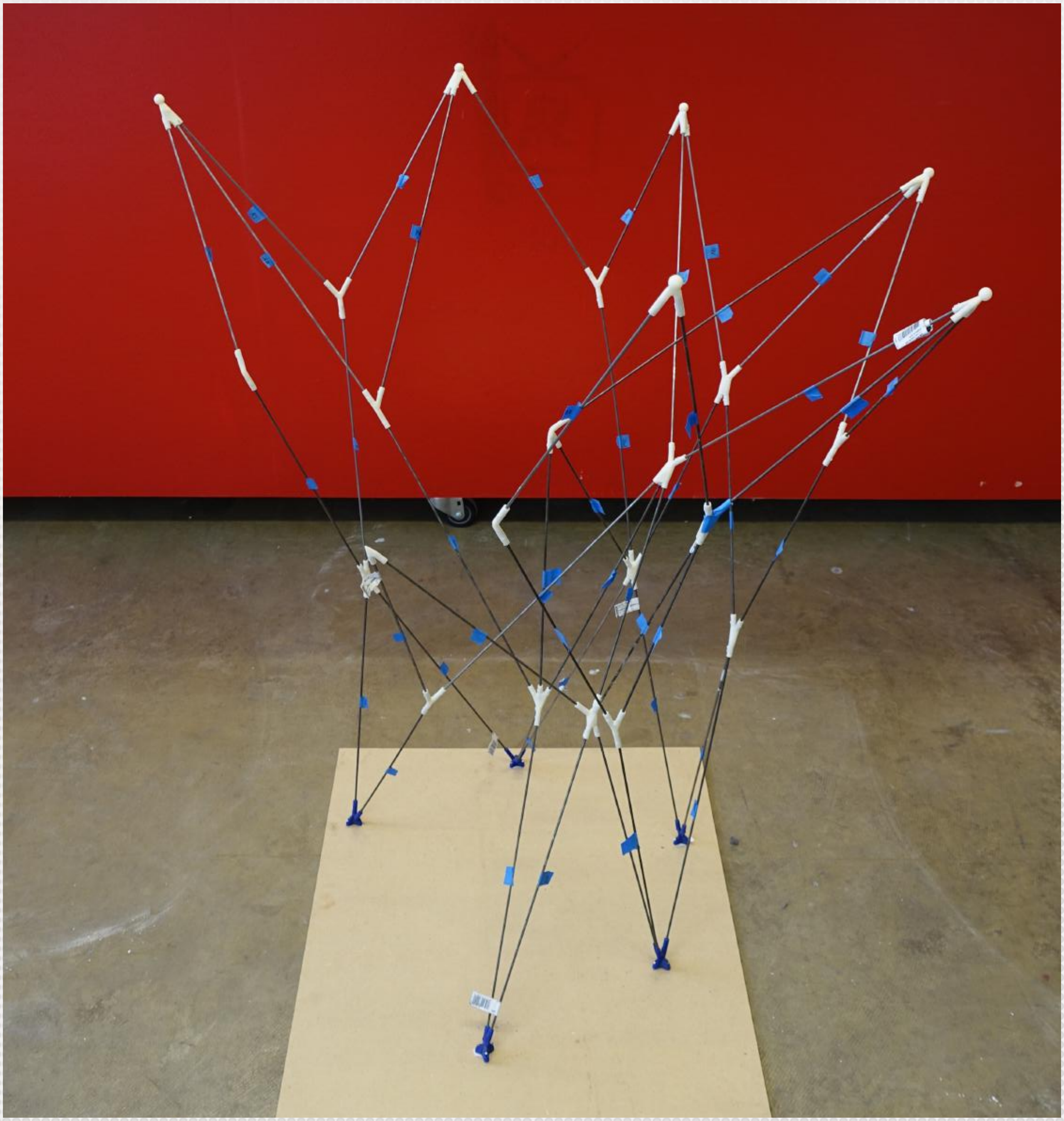


