# Efficient Meshes from Point Clouds for Tactile Internet

**Willem Stuijt**
**Supervisor(s): Kees Kroep, R. Venkatesha Prasad**
**EEMCS, Delft University of Technology, The Netherlands**
22-6-2022

## Abstract

We can see and we can hear through the internet, but not yet touch. Tactile Internet (TI) is the paradigm that will enable the transmission of tactile feedback through the internet. TI requires Ultra Low Latency (ULL) networking to prevent desynchronization. Reaching the ULL requirements for medium to long distances is not possible without breaking the known physical limits of the speed of light. We propose instead to bypass these requirements by running a physical simulation of the controlled environment to perform predictions before the information can arrive. To this end, efficient low-polygon 3D meshes from the objects in the environment are required for physical simulation. The focus of this work is on reconstructing these meshes from point cloud scanners. Through a combination of Marching Cubes [10] mesh reconstruction and Hoppe's Mesh Simplification [6], point clouds with up to $20\,\mathrm{mm}$ Gaussian noise can be accurately reconstructed into compact low polygon meshes.

## 1 Introduction

Right now the internet is limited to the transfer of visual and auditive information. Yet humans are an incredibly tactile species, especially when it comes to the execution of motor tasks. Tactile Internet (TI) aims at making the transmission of tactile feedback in real-time from remote locations around the world a reality. This would enable a whole new set of use cases for remotely operated robotic actuators, such as for remote surgeries or maintenance of inaccessible satellites through remote-controlled robotic arms.

The general use case of a TI application involves a human operator, using advanced sensors and actuator devices. These devices include sensing gloves, robotic arms, and exoskeletons among others. The motion input from the user is then transferred over the network and received by the controlled environment. The received input is then used to execute the desired intent of the human operator by robotic actuators. The controlled environment must then communicate with the operator the updated environment as well as force feedback to keep the system in sync.

While promising, TI still poses various challenges. The biggest one is the ultra-low latency (ULL) requirement of sub 10ms round-trip latency [5]. Above 10ms, the quality of feedback between the human operator and the controlled environment becomes of poor quality. While advancements in networking and innovations in 5G are making drastic improvements on latency times, physical laws still set a ceiling for maximum improvement. Even if networks transmitted information as fast as the speed of light in a vacuum, the round-trip latency of communications between opposite parts of the globe would still be above 200ms [4]. But what if the tactile feedback information could be known before it is received?

A physical simulation of the controlled environment running on the master environment could be used to predict the results of the human operator's actions in real-time, thus bypassing the ULL requirements. This creates a whole new set of challenges. Such a simulation would require sophisticated synchronization algorithms. These would keep the controlled environment up to date with the simulation despite medium latency and package loss while making sure the human operator does not notice that the physics with which he is interacting is simulated. The simulation would also require a robust method of transforming the real-world environment into a digital environment that can be used for computations. This is what this research project is mainly focused on.

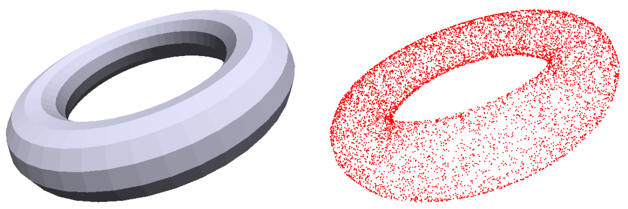
Figure 1: On the left is an example of a polygon mesh. On the right an example of a point cloud sampled with random Montecarlo from the left mesh.

**Point Clouds** Humans and animals can reconstruct their visual environment through color only. But it is much easier for computers to do the same if the images have depth information. This is what Point Clouds are. A point cloud is a set of 3D points used to represent laser scans and depth images. An example can be seen in Figure 1. It used to be that 3D depth imaging could only be done with expensive equipment but this is no longer the case. Nowadays cheap commodity hardware that can produce point clouds is easily available for consumers, with even high-end smartphones having point cloud scan capabilities.

