3D Point Cloud Completion from 2.5D Data

Teun Buijs
Supervisor(s): Kees Kroep, Dr. RangaRao Venkatesha Prasad
EEMCS, Delft University of Technology, The Netherlands
22-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachlor of Computer Science and Engineering

Abstract— **The tactile internet can be described as the next step of the internet. It can empower people, adding physical interactions to what normally is just an audiovisual experience. It however does need ultra-low latency. A solution to having delays can be found in environment simulation. This paper describes a way to find the correct shape of objects within the observed environment. There already are AI-based approaches, but the problem is that they are not transparent in how they work. Hence the research question in this paper: Is it possible to create a 3D point cloud completion algorithm from 2.5D data, without resorting to the use of AI? A three-step approach was chosen: dividing the scene, reconstructing its objects, and turning everything into meshes. The focus was on the reconstruction. The basic principle used is that of symmetry. The main conclusions are that this approach can work, but only for relatively simple objects. And whereas the results are easier to understand, the logic-based approach is slower and less accurate than existing approaches.**

1 Introduction

Transport costs both time and money, scaling with distance. Bringing a surgeon to the other side of the world for an operation or sending an astronaut to a satellite for repairs is very expensive. Therefore alternatives that allow circumvention of transport are being researched. Such as ways to empower people by adding physical interactions to what would normally allow only audiovisual experience.

One of the alternatives is called remote piloting or remote operation. This is the act of having a pilot perform tasks through a robotic replacement that has been positioned at the site of operation. To achieve a necessary level of precision the robot has to give haptic feedback on the environment to the pilot. To facilitate the necessary response speed the concept of tactile internet is introduced.

Tactile internet can be described as the next step of the internet [8]. It describes an internet that has to have ultra-low latency (1-10 ms) with close to no errors. This is important due to the effects of a delay in remote work [2]. Using the fastest information transferring medium, light, response times can be measured in milliseconds. Even this is not enough. Going by the speeds as posted by NASA, the ping for halfway across the earth would take around 0.13 s or 130 ms [1]. Based on manual testing, it has been concluded that even a response time of 5 ms can lead to potentially disastrous results.

A solution to having such delays can be found in the world of environment simulation. By simulating the robot's environment, physics calculations can be made locally. This allows for the data transfer delay problem to be exchanged with a simulation accuracy problem. To obtain a simulation that mimics the environment of the robot, the robot needs to detect its surrounding features.

Figure 1: An example of a use case of tactile internet. A robotic hand is mimicking the movements of the operator's hand, giving sense feedback of the object it is holding. [5]

Accurately perceiving the environment comes with its own pitfalls. Getting the correct shape, mass, and texture of each object in the observed scene has complications with measuring these properties in unknown scenarios. Finding the correct mass and texture is out of scope for this research, and will not be discussed further.

Finding the correct shape of objects within the observed environment mostly has trouble with occlusion, parts that are blocked from sight. The Kinect can take images with an added depth element, allowing for the formation of point clouds. Point clouds are sets of 3D points, that can show the shape of objects. A Kinect cannot see the backside of an object, nor can it see whatever is blocked from sight by the object. This means that the pictures it can take are not truly 3D, but rather 2.5D.

An accurate simulation needs fully 3D objects for any realistic interaction to take place. Point clouds are also hard to use within simulations. That is why the research shall handle turning 2.5D point clouds into 3D meshes.

This work is an attempt at answering the following questions:

1. How to reconstruct obscured sections of objects in 2.5D data without the use of machine learning?
2. How to reconstruct occluded sections of objects in 2.5D data without the use of machine learning?
3. How to lessen the effect of noise when reconstructing a point cloud?

This paper proposes a method of reconstructing 3D point clouds from 2.5D data via the use of symmetry and mirroring. This technique works primarily

for simple objects, and will likely not have accurate results for asymmetrical objects or objects that have certain properties entirely hidden from view.

A quick overview of the structure of this paper. Existing works to combat the same problem are found in section 2. Section 3 covers the thought process that was followed during the research process. Section 4 explains how the algorithm works. An overview of the results obtained and the evaluated performance is given in section 5. Discussed in section 6 are the ethics and reproducibility of the study. Section 7 contains branches of possible research that may be done. Section 8 contains the conclusion of this paper.

2 Related Works

There are little to no papers to be found linking tactile internet and point cloud reconstruction. The following subsections will present these subjects concerning teleoperations separately.

2.1 Tactile Internet

Tactile internet can assist with the evolution of teleoperation, though there are still some challenges to be completed before true teleoperation can be a reality [6]. One of these challenges is the impossibility of tactile internet's promise of $1\,\mathrm{ms}$ round-trip delay when considering large distances. Currently, there is very little research that can be found that tries to solve this. This paper is an attempt to rectify that using a simulation-based approach.

