



Utrecht University



Faculteit Bètawetenschappen

A simulation environment for robot depth and
color sensors
MASTER THESIS

Tim Hoogenkamp

Department of Information and Computing Sciences

Supervisor:

Prof. dr. R.C. VELTKAMP

December 27, 2019

Abstract

Robots with depth and color sensors, such as the humanoid robot Pepper, can be used for applications when autonomous navigation and object detection are available. However, most of these robot vision sensors lack the quality for such tasks.

For that reason we built a simulation where the generation of the output of these sensors is the main focus. The simulation provides a testing environment and could provide assistance in real world navigation and object detection tasks.

However, The quality of these synthesized views representing the vision sensors of such robots is not good enough to perform these tasks. In this master thesis we look to create our own dataset of a real world environment for the simulation and we improve the results of these synthesized views with our own captured depth and color data.

In this work we have two objectives. The first objective is to create a dataset representing a real world environment that can be used in the simulation. We show how to best obtain the data. We generate the 3D reconstruction of the environment and then combine the 3D reconstruction with our own captured depth data. The second objective is to build a fusion algorithm that improves the quality of the depth data generated by the 3D reconstruction, by combining it with our own captured data. The ultimate goal is to enhance the quality of the synthesized images representing the color and depth camera of a robot.

We show how we create two datasets that can be used in the simulation environment and design a depth fusion algorithm that improves the quality of the synthesized views. We show that our fusion algorithm works in most cases, but that for large distances the algorithm needs some additional tuning or additional steps during capturing the data should be taken. We discuss how to further enhance the results of the fusion algorithm as well as how to improve the simulation and how it could be used for testing purposes.

Contents

1	Introduction	1
2	Context	3
2.1	Simulation	3
2.2	Functionalities	4
3	Related work	8
3.1	RGB-D dataset	8
3.2	3D reconstruction	10
3.3	Image quality	11
3.4	Depth fusion	13
4	Pilot	15
4.1	Depth camera	15
4.2	Data collection	16
4.3	Results	19
5	Objectives	21
6	Approach	23
6.1	Gathering data	23
6.2	Depth fusion algorithm	24
6.3	Depth fusion evaluation	28
7	Experiments	31
7.1	Dataset	31
7.2	Depth evaluation	33
8	Results	34
8.1	Statistical analysis	35
8.2	Per-trajectory analysis	42
9	Conclusion & future work	46
9.1	Conclusion	46
9.2	Future work	47
A	Colmap parameters	49
	References	IV

1 Introduction

Humanoid robots Humanoid robots are becoming more popular in many different research areas. The robots having two arms, two legs and a head have a physicality similar to that of a human. Some models only replicate the human body from the waist up, but the effect is nearly the same. Often equipped with speech recognition and the ability to mimic human gestures they are used for a variety of tasks. They are among other things used for educational purposes, greeting guests and visitors, entertainment and research. The reason for deploying a humanoid robot can vary per task. In most cases it is exciting and new to interact with a robot and the human physicality of the humanoid robot helps put people more at ease for the communication. Other advantages include doing tasks that are dangerous or very simple for humans to do, making a humanoid robot a suitable substitute.

Welcoming customers A specific example of a task for a humanoid robot is to receive customers at a banking company's building. A humanoid robot would be able to welcome customers and process their requests. The robot then can either tell the customers where in the building they need to be or guide them towards their destination. However, the communication of humanoid robots is mostly limited to preprogrammed input and responses. This makes the flexibility of a robot as a host somewhat limited. On top of that the ability to guide a customer to her destination is also limited as accurate autonomous navigation for most robots is still lacking.

Humanoid robot at ING The banking company ING has a humanoid robot: Softbank's Pepper. At ING they would like to use Pepper as a host in their banking buildings, however the limitations mean that this is not possible yet. One of the problems we want to focus on is the inability of Pepper to properly navigate in an environment. Robert Groot [Gro18] explored the possibilities for Pepper to be able to autonomously explore an environment. He used Simultaneous Localization And Mapping (SLAM) and navigation to achieve this. However, due to the lack of quality of the depth and color images produced by the cameras of Pepper, the reconstructed map was not accurate. This resulted in a bad localization for Pepper which results in the inability for Pepper to autonomously navigate the environment.