**Polygon Meshes** Using depth cameras in the controlled environment, the color and depth data can be used to reconstruct 3D models for physical simulation. 3D objects are usually represented as polygon meshes. Meshes are a collection of vertices and faces, with the faces generally being triangles or quadrilaterals. An example polygon mesh can be seen in Figure 1. A physics engine can then use these meshes to simulate collisions, friction, and reactive forces for tactile feedback.

The main sub-question of this research project is to find the most efficient way of generating a polygon mesh from a point cloud with as much resemblance to the original point cloud as possible. The technique should be robust against potential noise on the point clouds since commercial point cloud scanners are not perfect and have readings with various degrees of noise. Also important is the reduction of the polygon counts of the generated mesh to reduce bandwidth utilization and the processing power necessary to run the simulation.

### 1.1 Contribution

This paper brings a few contributions to the field of Tactile Internet. Mainly it describes a procedure for turning real-world object point clouds into polygon meshes for tactile simulation. Secondly, it provides insight into how much noise is tolerated by the popular Marching Cubes mesh reconstruction technique. This paper also investigates the ideal degree of mesh simplification on reconstructed meshes. This makes it possible to know how much mesh simplification is too much before the error increases in a noticeable way.

## 2 Methodology

The procedure described in this paper can be split up into two major steps. Mesh reconstruction and then mesh simplification. In this section, both of these parts will be described in detail.

### 2.1 Mesh Reconstruction

There are plenty of mesh reconstruction techniques in the literature. A survey by Huang et al. [8] benchmarks various of the most well-known mesh reconstruction techniques. Among the techniques described, Screened Poisson Surface Reconstruction [9] stands out as one of the best-performing methods, even surpassing deep-learning based models.

**Screened Poisson Surface Reconstruction** This technique consists of modeling the surface reconstruction problem as a Poisson equation. The biggest downside of this technique is that it is not designed to work well with non-watertight surfaces. This is a problem for Tactile Internet since many objects will be scanned from only one perspective if they have not moved yet, thus yielding an incomplete point cloud. And as seen in Figure 2, Poisson Reconstruction leaves some residue on its attempt to close the mesh. While it is possible to have yet another algorithm for trimming the excess, this is both computationally expensive and error-prone.

**Marching Cubes** An alternative to Poisson Reconstruction that does not require watertight surfaces is Marching Cubes [10]. The simplest version of this algorithm consists of a fixed-size cube that traverses in small steps the whole point cloud, inserting different arrangements of triangles on the reconstructed mesh depending on how many and the position of points of the cloud found inside the marching cube. More advanced versions include an adaptive cube and step size [12] which results in more detailed geometry. This is at the expense of more complex meshes and longer computation times.

As seen in the Screened Poisson Surface reconstruction paper [9], in the table comparing qualitatively and quantitatively 8 other different surface reconstruction algorithms. The Marching Cubes implementation of Hoppe et al. [7] provides a good trade-off between quality and speed. Next will the different mesh simplification algorithms be analyzed.

### 2.2 Mesh Simplification

All the mesh reconstruction algorithms generate very detailed meshes with excess triangles. This holds especially true for very simple geometries, such as flat faces and simple curves.

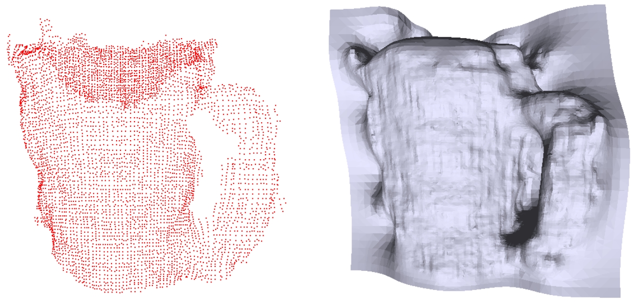

Figure 2: On the left is a partial point cloud scan of a mug. On the right is a mesh generated using Poisson Reconstruction [9]. Since the point cloud does not represent a watertight surface a residue layer forms on the outer part of the mesh. The point cloud scan of the mug was obtained from [1].

A simple flat face from a reconstructed mesh can use hundreds of triangles when it should be possible to represent it with just 2 triangles. Although properly finding simple surfaces that could be simplified in a point cloud is a very difficult task. More complex surfaces are not as straightforward to simplify. Many existing techniques deal with this problem. The most prominent will be listed in this subsection.