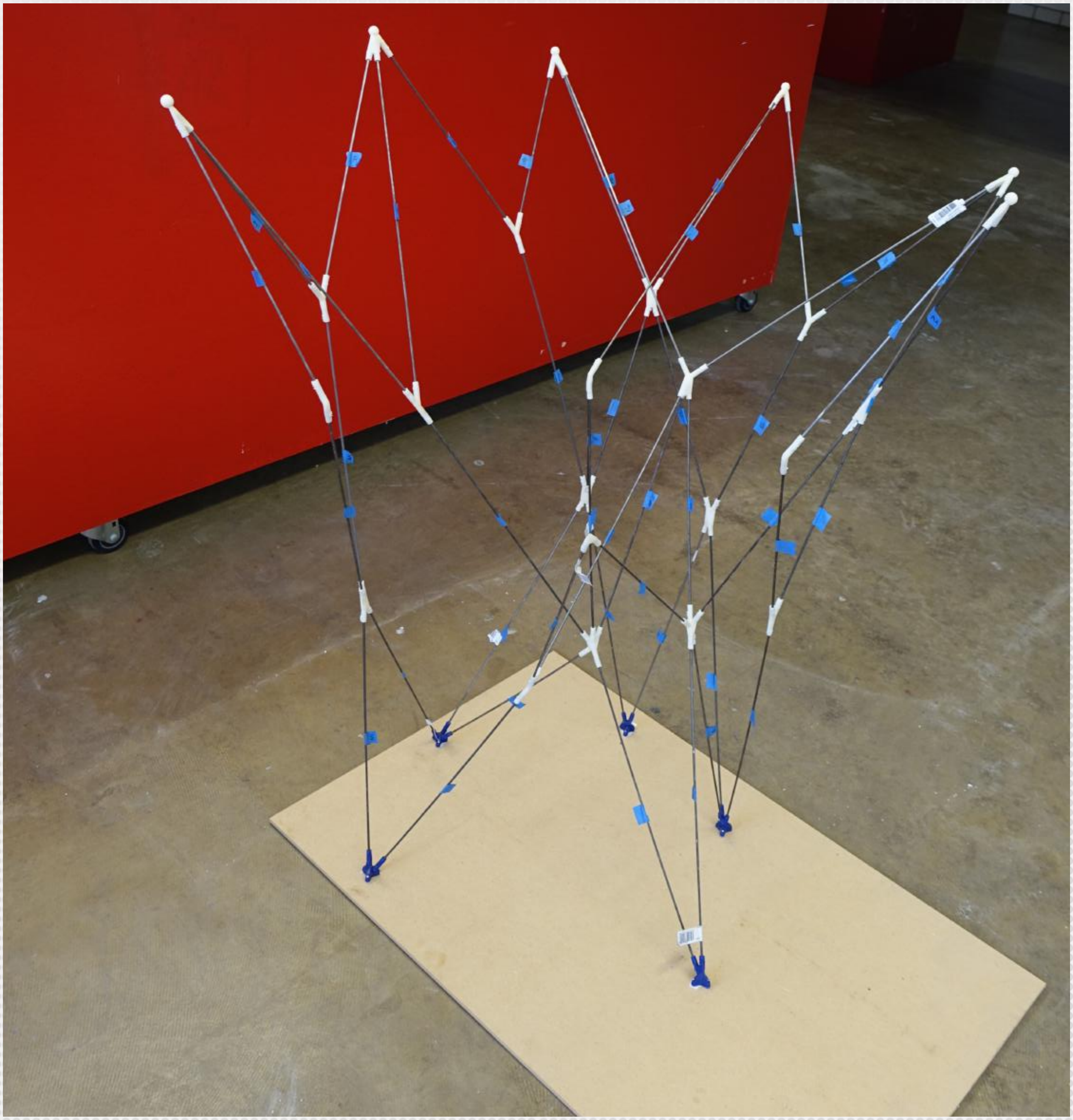

H

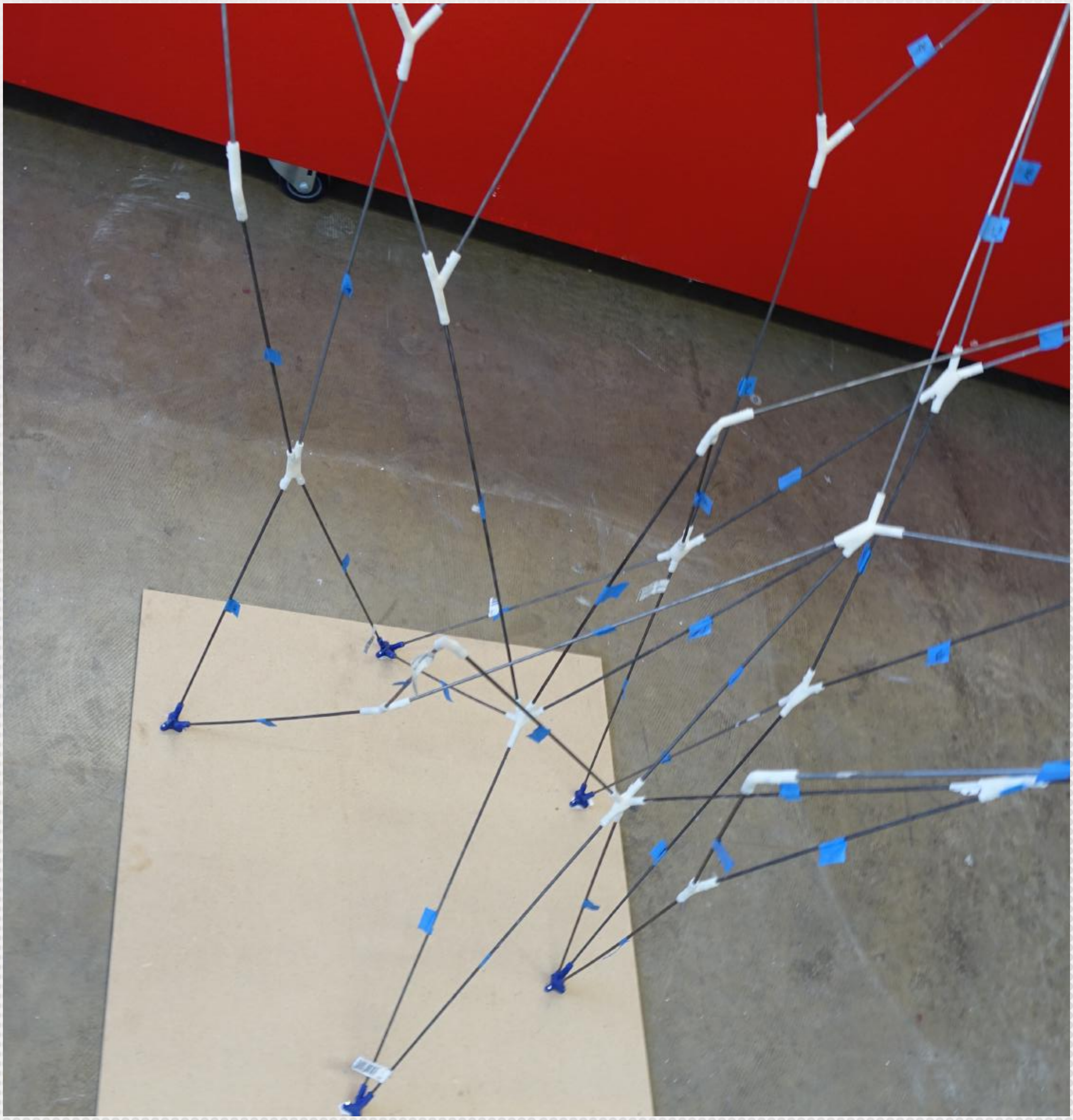