Solution and goal Since there are limitations of what Pepper is able to do we want to construct a virtual environment where we can test navigation and other algorithms. Specifically we want to create a simulation with a reconstructed real world environment where we can safely and easily test tasks like navigation. Since the robot would always operate in the same room or rooms, reconstructing a real environment for the simulation has an advantage as the simulation would be an accurate representation of the static environment a robot will be in. For many applications the static environment will not change, so a simulation provides us with a realistic testing ground for algorithms for the robot. The reconstructed environment can also provide an external map that robots can use as assistance in navigation tasks.

Possibilities Creating a simulation for robots also opens the possibility to test object detection algorithms. In a virtual environment we can synthesize a larger number of images of what the camera of a robot would capture, in less time. These images can be used for deep learning algorithms that would train robots to recognize

objects, so that at some point interaction with these objects becomes a possibility. This also allows testing algorithms which use a combination of navigation and object detection. The combination provides algorithms for behaviour where the robot can move around objects to get a better grasp of what the object really is. Another example is for the robot to find certain objects to move towards and interact if the robot has these capabilities. Other advantages of a simulation besides generating data for deep learning algorithms include testing in a safe environment and reduction of the required manpower for testing certain algorithms. For example creating a situation in a crowded environment can lead to unsafe situations when testing robot algorithms, and would require a great number of people to run such tests.

Reconstruction Before we can test the algorithms we first need to build the reconstructed environment in order to create a simulation for a humanoid robot. This can be accomplished with Image Based Reconstruction (IBR). With this algorithm we can generate a scene for the simulation and provide synthesized views which will represent the cameras of the robot. For the reconstruction we will be using, testing and adjusting a 3D reconstruction pipeline currently under development at Utrecht University. This work based on work by Hedman et al. [Hed+16] will provide us with the necessary scene and synthesized views. In order to use the pipeline we want to build a RGB-D dataset that would function as our own input.

Project goal In short, the goal of this project is: Create a simulation environment for robots by reconstructing a real environment and use it as a safe and easy testing ground for object detection and navigation algorithms.

2 Context

In this section we will describe the context of the project. We will be looking at the simulation which is under construction and at what we want the simulation to be able to do. The main goal of the simulation is to create novel synthesized depth and color images. These images represent the output of the cameras of the robot and this data can be used for testing navigation and object detection algorithms. We achieve synthesizing views with Depth Image Based Reconstruction (DIBR). The input for the DIBR algorithm is a set of captured color images in a real world environment, paired with depth maps for each color image. With the color and 3D data we are able to synthesize the views of the robot that simulates what the cameras would capture if the robot would be present in the real environment. In subsection 2.1 we will look at the data needed for the DIBR algorithm and at what needs to be done to generate our own data to get a simulation of a real world environment of our choosing. In subsection 2.2 we will dive into more functionalities around the DIBR algorithm to give control and visualizations to the user, and into desired functionalities for testing algorithms.

2.1 Simulation

Data For the simulation we first need the color and depth data that can be used as input. The DIBR algorithm is being tested with the dataset created by Hedman et al. [Hed+16]. Hedman also used the dataset to generate synthesized views. The dataset consists of captured low spatial resolution (640 x 480) RGB and depth images taken from a RGB-D video of the environment as well as very high spatial resolution (4912 x 3264) color images which have been pre-processed in Adobe Lightroom. The dataset contains the high resolution per-view depth maps, generated to match the high resolution color images. They also provide the camera pose estimations calculated and used in the reconstruction process. The high resolution depth and RGB data can be used in DIBR. However, there still are some limitations regarding the results of the synthesized images using this dataset. The first problem is the coverage of the images, which is too low to allow free viewpoint DIBR. We want the robot to be able to move and look around freely and therefore need as much coverage as possible. Some of the areas in the dataset are not covered by the color images and consequently the depth maps as well, resulting in synthesized images with large holes. Another issue is the accuracy of the generated depth maps, which does not result in the desired quality for the synthesized views. The reason is the quality of the depth maps, which is too low to provide accurate reconstruction results. The low spatial resolution depth maps are not utilized, as they cannot help us much with refining this data. The main problem with the low spatial resolution depth data is that it does not cover the same areas as the high resolution color images. Although it would be possible to combine the recorded low spatial resolution depth maps to enhance the results, we did not focus on this part as we want to build our own dataset. However, we could utilize captured depth data easier, if the depth maps correspond to the color images used for the reconstruction.