**Edge Collapse based Algorithms** There is a subset of mesh simplification techniques that falls into the Edge Collapse category. These techniques rely on the property of triangle meshes that an edge can be easily removed by collapsing it into one of its vertices. This simple operation can be applied multiple times until the desired level of detail is achieved. But for these simplifications to be accurate, the next edge to be collapsed must be carefully chosen to minimize the loss of the original mesh's shape. For this, different cost functions have been developed. The simplest one of all consists of choosing the shortest edges whose faces form the least sharp angle [11].

$$||u - v|| \cdot \max_{f \in T_{\mathrm{u}}}\{\min_{n \in T_{\mathrm{uv}}} \frac{1 - f_{\mathrm{norm}} \cdot n_{\mathrm{norm}}}{2}\}, \tag{1}$$

Where $u$ and $v$ are vertices in the mesh. $T_{\mathrm{u}}$ is the set of triangles that contain $u$ and $T_{\mathrm{uv}}$ is the set of triangles that contain both $u$ and $v$. $x_{\mathrm{norm}}$ stands for the direction of the normal of plane $x$.

This algorithm is very fast but not great at preserving global features of shapes. One of the more advanced techniques is that of Hoppe [6].

$$E(M) = E_{\mathrm{dist}}(M) + E_{\mathrm{rep}}(M) + E_{\mathrm{spring}}(M), \tag{2}$$

Where $M$ is a mesh. The $E_{\mathrm{rep}}(M)$ term is a penalty for the number of vertices in the mesh. The $E_{\mathrm{spring}}(M)$ term is the spring energy and it is introduced for regularization.

Instead of using a cost function, Hoppe attempts to minimize an energy function through well-chosen edge collapse operations. While at first sight Hoppe's mesh simplification seems prohibitively slow based on the results published in the paper (simplification taking tens of minutes for a single,

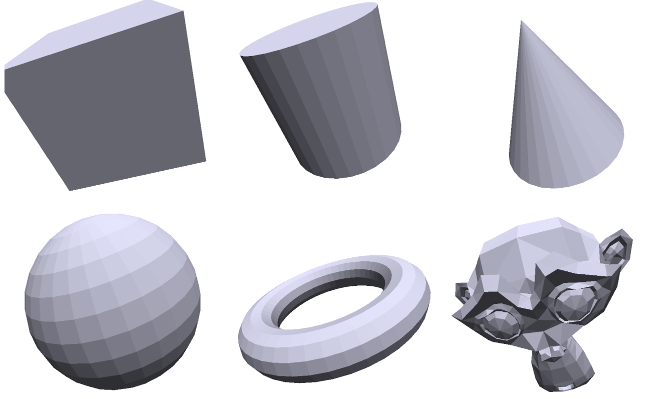
Figure 3: Original meshes used for testing. All shapes were created with Blender [3]. Top row: cube, cylinder, cone. Bottom row: sphere, torus, monkey.

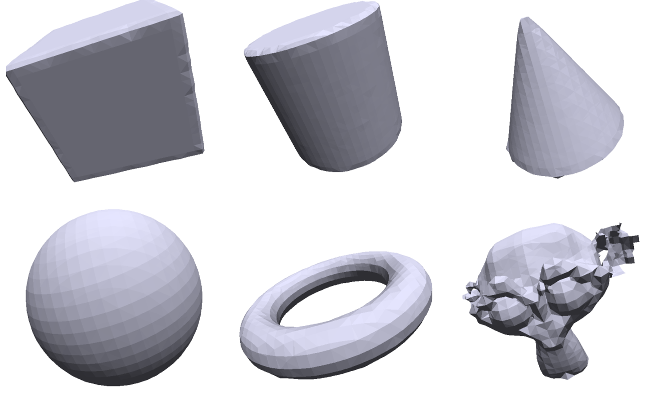
Figure 4: Reconstructed meshes from sampled point clouds with no noise and 0.1 sampling distance.

medium mesh), this paper was originally published in 1996 and computers have become exponentially faster since.

By combining a surface reconstruction algorithm with a mesh simplification algorithm an accurate yet efficient mesh can be generated for physical simulations.