2.2 Point Cloud Reconstruction

For a simulation-based teleoperation approach to work, accurate simulations based on real-life data need to be created. A lot of research has been done on the subjects that make up this problem, though they seem to be split into two different types. The first type of research into reconstructing point clouds is based on machine learning approaches, while the second type seems to be more interested in patching up holes in a point cloud.

The amount of advancements made in AI-based development is not a small figure. Its contribution to the point cloud reconstruction scene can be seen. Examples of this can be found in 3D-RecGAN++'s development by Yang et al., the work by Xu et al., and 3D-R2N2 by Choy et al. [12][11][3]. The problem with AI-based approaches is that they are not transparent in how they work. It is practically impossible to understand how an AI has generated its solution, which complicates improving and debugging it.

While not as popular as machine learning, logical approaches can be found that pursue the solution to a similar problem, patching up holes in point clouds. For this paper, the most notable work is that of Cui, Zhang, and Wang who created a hole repairing algorithm based on the notion of symmetry [4]. It is notable because the algorithm described here also makes use of symmetry.

3 Methodology

The initial goal for this work was acquiring a 3D mesh from images taken by a depth camera without resorting to machine learning. This could then be used to allow for the sense of touch to be simulated real time, instead of relying on a feedback channel from a robotic apparatus.

To accomplish this task the research would be divided into multiple steps. It was decided that a three-way divide would be the best way to move forward. This resulted in the steps of; dividing the scene, reconstructing its objects, and turning everything into meshes.

Of these three priority would be given to the second step, reconstruction, due to it being the most vital part of the resulting algorithm. Due to its complicated nature, there was little time for work on the other steps.

An example of a separation algorithm was provided by the Point Cloud Library (PCL) and was supposed to be a placeholder until the time came to find a more suitable version [10]. The chronological limitations of this project ensured that this time would not come to pass.

To create a robust point cloud 3D reconstruction algorithm, the concept of symmetry in objects was utilized. The reason symmetry was chosen was that it was believed that humans unconsciously applied a similar process to identify the unseen side of unknown objects. Symmetry is a common occurrence in nature and the man-made, which might help explain the human identification process [9]. Mirroring objects through their symmetries is nonetheless a complicated task in itself. That is why this step was also divided into multiple sub-steps.

To find a symmetry plane in an object, the corners of an object had to be found first. By taking a point in the middle of these corners, and adding a vector in the direction of one of the corners you get a possible symmetry plane.

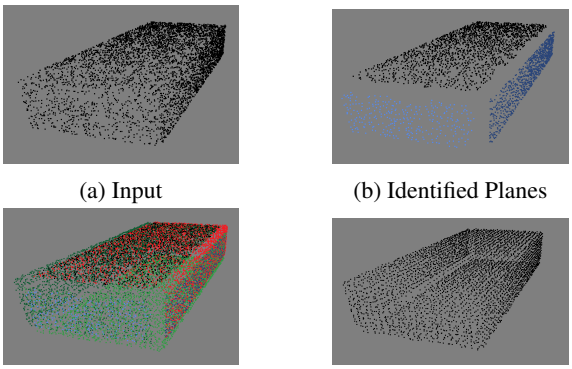
Finding the corners of an object had some complications. The techniques found in previous works did not perform well under the constraints of this project. Therefore a new technique was devised to accomplish this task.

This new technique operated by removing the edges of an object, including the corners, until only the planes remained. After separating the planes, which was done using another of PCL's methods, the corners would be found. Many corner-finding algorithms were thought up, only to be discarded when they didn't work, until a solution was found that

worked well most of the time. This solution was used for a few weeks until it was deemed unsatisfactory. The final solution didn't find corners in the point cloud, instead it made them.

After using the corners to create potential mirrors of the object and by comparing the mirrored points to the original points of the object, the accuracy of the symmetry plane could be calculated. Only the symmetry planes with high accuracy should be used, to not generate features where there should be none.

The complete point cloud could now be turned into a mesh. Again, because of the limitations put on this project, the viable solution would be to form a naive mesh.



(a) Input (b) Identified Planes

(c) Mirrors (d) Final Product

Figure 2: A simple rectangular object is simple to reconstruct when 3 of its sides are observed.

4 Implementation

To help with the clarity of processing, the algorithm was developed in two different parts. Each part will be explained in the subsections below.