Preprocessing In order to get better results for our synthesized images we generate

our own depth maps for the high spatial resolution color images. We calculate the new depth maps and new camera poses using the Colmap software. Colmap is designed for state-of-the-art 3D reconstruction and uses Structure-from-Motion (SfM) to calculate the estimated camera poses [Colmapsfm]. With this information the software uses Multi View Stereo (MVS) to generate normal and depth data for every pixel and fuses the data in a point cloud [Colmapmvs]. Then it can also generate the global mesh using screened Poisson surface reconstruction [KH13], but we only use the point cloud since the results of the Colmap reconstruction do not reach the desired quality yet as well. After obtaining the depth data, the data is refined to enhance the quality and fill some of the missing information. The resulting depth maps are used together with the initial color images and calculated camera poses for the construction of free viewpoint synthesized views. The full overview of the process is depicted in figure 1.

DIBR In order to generate the synthesized views we use a DIBR algorithm. In DIBR the texture of the color image is projected from reference views to the new view based on the camera poses and depth data. Our DIBR algorithm not only selects input views based on the angle and distance from the desired synthesized view, but also takes the overlap between the images into account. On top of that we use more than two images for the generation of synthesized views. After a selection of input images is made the relevant pixels are selected and projected onto the novel synthesized view with 3D warping. In our DIBR algorithm we generate photo-realistic real-time free-viewpoint images in the scene, given the input quality is high enough and has full coverage. Since the dataset from Hedman cannot provide us with this quality and coverage, we will focus on creating our own dataset in this work that comes closer to realizing that goal.

Capturing data In order to achieve more accurate results for the synthesized images this work will focus on utilizing captured depth data to enhance the quality of the Colmap generated synthesized depth data. To achieve that we want to capture relatively high resolution depth images (preferably 720p or higher) paired with color images to use as input for the DIBR. We first want to observe what happens when we capture our own data and use Colmap to calculate the depth maps. If the quality turns out to be low, we can use the captured depth data to refine the depth data generated by Colmap. In section 4 we will discuss a pilot we did capturing data and its results to show where to go to from there. First we will give an overview of the full simulation environment as well as the desired functionalities.

2.2 Functionalities

Core The simulation environment currently consists of several visual and control options. The main screen is generated in Rviz, a Robot Operating System (ROS) framework that can visualize 3D data and interactions. We load the point cloud of the real world environment generated by Colmap, as well as a representation of Softbank’s Pepper robot as visualizations. We use Pepper to demonstrate the framework, but it would work with an arbitrary robot as long as the robot has cameras. Pepper can be controlled with key presses to move forward, backward and to rotate as Pepper would in the real world. Pepper is loaded with the Universal

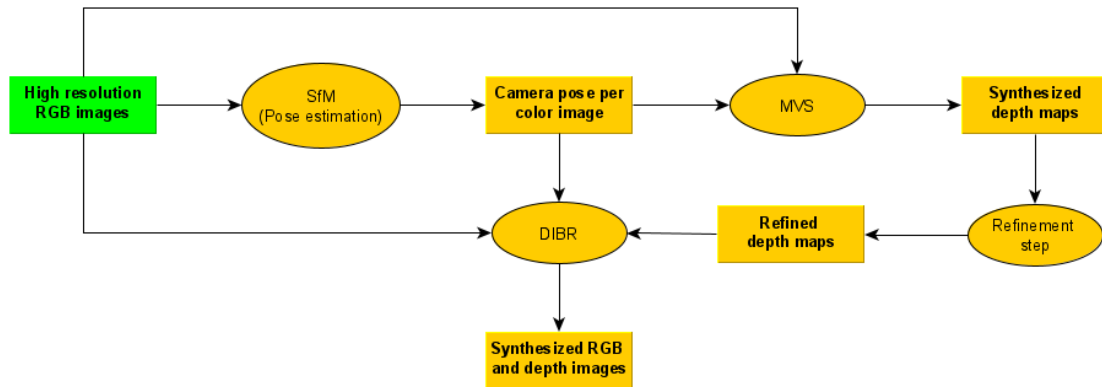


Figure 1: Overview of steps of the 3D reconstruction. The green block represents the input from the dataset. Yellow ovals represent algorithms in the pipeline and yellow blocks show the output of the algorithm, which can be input for a later stage.