## 3 Experimental Setup

The goal is to generate, from a real object, a mesh that feels accurate to human touch and is as low in polygons as possible so that valuable bandwidth and computation costs are saved for the simulation. To properly measure the accuracy of the technique a ground truth model is necessary.

A test set of meshes ranging from simple geometric shapes to more complex geometries was used for the experiment, these can be seen on Figure 3. These meshes are primitives of the open-source 3D modelling software Blender [3]. All their bounding boxes are approximately $1\,\mathrm{m}$ wide. A synthetic point cloud will be sampled from an already existing polygon mesh using random Montecarlo sampling on the faces of the mesh. The generated mesh from this point cloud can then be reliably compared with the original mesh from which the point cloud was sampled. To test for the robustness of the solution, varying amounts of noise was added to the point cloud sampling.

To perform accuracy measurements an error metric is necessary. It is difficult to compare meshes directly from their component vertices and faces as these will not be triangulated in the same way. An error metric that works independently of the original vertex positions is necessary. The most well-studied error measurement for 2 polygon meshes that fulfills these criteria is the Hausdorff Distance [2]. The Hausdorff Distance is defined as the minimum distance between a vertex on one mesh and all the other vertices on the other mesh. Since the number of vertices can get quite low on the more simplified meshes, vertices are also sampled from the faces of the mesh. The Mean Error results shown in this paper are produced from averaging 50,000 samples of the Hausdorff Distance between meshes.

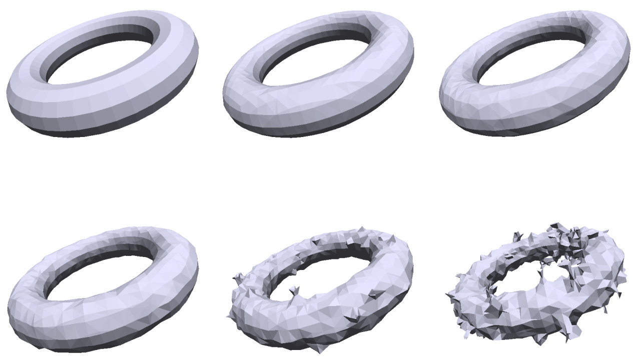
Figure 5: Marching cubes reconstruction varying noise with 0.1 sampling distance. Top Row: Original, $0\,\mathrm{mm}$ and $10\,\mathrm{mm}$ noise Bottom Row: $20\,\mathrm{mm}$, $30\,\mathrm{mm}$ and $40\,\mathrm{mm}$ noise. From $30\,\mathrm{mm}$ on wards, the noise results in significant deformations.

## 4 Results

In this section, the results will be analyzed. First the results of Mesh Reconstruction and then those of Mesh Simplification using the reconstructed meshes as input.

### 4.1 Mesh Reconstruction

On the point clouds with no noise, marching cubes generates an accurate mesh reconstruction of the original sampled mesh. Examples of the reconstructed meshes with no noise applied can be seen in Figure 4. Curved shapes like the torus are very accurately reconstructed. Sharp edges such as in the cube lose some of their detail. The monkey loses a lot of detail on the mouth, ears, and eyes but its overall structure remains.

Marching cubes, using a sampling distance of 0.1, works well with up to $20\,\mathrm{mm}$ of noise. Using marching cubes on the point clouds with $40\,\mathrm{mm}$ noise produces meshes with significant deformations. Even on the $20\,\mathrm{mm}$ noise meshes, there are still some small but significant deformations, but on av-

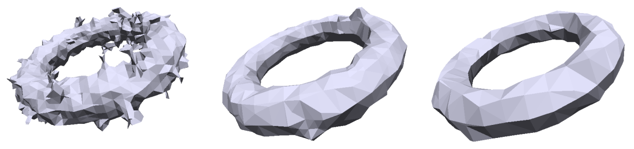

Figure 6: Marching cubes reconstruction on torus with $40\,\mathrm{mm}$ noise and varying sampling distance. With sampling distances of $0.1$, $0.15$ and $0.20$ respectively. A higher sampling distance reduces the detail of the reconstructed mesh but improves it's tolerance to noise.