4.1 Step 1: Separation

The first step to turning a 2.5D scene into a 3D scene is the separation of objects. It helps with focusing on the objects themselves, ensuring that outside interference in the reconstruction step is minimized. By handling each object independently, prioritizing objects that are more likely to be interacted with is also possible. This could allow for a slight performance boost.

Separating the scene into different objects would also help with dealing with occlusion. By ignoring the objects in front of the object, each hole in the object can be handled as missing data. These could then be patched using the same technique that was used for the unseen side of objects.

The separation itself was done using Rusu's work on PCL [10].

4.2 Step 2: Reconstruction

The reconstruction of an object is the most complicated part of the algorithm. The basic principle used here is that of symmetry. By mirroring the object through symmetry planes, an estimation could be made of the unseen parts of the object.

To find the symmetry planes, the defining features of the object first have to be identified. For most objects, the defining feature can be found in the corners of said object. By taking two close corners, a potential symmetry plane can be created. After mirroring the object in this potential symmetry plane, the accuracy could be calculated. Should the accuracy satisfy the requirements set for the objects in the scene, the symmetry plane is considered real and further testing can be done.

As most point clouds are filled with noise, PCL's voxelization algorithm initially processes the point cloud. This reduces the noise and the number of points in the resulting point cloud without losing much detail.

To find the corners of an object, a custom technique was created. First the planes of the object needed to be found. To find the planes of an object, it removes all the edges of the object.

The edges are found by using PCL's normal generation, and finding which points have normal vectors that differ from those around them. The difference was calculated by summing the dot products of the point's normal vector and its nearest k points. A value of 16 for k gave satisfactory results and was thus used. The algorithm marked all points that had an average normal dot product of 0.97 or higher as a non-edge.

After the planes are found, the planes must be separated. This is so that the corner finding algorithm finds the corners of each plane. This separation was done using PCL's RANSAC method, with a distance threshold of 0.015 [10]. Figure 2b is an example of an object that has its edges removed and its planes separated.

The corner generating algorithm generated corners by finding the intersection line of each combination of planes. Each plane then has every point mapped onto this intersection line. The points on the intersection line that are furthest apart are then chosen to be the corners of the plane. This means that per unique combination of planes there are 4 corner points.

Using these corners to generate possible symmetry planes, mirrors are made through them. An example of mirroring can be found in figure 2c, which has three mirrors. The accuracy of these mirrors is calculated using the Hausdorff Distance. This algorithm sums up the distances of all the mirrored points and their nearest original points.

When mirroring there is a possibility for the result to have multiple similar mirrors. This clutters

the eventual object and is undesirable. To combat this, each mirror is compared to a previously generated mirror. If the calculated difference is less than 0.001 it means that the mirrors are too similar, so it doesn't apply the new mirror.

Errors during mirroring can occur, usually this happens when the algorithm tries to mirror though a symmetry plane that is not present in the object. As such, an acceptance threshold is applied that removes all planes that do not have a Hausdorff score of less than this threshold.

To obtain the final product, all the mirrors that comply with the preset standards are added to the input point cloud. This point cloud is then voxelized with a voxel size that is double the size of the initial voxelization.

5 Results

The examples used in this paper were initially created as objects in the open-source 3D modelling software Blender then processed into point clouds in the open-source point cloud processing software CloudCompare.

There are three parameters that can be changed to influence the outcome of the algorithm. These are voxel size, noise, and mirroring acceptance score. To obtain the results from figure 2 the voxel size was 0.1 and the mirroring acceptance score was 2.0.

To apply parameter changes, a custom point cloud visualizer was developed that would allow for parameters to be changed on the fly. The only influence this had on the eventual algorithm is the assistance with which it was created. The figures of point clouds in this paper are taken from this visualizer.

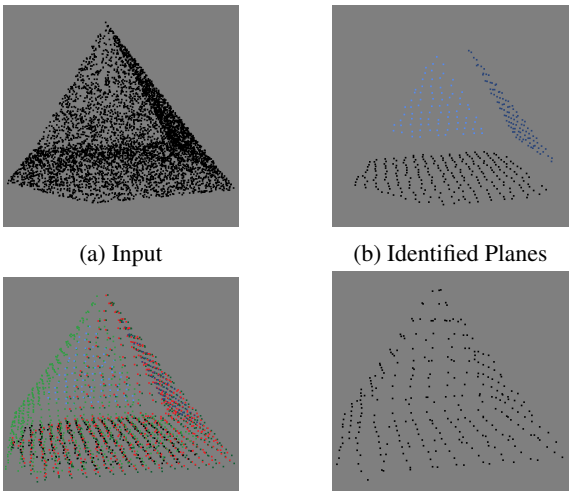
5.1 Shapes

To display the results of this algorithm on different simple objects, the figures 2, 3, 4, and 5 give an overview of what can be expected of the different stages of the algorithm.

