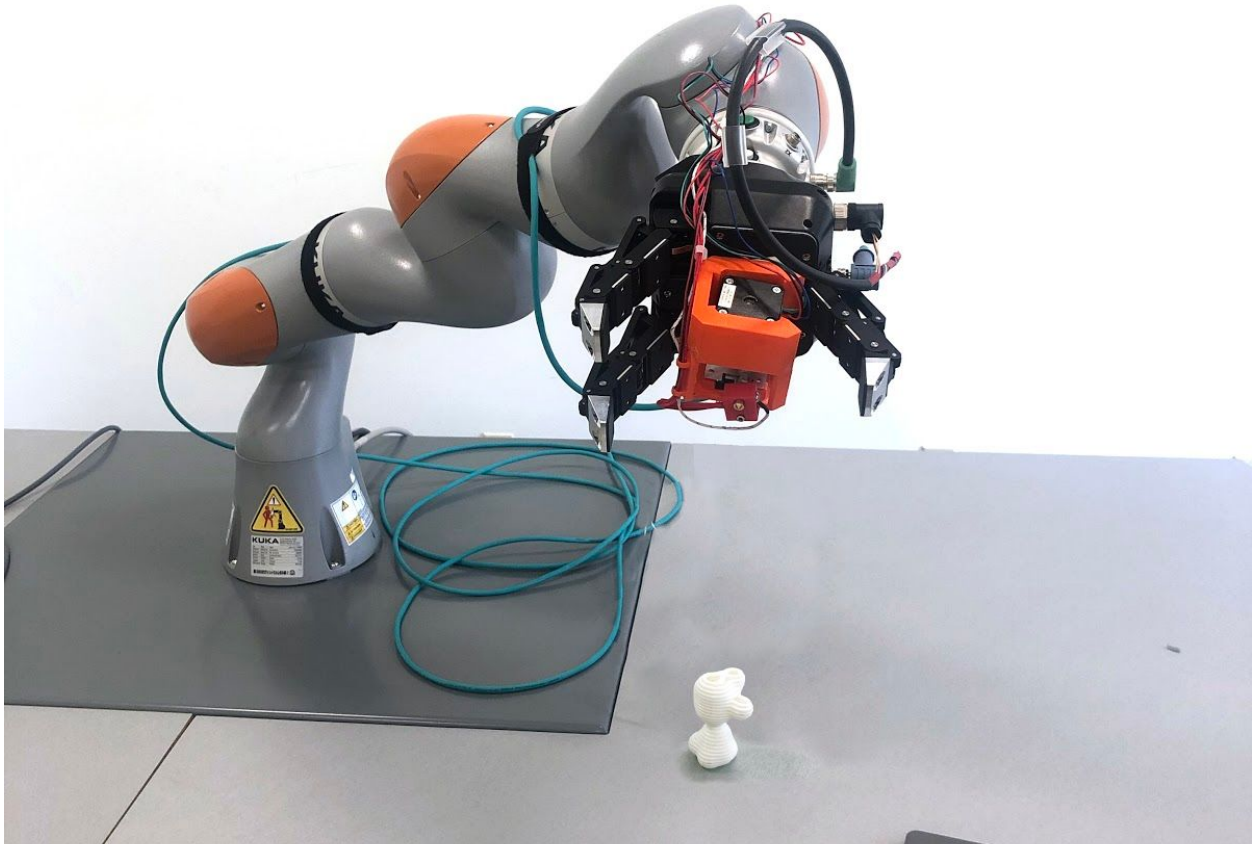
# Final Project

## Simulated Interactive Fabrication with Virtual Reality and a 3D Printer

Seo Hyun (Lina) Lim, Paul Meuser




# 1 Abstract

The simulation of Interactive Fabrication with Virtual Reality and a 3D Printer is important because it should significantly reduce the time it takes to build any object. With our own implementation of TiltBrush, our project lets the user draw freely in Virtual Reality space, and the drawing gets converted to a 3d model that is printable by a 3D printing robotic arm. The ultimate goal is to render the simulation work in real life, possibly in real time. This paper was successful in building the Robotic Modeling Assistant (RoMA) which enables a sculptor to work together with a robot arm, but is limited due to the boundaries of the drawing space and the possible shapes it can produce. Our technical approach to this

project involves a lot of exporting and importing fundamental functions from one platform to another. After exporting all the points of the drawing from Unity to Rhino, Rhino outputs it to its plugin Grasshopper which converts those points to build a 3D shape. This is then imported by Unity again so that the robotic arm is printing simultaneously. We build our own extruder which will also be open source so people can potentially mount it on any robot.