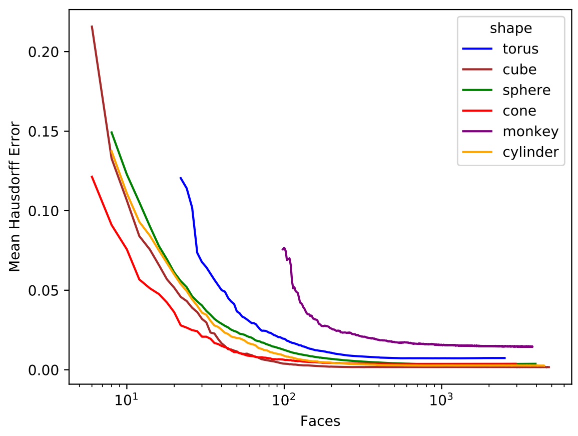

Figure 7: Hausdorff error as a function of number of faces on each mesh throughout simplification. The error remains mostly flat initially and then spikes as most of the redundant faces generated by Marching Cubes have already been decimated.

erage the meshes look well and the Mean Hausdorff Error is low. An example of the reconstructed torus with varying levels of Gaussian noise can be seen in Figures 5.

Varying the sampling distance of marching cubes can result in different degrees of tolerance against noise. The tradeoff is that a larger sampling distance results in less detailed meshes. Examples of varying the sampling distance can be seen in Figure 6. Since popular commercial point cloud scanners have a noise standard deviation of around $17\,\mathrm{mm}$ [13], a sampling distance of $0.1$ seems like a good choice for TI applications as it can accurately reconstruct point clouds with around $20\,\mathrm{mm}$ noise.

## 4.2 Mesh Simplification

After reconstructing the meshes, the next step is reducing the face count through simplification. From Figure 7 it can be clearly seen that a big percentage of the faces generated by mesh reconstruction are completely redundant and do not contribute at all to the important features of the mesh. This can be seen by the flat initial error curve. Well over 80% of the faces in each reconstructed mesh can be simplified without incurring much penalty on the error. Curved shapes like the sphere and torus are greatly simplified, even going below

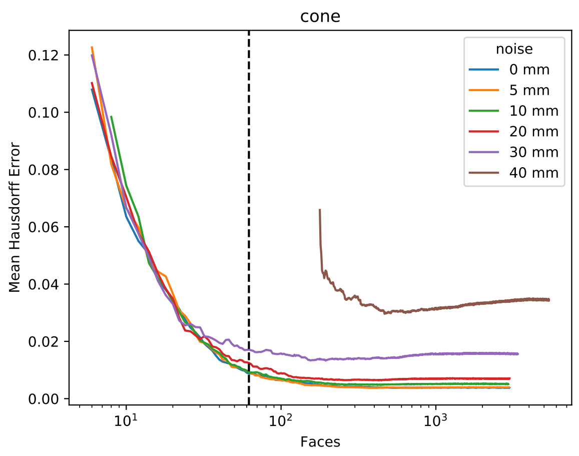

Figure 8: Mean Hausdorff error as a function of number of faces on cone with varying degrees of noise throughout simplification. The black dashed line symbols the face count of the original mesh. It can be observed here that mesh simplification even manages to initially reduce mean error on the high noise samples.

the original mesh's face count without a big increase in error. But for a sharp shape like the cube the original face count is not reached before a big jump on root mean square error. This is probably more to blame on the mesh reconstruction algorithm than on the mesh simplification algorithm since the initial meshes for simplification do not preserve the original sharp edges very well. As expected, the monkey mesh being the most complex, also has the highest amount of error. Also interesting to notice is how the mesh simplification algorithm even manages to reduce the error on high noise reconstructions in Figure 8.

One of the main questions on mesh simplification was how to know when to stop simplifying a mesh. Computing the Mean Hausdorff Error metric after each iteration of edge collapse is prohibitively expensive. Hoppe's mesh simplification algorithm aggregates the maximum directional residual, a form of error estimation, while it computes the next optimal edge for collapse. While not as accurate as Hausdorff error, this maximum directional residual can give a good estimate on when the simplification has gone too far. Looking at mean Hausdorff error plotted together with the maximum directional residuals on Figure 9, it can be seen that a good estimate is to stop simplification when the maximum residuals reach around $0.1$, this is right before the big spike in error starts to happen. An example of the $20\,\mathrm{mm}$ noise models simplified until the maximum directional residual reaches $0.1$ can be seen on Figure 10.