Robotic Description Format (URDF) meaning we can control all sensors, joints and links. In addition we also show markers for all camera poses and for the trajectory of Pepper in Rviz. With all the 3D information which is visualized in Rviz we calculate the synthesized views. We use the 6 DOF information from ROS to determine the camera pose for the color and depth sensor of Pepper. Based on this information, two new screens are generated to show the synthesized free viewpoint color and depth image. This represents the images the cameras of Pepper would capture at the location in the virtual 3D world. An example of the visualization can be found in figure 2. There are some additional features we would like to have for this simulation, described in the rest of this subsection.

Quality map One of the features we want to add to the simulated environment is a map that shows the expected quality of a synthesized view. For the quality map we want to show a top-view of the scene with a grid overlay. Each cell contains data in the form of a circle. The circle shows a color representation of the expected quality of the novel view from that pixel with a given orientation for that point. For an example we refer to figure 3. In this example the blue pixel would be the position of the robot. The colored circle shows the estimated quality of the novel view based on the orientation of the robot in that region. For this quality map we would need a way to define the expected quality of novel views.

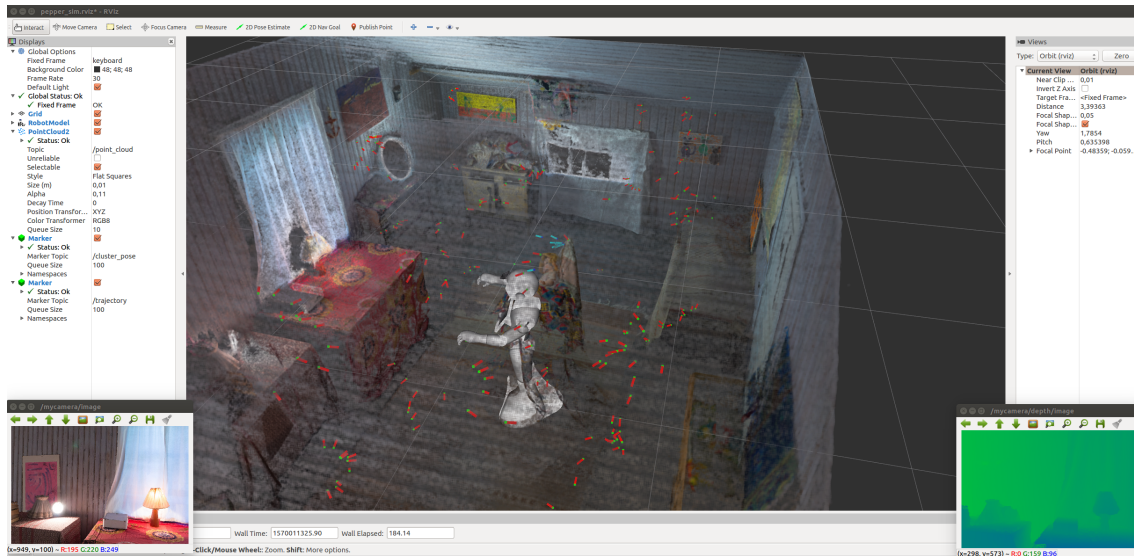


Figure 2: The simulation with the RViz screen in the middle with Pepper, the point cloud representation and the markers. The green points represent the location the image was taken and the red cylinders represent the direction in which the photo was taken. All the blue markers represent the images used for the DIBR algorithm. The bottom left is the synthesized color image and the bottom right is the corresponding synthesized depth image colored.

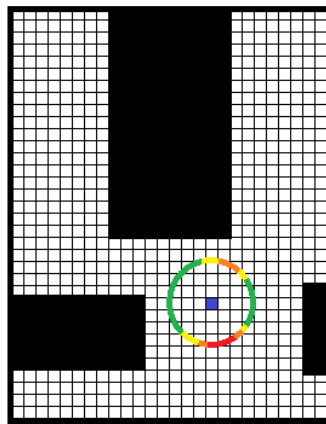


Figure 3: Example of a desired quality map. Solid regions represent objects like shelves and desks. The blue pixel would be the selected pixel for the novel view. The circle around the pixel shows a color coded quality of the novel view given the direction the camera is facing.