H

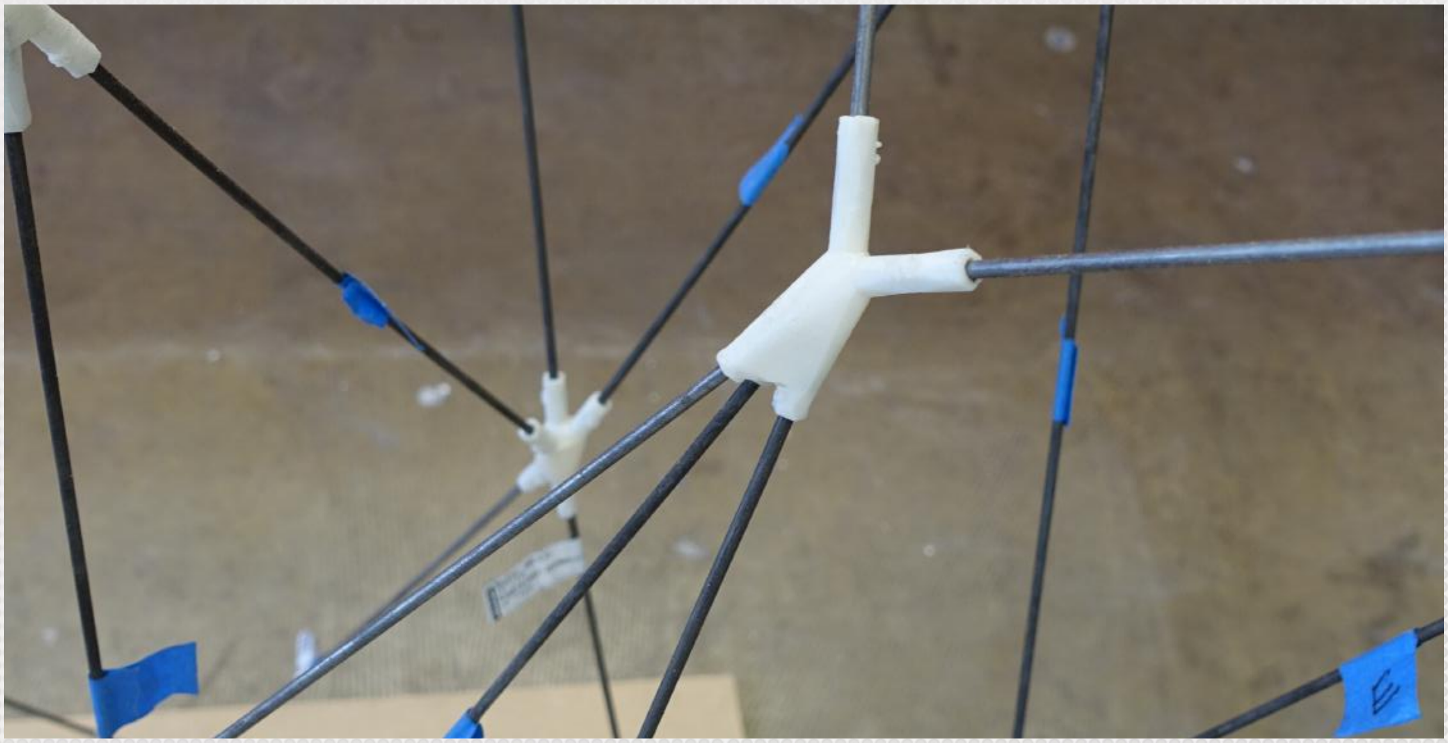

H










E