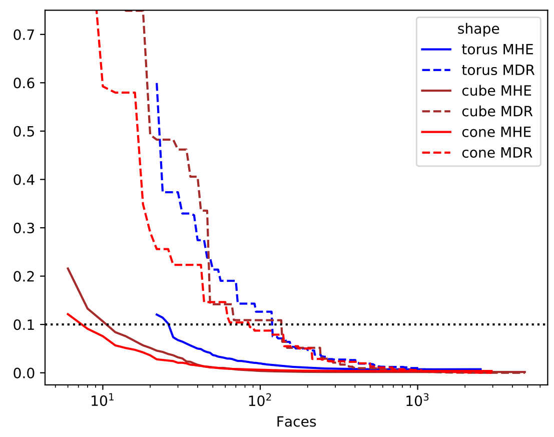

Figure 9: Mean Hausdorff Error (MHE) and Maximum Directional Residual (MDR) as a function of number of faces on each mesh throughout simplification. After the MDR reaches 0.1 the MHE starts spiking, stopping here ensures the error remains low without needing to calculate MHE on each simplification iteration.

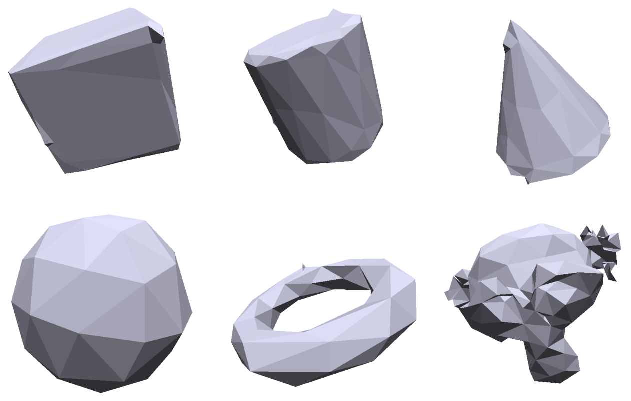

Figure 10: 20 mm Gaussian noise point clouds reconstructed into meshes simplified until maximum directional residual exceeds 0.1.

| Shape | $N_{\text{faces}}$ | Recon $N_{\text{faces}}$ | Result $N_{\text{faces}}$ | Reduction % | MHE |
|---|---|---|---|---|---|
| cone | 62 | 2952 | 104 | 96.5 | 0.00887 |
| cube | 12 | 4796 | 118 | 97.5 | 0.00488 |
| cylinder | 124 | 4476 | 158 | 96.5 | 0.00748 |
| monkey | 967 | 3770 | 926 | 75.4 | 0.0213 |
| torus | 1152 | 2512 | 134 | 94.7 | 0.0143 |
| sphere | 960 | 3948 | 116 | 97.1 | 0.0111 |

Table 1: Shows the overall results of running the complete pipeline on the 20 mm Gaussian noise test clouds. $N_{\text{faces}}$ is the number of faces on the original meshes. Recon $N_{\text{faces}}$ is the number of faces on the mesh reconstructed using Marching Cubes. Result $N_{\text{faces}}$ is the number of faces on the reconstructed mesh after simplifying until the maximum directional residual exceeds 0.1. Mean Hausdorff Error (MHE) is the average error of the resulting mesh. The resulting meshes can be seen on Figure 10.

## 5 Responsible Research

It is important to ensure the reproducibility of the results described in this paper. To ensure this, the source of the original test meshes was described in the previous section, as well as the methodology used for error measurement. All methods used from other works are cited and all the algorithms used are specified along with their respective parameters used.