# 2 Introduction

This projects aims to be modular. We are creating a framework and a creative platform which other people can expand on and edit. Current works are not open source and inaccessible.

Our technical approach involves building our own version of TiltBrush in order to solve the issue of creating unnecessary strands of filaments and robot confusion. We get rid of the problem of safety and therefore limitations of the arm's movements and speed by placing the user and the robot in two completely different space but still allowing them to communicate. Through our own technical approach, the user also does not have to be a professional designer or sculptor. We use Grasshopper plugin from Rhino to convert the user's drawings to a printable 3D model. Finally, we have built this in simulation and therefore any risks and dangers can be eliminated before being executed in reality.

We know our approach works because it had already been given to us that it is possible for a 3D printer to recognize the virtual models and construct it in real life. The difference is that it will first all be digitalized (including the robot), and that instead of having built in 3D geometry shape modules for the user to use to build an object, it will be up to the user to "draw" the shape. We have already added a middle process that does the converting from the user's input drawings to a printable 3D format. The robot arm also does the 3D printing action in real life.

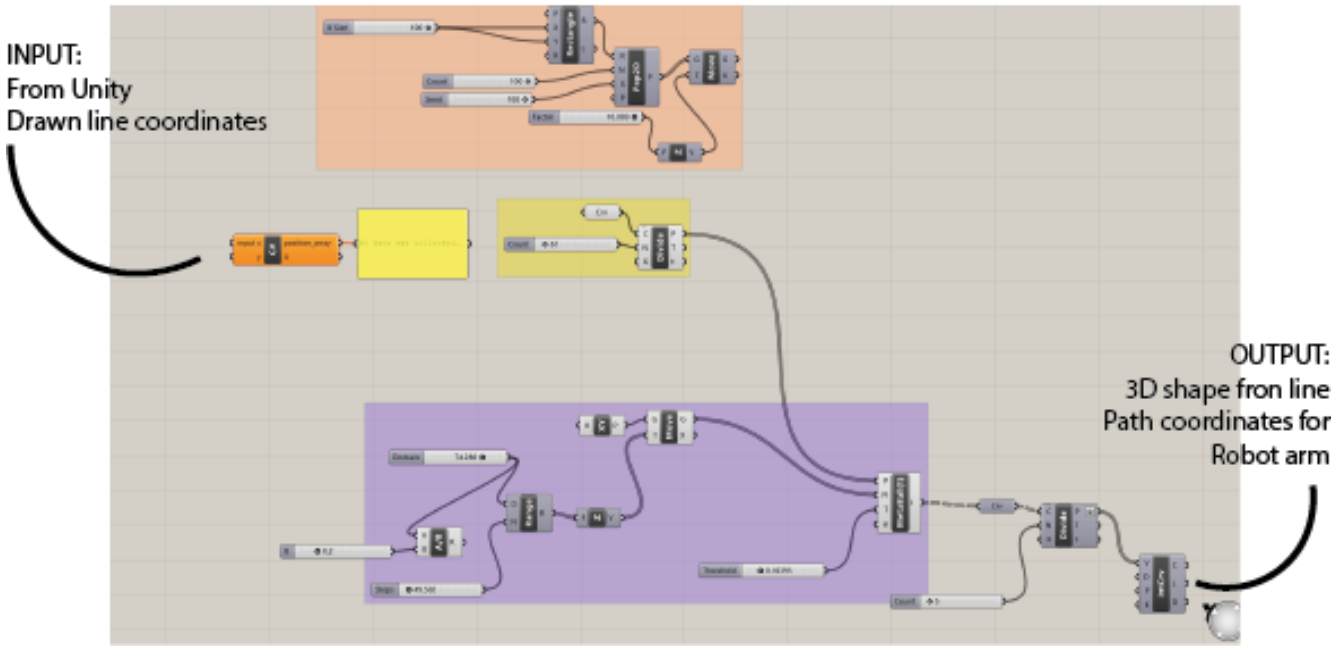