Navigation One of the reasons for the simulation is testing autonomous navigation algorithms. With sufficient quality of the synthesized images representing (a possibly better version of) the color and depth sensors of robots we can add autonomous navigation algorithms like visual SLAM to the simulation. In combination with the quality map we can generate routes or points of interest to navigate towards and to predict the performance when executing the navigation algorithm. If the robot needs to navigate in the same room as the simulation we can use the generated sensor

data as well as the prediction map to assist with real world autonomous navigation tasks.

Object detection Another functionality we want in our simulation is the ability for object detection algorithms to be designed and tested. For this we need ground truth by making sure we have 2D and 3D annotations in our scenes that represent the label of the actual object. With these labels we are able to generate training and validation data for object detection or to use it as a testing ground for existing trained object detection algorithms. Then we can create and test algorithms that can combine navigation and object detection for tasks a robot could do. For example if initially the object class is recognized, but the specific object is not. In that case the robot can move toward and around the object to get more information to provide a more specific description of the object.

Real world Finally we want to be able to use the data generated in the simulation to assist in the real world tasks. This is why we want to be able to generate our own data which is good enough to perform the tasks in the simulation. Creating the simulation in ROS as we are doing means we can use the data as most robots use ROS to generate their behaviour. The largest benefit would be to generate higher quality color and depth data that can provide better data for vision based navigation and object detection algorithms. The quality map can help determine if the data at that location of Pepper would actually benefit from the algorithm, or in case there is too much missing or unreliable information, the robot would be better off relying on its own sensors.

3 Related work

In this section we will discuss some of the related work. There are four subsections that discuss four subjects important to this work. We start by discussing RGB-D datasets in 3.1, since we make our own dataset as well. In the next subsection we look at the process of 3D reconstructions as used by Colmap. Although our work does not create our own 3D reconstruction algorithm we do use the RGB-D data as input for the algorithm, so we explore the options in subsection 3.2 and the choice of the Colmap software. In this work we are trying to increase the quality of synthesized views by capturing our own dataset to increase coverage, but also by capturing additional depth data. To be able to evaluate the generated synthesized images we also look at how to access image quality for DIBR in general and specifically for our DIBR algorithm in subsection 3.3. The last subject will be depth fusion discussed in subsection 3.4, as we fuse the depth maps captured by the camera with the Colmap generated depth maps.

3.1 RGB-D dataset

Depth added to datasets The most important component of the simulation is synthesizing the views of the cameras of the robot. Creating synthesized images with IBR starts with having a RGB-D dataset that can be used as input. Since RGB datasets can be used for all kind of scene understanding tasks, it became a popular research topic. The available commodity depth sensors created interests in adding depth data to RGB data to open more possibilities. One of the earlier works in RGB-D datasets is the hierarchical multi-view object detection dataset [Lai+11]. Their dataset contained images of 300 objects that were a combination of a depth and RGB image. With their dataset they showed the value of the additional depth data by creating a 3D reconstruction of a scene with several of the objects in them. Their goal was to create an object detection dataset for robots, so the robots can find and pick up the items that were in their dataset. Another dataset created around the same time is the NYU [SF11] RGB-D dataset. Their focus was creating a dataset that can be used for more general scene understanding tasks. To achieve this the dataset contains RGB and depth images taken of indoor scenes. The dataset contains 2D per-pixel annotations, as annotations are essential for training object detection and other scene understanding algorithms. Although the dataset was considered state of the art at the time, there are some limitations. The lack of actual 3D information, caused by a limited coverage and lack of 3D annotations make this dataset less suitable for 3D reconstructions. The number of scenes is also limited, which is why more researchers worked on creating larger RGB-D datasets, with better and more annotations.