## 6 Future Work

There is still a lot of research to be done. Some potential research paths stemming from this paper are the following:

- **Tactile Error Metrics** While the Hausdorff Distance is a great general purpose error metric for mesh comparison, it is unknown how this metric correlates with how accurate an object feels to touch. Further research could experiment on how much Hausdorff error (or perhaps other error metrics) is tolerated by human operators before they notice that what they are touching is not the object itself, but a reconstruction.
- **Further testing with organic shapes and real point clouds** All testing on this paper was performed on a small and artificial set of objects. While most real-world man-made objects with which we interact daily have similar shapes to what was tested. It would still be relevant to test the procedure with more organic shapes and using real point clouds instead of sampled point clouds.
- **Preservation of sharp features** The accuracy of this method on smooth surfaces is very promising. But sharper features are not preserved that well. This can be seen in the jagged edges produced from the reconstruction and simplification of the cube, cylinder, and cone.
- **Detection and Simplification of planes** Planes are not simplified completely. This can be seen on the cube requiring 5 times more faces than on its original mesh. Future algorithms could have a special case for surfaces that are planes and simplify them accordingly. Flat

planes are quite common in the real world so this could result in significant improvements.

- **Environment motion to simulation motion** Since even the best mesh reconstruction techniques are still expensive to compute (with run times in the seconds instead of milliseconds), it is not possible to execute mesh reconstructions on every frame of the simulation. There has to be a way for the system to generate a mesh once per object. And then map the motions of the real object to the mesh inside the simulation instead of having to reconstruct it again.

## 7 Conclusions

Tactile Internet is still in its infancy and the research to achieve the ambitious goal of Ultra-Low Latency (ULL) is just getting started. We believe that the best approach for achieving this is through a simulated version of the remote physical environment. This research paper should be a stepping stone in this direction, providing a potential technique for mesh generation from objects in the physical environment.

Various mesh reconstruction methods were analyzed. Poisson reconstruction techniques are not suited for partial point cloud data. Marching Cubes surface reconstruction can accurately reconstruct point clouds while preserving significant detail tolerating up to $20\,\mathrm{mm}$ of Gaussian noise. Improvements can still be made to the preservation of sharp edges and features.

Reconstruction techniques produce very dense and redundant meshes. Mesh Simplification manages to cut the number of reconstructed faces by up to 97% without significant increases in error. This makes the resulting meshes almost 2 orders of magnitude more bandwidth and compute efficient, ideal for Tactile Internet utilization.

Combining Marching Cubes with Hoppe's Mesh Simplification produces accurate and efficient low polygon meshes that can be readily used on future physical simulations for Tactile Internet (TI).

## References

[1] Cluster recognition and 6dof pose estimation using vfh descriptors. https://pcl.readthedocs.io/projects/tutorials/en/latest/vfh_recognition.html#vfh-recognition. Accessed: 2022-06-16.

[2] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17:167–174, 6 1998.

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[4] Phillippe H. Eberhard and Ronald R. Ross. Quantum field theory cannot provide faster-than-light communication. *Foundations of Physics Letters*, 2:127–149, 3 1989.

[5] Vineet Gokhale, Mohamad Eid, Kees Kroep, Venkatesha Prasad, and Vijay Rao. Toward enabling high-five over wifi: A tactile internet paradigm. *IEEE Communications Magazine*, 59:90–96, 12 2021.

[6] Hugues Hoppe. Progressive meshes. pages 99–108. ACM Press, 1996.

[7] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. pages 71–78. ACM Press, 1992.

[8] Zhangjin Huang, Yuxin Wen, Zihao Wang, Jinjuan Ren, and Kui Jia. Surface reconstruction from point clouds: A survey and a benchmark. *arXiv preprint arXiv:2205.02413*, 2022.

[9] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

[10] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21:163–169, 8 1987.

[11] Stan Melax. A simple, fast, and effective polygon reduction algorithm. *Game Developer*, 11(Nov):44–49, 1998.

[12] Renben Shu, Chen Zhou, and Mohan S Kankanhalli. Adaptive marching cubes. *The Visual Computer*, 11(4):202–217, 1995.

[13] Michal Tölgyessy, Martin Dekan, Ľuboš Chovanec, and Peter Hubinský. Evaluation of the azure kinect and its comparison to kinect v1 and kinect v2. *Sensors*, 21:413, 1 2021.