5.2 Noise

Noise is an ever occurring problem when images are taken. This is no different for depth cameras. To test the effect of noise on the algorithm, a noise generator was implemented. This noise generator provides a bit of Gaussian noise to each point in the input point cloud. It takes as an input a value for the spread. The noise value generated is a value that has a $10\,\%$ chance of being equal to or greater that this input noise value.

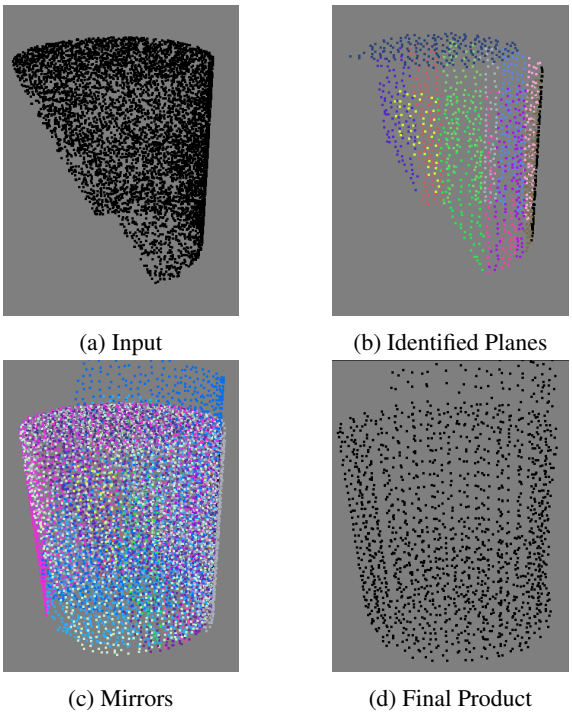
As can be seen in figure 6, noise can have a substantial effect on the algorithms final product. The voxelization of the input point cloud does help negate a portion of the noise, but when the noise value is



(a) Input (b) Identified Planes

(c) Mirrors (d) Final Product

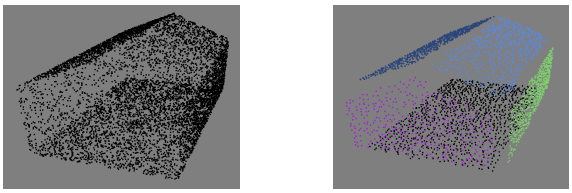
Figure 3: A pyramid is shown here as an example of an object that has triangular planes instead of rectangular planes.



(a) Input (b) Identified Planes

(c) Mirrors (d) Final Product

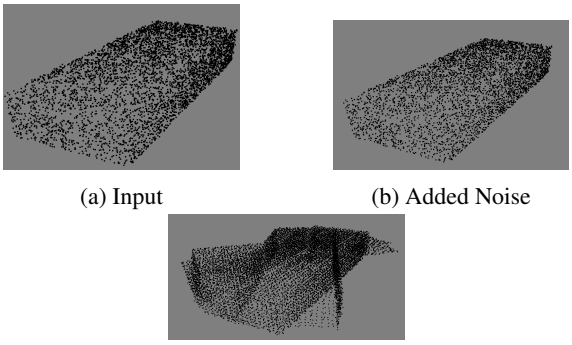
Figure 4: A cylinder has a round body, which means it has lots of potential symmetry planes. While the algorithm makes a few errors, the human eye can recognise the cylindrical shape.



(a) Input (b) Identified Planes

(c) Mirrors (d) Final Product

Figure 5: This simple house shows the working of the algorithm on a more complex object than a rectangle or a pyramid.



(a) Input (b) Added Noise

(c) Final Product

Figure 6: Even a bit of noise can have a vast negative impact on the algorithm's performance.

greater than or equal to the voxel size the algorithm's performance takes a serious hit.

5.3 Discussion

The results of this research show that the more complex an object gets, the more planes an object gets, and the more difficult it is to give an accurate reconstruction of such an object. When the amount of planes of an object is around 10 the time it takes to compute the final product is much higher than acceptable.

Compared to other works that make use of AI, this work's results might seem unsatisfactory. Machine learning allows for more complex objects to be reconstructed in less time. The importance of this work lies not with the limitations of its work, but rather with what it represents. By not using machine learning to compute the unobserved side of objects, this method shows that even without machine learning it is possible to reconstruct a 3D point cloud from a 2.5D object.

The algorithm is simple to understand, and alterations may allow for a better performance. With time it may be able to outperform AI-based approaches. This time is far off though, as the quality of these other approaches is still improving as well.