Annotations & 3D reconstruction The main challenges for generating a RGB-D dataset are annotation of objects or classes and getting enough data. Especially annotating objects is a labor intensive job that still needs to be done by hand. In the SUN3D [XOT13] dataset they tried to automate some of this process. Their goal was creating 3D reconstructions of the scenes guided by some initial annotations. After the 3D reconstruction some remaining annotations are guided by the 3D scene

and initial annotations. However, this still relied on human work and as a result only a small portion of the dataset has camera poses, semantic labelling and a 3D reconstruction. In the NYUv2 [Sil+12] dataset they were pretty successful in automating segmentation and categorizing the segments to different classes. Unfortunately the dataset lacked coverage for 3D reconstruction tasks as the dataset was not created with this purpose in mind. The SUN RGB-D [SLX15] dataset was created as follow up, as the annotations in the NYUv2 set were done only on the 2D data and the dataset did not contain as much annotations as desired. They bench-marked the sets by testing the data with some scene understanding tasks. Their goal was to combine some of the existing data as well as to create their own in order to have a dataset that was comparable to the size of existing RGB training datasets. One of the problems was that this dataset still was not very suitable for 3D reconstruction. A dataset more focused on 3D reconstruction and providing annotations in 3D is SceneNN [Hua+16]. The dataset was the first to create triangle meshes for the whole scene and they annotated their data per-pixel and per-vertex for most of the reconstructed scenes in their dataset. Their work allowed them to generate synthesized views through IBR and work on shape completion and intrinsic decomposition algorithms. For their 3D reconstruction they used the method introduced by Choi et al. [CZK15]: at that time a state of the art 3D scene reconstruction algorithm. One of the latest works is the ScanNet [Dai+17] dataset. They focused on creating a way for collecting the data with a larger group of people instead of just experts that had to annotate, but most importantly had the large group collect the RGB-D data. This means this work has one of the largest annotated datasets for RGB-D scene understanding tasks today. There is a lot of work on collecting and generating the RGB-D data for algorithms in the community. The key difference can be observed in the way they create the annotations, the amount of annotations and the way the depth data is acquired. Specifically depth data can be different, as some prefer synthesizing the depth data, while others use captured depth data from depth cameras.

Matterport All of the previously mentioned works collected data using commodity RGB-D sensors. There are other options like the Matterport camera. A camera that generates panoramas from a single point by rotation and capturing a lot of RGB and depth images and combining them. Entire buildings can be captured by placing the camera two to three meters apart and take panoramas at each point. The software that comes with the camera can combine all input data and is able to do the reconstruction and pose estimation. It allowed researchers to generate 3D reconstructions of entire buildings and made capturing the data a lot easier [Cha+17; Arm+17]. A large drawback of the device is that it is extremely expensive compared to the commodity sensors and thus has not been used a lot for research in 3D reconstruction.

Conclusion A lot of RGB-D datasets have been created over the past decades and much research has been conducted for reconstructing environments using RGB and possibly depth data. The use of depth data varies from method to method and improvements in terms of accuracy can still be made within depth acquisition as well as within reconstruction tasks. When we are testing object detection algorithms it will also be important to annotate the data acquired, which requires a lot of time

and is not something we do in this work.

3.2 3D reconstruction

IBR The RGB-D datasets can be used as input for DIBR algorithms. Although the quality of the results may vary, the basis is generating synthesized views from images using RGB images and when using either synthesized or captured depth data it becomes DIBR. IBR can be categorized in three different techniques: rendering without geometry, rendering with implicit geometry and rendering with explicit geometry. However, it should be considered more of a continuum, rather than a classification [SK00]. Explicit rendering is dependent on accurate textured geometric representations and uses less images. On the other end of the continuum one finds the IBR methods that just use a lot of images and interpolate the images to generate the synthesized views. These kind of IBR algorithms are based on the plenoptic function, which calculates the intensity of light rays going to the camera at every position and orientation. In between are the methods that apply both approaches. Methods that fall in between would be methods that for example use feature point matching between the images and calculate the optical flow. The introduction of capturing depth data caused a shift in research towards more explicit methods.

ICP With the availability of cheap depth sensors a lot of the 3D reconstruction techniques focus on utilizing the easy way of collecting depth data with the color data and use that data to generate the explicit geometry. Iterative closest point (ICP) [CM92] is one of the methods that utilizes the depth data to generate the reconstruction. The method estimates the relative motion between two frames and together with loop closure detection and correction it can generate large scene representations. Henry et al.[Hen+12] used ICP in combination with RANSAC[FB81] to build a dense 3D map of an indoor environment. Their modification uses FAST feature matching and RANSAC to generate an initial guess for the model, to then use ICP to determine the best alignment between the frames. The KinectFusion[New+11] reconstruction was created specifically for the depth data collected by the Microsoft Kinect. They fused the depth data into a single surface model generated real-time. The model is represented as a volumetric truncated signed function (TSDF). Their representation helps merging the next depth frame into their model, resulting in a smoother model that can be updated real time.