# 3 Related Work

The previously mentioned paper falls short due to the following reasons: To ensure the safety of the designer, the robotic arm's movements and speed are intentionally limited; The platform can be rotated at will by the user, while the work of the robot is unfinished, and therefore the robot may have to finish the undone parts from a new position. The robot will have to relocate, which may create unnecessary strands of filaments or even confusion; The user has to be somewhat professional in that the user has to have a prior knowledge / training with CAD or other 3D graphic model building applications, if not an actual designer; The model has to be limited to the size of the rotating platform.

Our approach gets rid of the problem of safety, since the robotic arm and the user use two completely different spaces. The user stays in the VR environment, within the area that is caught by the camera, and the robotic arm can be in a different room or space and it will still automatically be printed. This works because the CSV file that holds the coordinates is sent and saved into the machine connected to our robot Kuka.

# 4 Technical Approach

The overview of the problem was creating shapes from lines. This creates a whole new way of sculpting that is especially relevant in VR by bridging the gap between drawing and sculpting. The technical approach involved grabbing the coordinates from the lines drawn in VR and feeding them into Grasshopper in which we then generate the shape. From the shape we then generate the tool paths which we then give back to Unity which then feeds it to the move it file for the robot.




Grasshopper input to output conversion

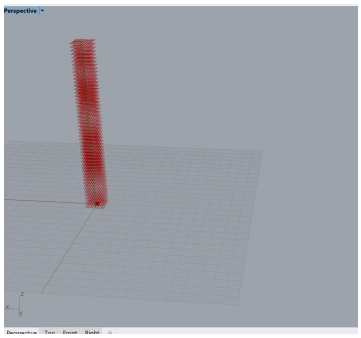


Fig.1

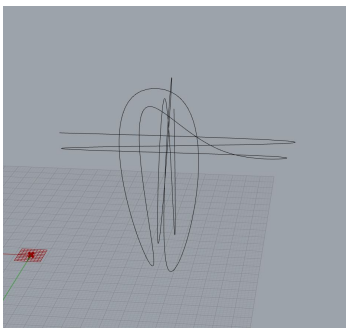


Fig.2 (Line XYZ)

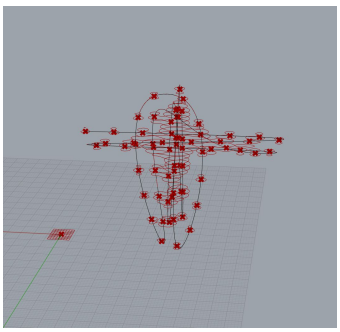


Fig.3 (Points on Line)

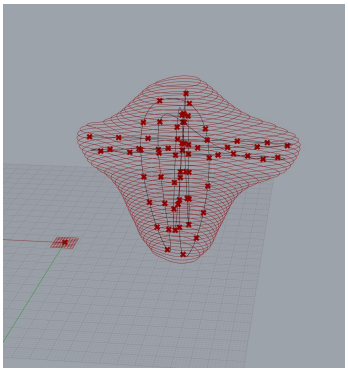


Fig.4 (Metaball Shape)