6 Responsible Research

As no direct collaboration has taken place during the process of this research, no data has been obtained from sources that are not listed in this paper. All responsibility for the integrity of this paper falls to the author.

The results of this paper are rather basic, as the amount of nuance that is necessary for 3D object reconstruction is difficult for an algorithm to keep track of. The shapes that were chosen to represent the results of this work may not give a full representation of what might be encountered in the application of the algorithm.

While this tool may be used for less ethical ends, military drones, for example, it is not unique in its work. As has been shown in section 2, there are already machine learning algorithms that can produce similar results.

The methodology explains the thought process behind the design decisions of the algorithm, while the implementation describes how the algorithm processes a 2.5D point cloud into a 3D point cloud. The results explain how the examples in the figures were obtained. These sections should hold enough information to allow for the reproduction of this research.

7 Future Work

While the accomplishments of the algorithm work well for most simple shapes, some mirrors can cover

parts of the object that are visible to the camera. This can have adverse effects on the algorithm's eventual use, for that is not what it is supposed to do. A filter should be designed that either removes the mirrors or the points of the mirrors that block the view of the input object.

The algorithm currently has little that allows for the generation of an object's lower plane. Depth cameras cannot see the bottom of an object, which means it's paramount for an algorithm to be able to reconstruct this. The current untested idea is to map all of the object's points onto the plane that the object rests on, though this will not work if there is little surface area to work with.

There have been troubles with getting real-life examples to work with the plane detection algorithm. An attempt was made at creating an algorithm that would help combat this, but it didn't succeed. While it is not certain, it is suspected that the problem either lies with PCL's normal estimation or with the plane detection algorithm itself.

There has also been research performed on updating point clouds with new information [7]. This can be useful for the algorithm for it would help to remove mistakes.

8 Conclusion

This work was primarily meant as an experiment into the possibility of 3D object recreation without the use of an AI-based approach. An algorithm was developed using the principles of symmetry. This algorithm has achieved the goal of reconstructing simple shapes, including compensation for partially obstructed objects.

However, more complicated shapes are made up of more difficult geometry which makes them more difficult to calculate. The noise that a depth camera generates can also have a large negative impact on the performance of the proposed algorithm.

Several improvements have been proposed and can be implemented to obtain further insights into the working of such algorithm-based approaches. Still, the current progress of AI-based techniques vastly outperforms the performance of the algorithm-based approach.

References

[1] How "fast" is the speed of light. `https://www.grc.nasa.gov/www/k-12/Numbers/Math/Mathematical_Thinking/how_fast_is_the_speed.htm`. Accessed: 2022-05-23.

[2] Alejandro Alvarez-Aguirre, Henk Nijmeijer, Toshiki Oguchi, and Kotaro Kojima. Remote control of a mobile robot subject to a communication delay. In *ICINCO*, 2010.

[3] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction, 2016.

[4] Linyan Cui, Guolong Zhang, and Jinshen Wang. Hole repairing algorithm for 3d point cloud model of symmetrical objects grasped by the manipulator. *Sensors*, 21, 2021.

[5] Vineet Gokhale, Kees Kroep, Vijay S. Rao, Joseph Verburg, and Ramesh Yechangunja. Tixt: An extensible testbed for tactile internet communication. *IEEE Internet of Things Magazine*, 3(1):32–37, 2020.

[6] Vineet Gokhale, Kees Kroep, Vijay S. Rao, Joseph Verburg, and Ramesh Yechangunja. Tixt: An extensible testbed for tactile internet communication. *IEEE Internet of Things Magazine*, 3, 2020.

[7] Olaf Kähler, Victor A. Prisacariu, and David W. Murray. *Real-Time Large-Scale Dense 3D Reconstruction with Loop Closure*, volume 9912 LNCS. 2016.

[8] Nattakorn Promwongsa, Amin Ebrahimzadeh, Diala Naboulsi, Somayeh Kianpisheh, Fatna Belqasmi, Roch Glitho, Noel Crespi, and Omar Alfandi. A comprehensive survey of the tactile internet: State-of-the-art and research directions. *IEEE Communications Surveys Tutorials*, PP, 4 2020.

[9] Joe Rosen. Symmetry at the foundation of science and nature. *Symmetry*, 1, 2009.

[10] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. IEEE.

[11] Jiamin Xu, Weiwei Xu, Yin Yang, Zhigang Deng, and Hujun Bao. Online global nonârigid registration for 3d object reconstruction using consumerâlevel depth cameras. *Computer Graphics Forum*, 37:1–12, 10 2018.

[12] Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3d object reconstruction from a single depth view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:2820–2834, 12 2019.