MVS Multi-view stereo (MVS) is a different 3D reconstruction method that takes a collection of images with known camera poses. This method is used when no depth data is available or the quality of the depth data is too poor for reconstruction purposes. MVS takes the RGB images and camera geometry data to generate 3D data in various forms. Most common ways to represent the 3D data are voxels, polygon meshes, level-sets or depth maps [Sei+06]. The idea is to match a pixel in one image to the corresponding pixel in the other images that represent the same point in the 3D space. There are two important steps for finding the corresponding pixels in the other images. The first one is to find possible candidates in all other images and after that the algorithm needs a way to measure how likely it is that the candidate is a match [F+15]. For the first part of the problem it is important to have the camera poses so the candidates can be found using the geometry of the

cameras.

Camera poses In the work of Seith et al. [Sei+06] they assume the camera geometry is known. However, in most cases the camera parameters are not known and need to be calculated. To achieve this many reconstruction algorithms first apply a Structure from Motion (SfM)[MMM12] algorithm, which calculates the camera parameters as well as 3D tracks. The method starts by calculating features in every image using a method like SIFT or FAST. After that the features are matched to generate the 2D tracks between the images. Those tracks are then used to calculate the camera poses. Although SfM is a robust algorithm, when providing the data it is important to realize that feature detection is the main idea for the algorithm, so the data needs to contain enough features. Most SfM algorithms use Bundle Adjustment (BA) to minimize the reprojection errors with a non-linear least squares error function as an optimization step.

Photo consistency measure The second step in finding corresponding pixels determines which of the matches is most likely to be the correct match. For the matching of the possible candidates MVS applies a photo-consistency measure. The method measures the agreement between the potential matches in the images based on illumination, materials and 3D geometry that was captured. The photo-consistency between pixels or regions of pixels is maximized to get the best matches in order to generate the 3D reconstruction. One of the latest works on MVS from Galliani et al. [GLS15] uses the Patchmatch idea where they iteratively optimized planes in scene space to obtain a depth map and normal field per view, while maximizing the photo consistency measure. This work focused more on the speed of the process by doing a global matching instead of per-pixel matching and utilizes the now available GPUs computing power for their algorithm. Overall the MVS method can be viewed as a constrained optimization problem, where multi-view photo-consistency is maximized as a function of geometry, viewpoints, materials and illumination.

Conclusion Since we do not have the camera poses in advance we need a SfM technique to determine them. We choose to use the MVS method to generate our 3D data as we want to generate the 3D maps instead of a mere representation of the scene through ICP. Therefore we use Colmap as the state of the art combination of MVS and SfM to determine the necessary camera poses, depth maps and point cloud.

3.3 Image quality

Image quality assessment With the results of Colmap new images can be synthesized by combining the 3D and color data. To confirm the quality of the resulting synthesized view an objective quality assessment is necessary. Measuring the quality of images is also known as Image Quality Assessment (IQA). Most objective IQA algorithms look at what humans consider good or bad quality and aim to recreate the results of subjective IQA tests. That is why when IQA algorithms are tested, they are compared with databases that provide ground truth information in the form of subjective quality scores. The ground truth is based on the human perception and are often presented as the Mean Opinion Score (MOS) or Differential Mean Opinion

Score (DMOS) for all images in the database [Cha13]. A high quality IQA algorithm aims to give scores to images that are similar to the MOS or DMOS scores for the images in such a dataset.

Image reference Most IQA algorithms are known as full-reference (FR) algorithms. Meaning that checking the quality of an image requires an undistorted reference image that is used to calculate the visual quality of the target image. Most algorithms are based on FR-IQA and thus assume a reference image is available. In some cases this is not true, which is why some reduced-reference (RR) and no-reference (NR) algorithms have been designed. These algorithms work with little (RR) or no (NR) information to estimate the quality [Cha13].