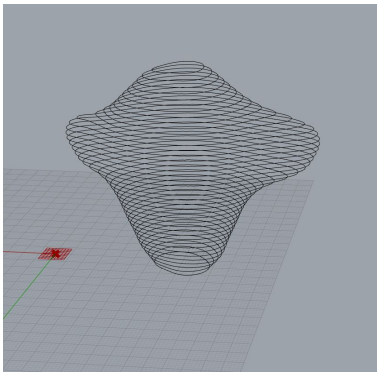
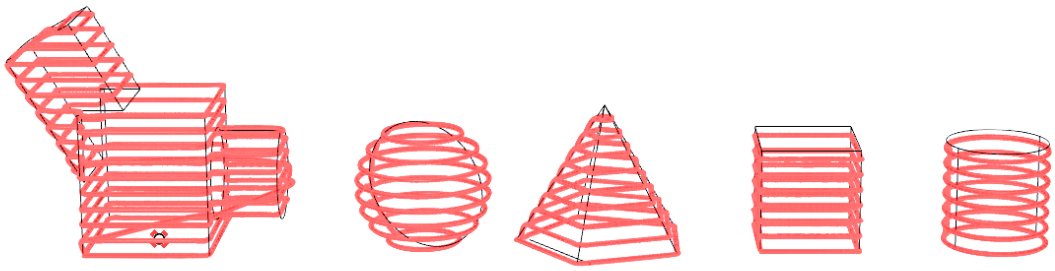
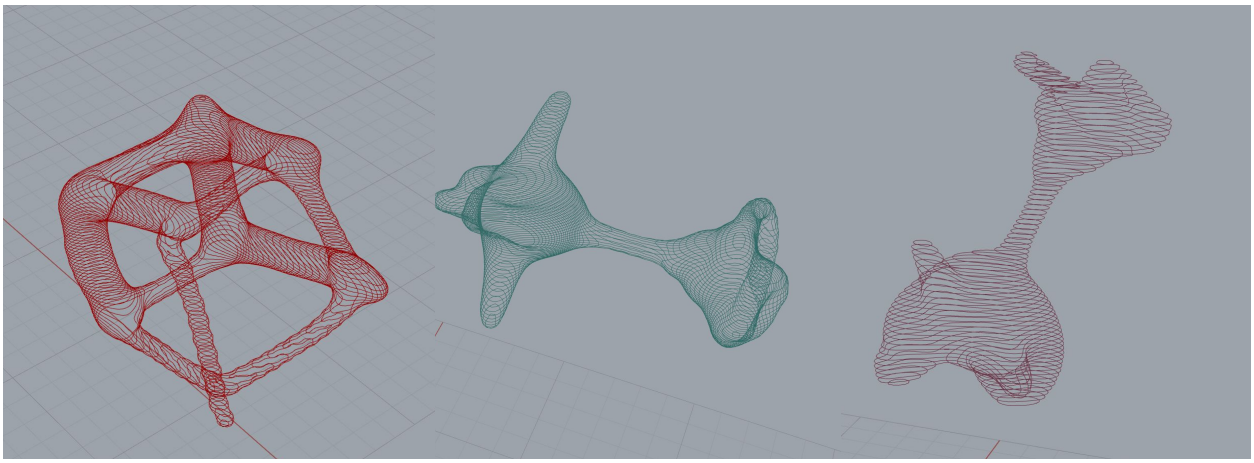


Fig.5 Pathway for robot

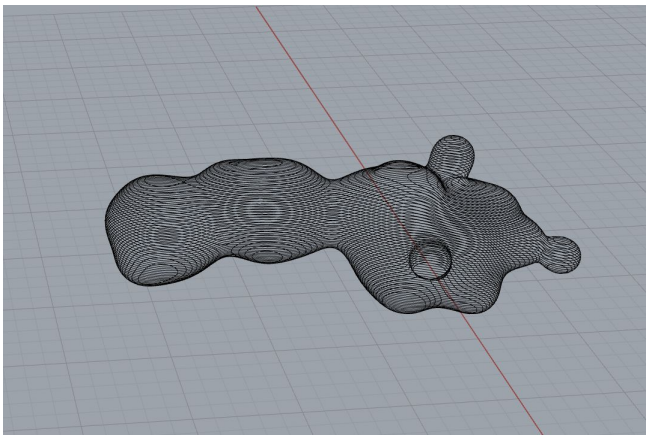
Using this method we were able to create our own printer paths for any shape. Since we are using Rhino(grasshopper) as the middle programm we are able to access all modeling tolls available in the program.



We can thus create printer paths for any kind of shape. Using rhino we are also able to merge solids into bigger more complex shapes. Since grasshopper is a parametric modeler we can generate the tool paths live and are able to update them with the shapes themselves. On top of this framework we built our line sculptor in which the user can use the VR controllers to sculpt in a 3D space. The coordinates are then fed to Rhino in which we can manipulate many aspects of the sculpt. For example the orientation or thickness of the sculpt. We are able to create carious non uniform shapes this way. We were able to design the program to also allow for shapes to be rotated, scaled and distorted after they were created in the modeling software. As can be seen in the green and dark red print paths in the following figure.



(Left: A shape with hollow inside, shows the variations in thickness dependent on the way it was drawn. Center + Right: Shows a Non uniform shape and the toolpaths generated in different orientations



(Left: This object was created using not using a continuous line but many shorter motions in a sketchlike manner)

# 5 Evaluation

The goal in building the technical approach was to 1) build a virtual scene for the user to draw in (build our version of Tiltbrush) 2) gather the values of the points from the lines drawn by the user, put these values in an array, put the array in a CSV file, amd export this array into a different platform called Grasshopper 3) With these inputs, model a 3D shape within Grasshopper plugin of Rhino 4) With the 3D shape, build a printable pathway for the robot arm 5) export this pathway into the machine connected to Kuka 6) With the input pathway, write python code that ultimately makes the Kuka perform the printing action in MoveIt 7) get the simulation on Unity! Meanwhile, another personal goal we wanted to do was to build the 3D printing hardware using the printing nozzle and attach it onto the arm so that it actually prints. We know we achieved our goal because the user is able to draw in Unity (in VR), the coordinates are saved, the 3d shapes are created, and finally, the arm goes through the printing motions as shown in the following video demonstration:
https://drive.google.com/file/d/1qrbxkuBxhZjKU6cUiVZTXXV_fqQk325J/view?usp=sharing
We have met all the goals, and are now working with the nozzle, and it will hopefully be printing soon.

# 6 Conclusion

For this project, we create our own version of Tiltbrush and build an algorithm that lets the user produce 3D drawings in a virtual environment. At the same time, the user will be able to edit and add onto this painting, and update the model in real time. The user uses it to paint in a 3-Dimensional space within a virtual environment, and given this 3D model, a 3D printer -- which is also in the virtual environment -- starts printing out the same model real-time. This will all be done in simulation and the final project would be uploaded online so that anyone can use and modify. The objective of this project would be to successfully print out the model from the robot we've built and make the program available for a wide range of audience.

We are still trying to figure out the best way to develop an algorithm for real-time implementation of our project. As of now, our program simply updates the csv file of coordinates, and therefore is not efficient enough. Deleting parts hasn't been implemented yet as well. To make the concept of real-time even closer, in the future, we may introduce the "lock" and "unlock" system to our algorithm so that the user can make modifications during the process of drawing and printing by locking the model for the printer to start printing, and unlocking certain parts the user wants to edit. Only the locked parts will be printable. This way, the locked parts will begin to be printed as soon as possible. Although locking and unlocking parts of the drawing improves our performance by decreasing risks of robot confusion -- due to deleting and adding new parts to the model -- it is still not perfectly real time because the robot arm will have to wait until the user decides he/she is fine with locking a part. We also have to develop algorithm for letting the user know which parts he/she can unlock by getting data from the robot side, as well as the algorithm to figure out which section of the model would be fit to be the base for printing. We plan on using normal vectors to do this. A problem of having a base is that the user will not be able to edit anything below the base, and thus the users have to know exactly what they are drawing so that they can select a part of their drawing model to be the base, or the starting point.

For the final project, we have to finish exporting the Grasshopper algorithm to Unity, the platform where we combine the user and the robot sides. In the user's side, we will have to come up with algorithms for locking and unlocking and for selecting base. Then, we have to accomplish the robot side of the project. This should be relatively easy, since we already have the printable file outputted by the Grasshopper, and simply need to insert it into the 3D printing nozzle attached to the robotic arm.

In the long term, there are many more ways to improve our project. We can improve performance of the real time and develop better algorithms for converting from the user's drawings to 3D shapes for accuracy of the model. Since our project will be up on an online source, anyone can modify and improve our project in the future as well.