IQA for synthesized views Assessing the quality of synthesized views is a special case of IQA for which there are specific problems to look for when trying to estimate the quality. The main new problem that can reduce the quality of the novel view are geometric distortions, objects that appear to be unnaturally bent or formed. This problem is introduced when the 3D data used for the generation of the synthesized view is inaccurate. Another problem is the reference image, or images in this case. Since there are multiple references, but none represent the exact image created, another way of assessing the quality is required. RR-IQA would still be possible since all pixels in the novel view are projected, but would require some extra steps that are not needed in normal IQA algorithms. Bosc et al. [Bos+11] created a database for IQA algorithms that objectively assess the quality of IBR generated synthesized images. They showed that regular IQA algorithms are not very suitable. Their dataset contains seven IBR algorithms and a set of synthesized views generated by each algorithm. They gathered the subjective MOS and DMOS scores for the results of each algorithm and compared the scores with several IQA algorithms. They show that normal algorithms do not match the scores of the subjective scores. A significant improvement in the algorithms can be made by taking the geometric distortions into account. Most of which are caused by the disocclusion problem. Since that is the case, IQA algorithms for synthesized views take edges into account as any inaccuracies are most likely found around the edges.

IQA algorithm for DIBR In our work we will be using the IQA algorithm of Farid et al. [FLG17] to evaluate the quality of the synthesized views. The proposed algorithm (DSQM) takes the views used to generate the novel view as reference to estimate the quality, making this a reduced reference IQA. The focus is on 3D distortion artifacts when estimating the quality. The algorithm uses signal phase congruency [Kov99] which targets the distortions around the edges, lines and corners. The algorithm was evaluated with the DIBR database [Bos+11] and had a high correlation with the subjective scores. The algorithm consists of three steps. In the first step the images used for the novel view are divided into patches and for each patch the corresponding patch in the novel view is found. In the second step the perceptual features of each patch are calculated using phase congruency. In the last step the resulting phase congruency maps are compared and the differences are calculated to estimate the distortion in the novel view patches. The average of the differences between the pairs of patches gives the final estimated quality.

Conclusion We are able to use the DSQM method to evaluate the synthesized views. The only problem is that the algorithm was made for DIBR algorithms

where only 2 input images are chosen for the creation of the synthesized view. Since our algorithm uses up to ten input views there is a small difference in the evaluation. Therefore we choose to alter the algorithm slightly, taking only images into account where the viewing angle is smaller than a certain threshold to avoid the comparison of patches with a large difference in viewing angle as those patches would not fully represent the same image part.

3.4 Depth fusion

Lastly we look at depth fusion algorithms. We will utilize a depth camera to enhance the image result generated by the Colmap algorithm. There has been some research in the area of fusing depth data from different sources of depth capturing techniques. Although our fusion will be from different sources, the techniques of fusing the depth data are similar.

Time-of-Flight and stereo fusion The most popular way to combine depth fusion data is through a Time-of-Flight (ToF) camera and stereo fusion. The reason for this being the complementing nature of the two depth data acquisition methods that allows for more accurate depth data. ToF is a fast system that does not depend on textures, but has a lower resolution and suffers from systematic errors. Passive stereo works great on high textured areas and has a high resolution, but a lower capture rate and is not able to estimate accurate depth values in textureless areas. In the fusion algorithms the ToF data needs to be upsampled and re-projected onto the stereo depth data in order for a fusion to work. In these fusion methods there are two categories, local and global fusing methods. Local methods are fast and can be parallelized, but can not deal well with local inaccurate data. Global methods tend to be a lot slower and the fusion is approached as an optimization problem [Nai+13].

Reliability fusion We look at the global methods, as our fusion algorithm is used as an offline pre-processing step and does not need to run real time. In the work of Zhu et al. [Zhu+10] they calculated the per-pixel reliability of each depth map and then fuses the data using a Maximum a priori-MRF (Markov Random Field) framework. They use Belief Propagation to globally optimize the results. The drawback is that it only works up to a range of one meter and the spatial resolution of the depth maps is quite low (400 x 300). In the work of Marin et al. [MZM16] they also generate reliability maps of the acquired depth data. Their method is based on the Locally Consistent (LC) method [Mat09], which is a locally global method that calculates the plausibility of the stereo correspondence using the geometric and photometric constraints. The plausibility is confirmed by the support of the neighboring pixels. Although LC is not a depth fusion method, in earlier work they extended the LC methods to work for two depth hypothesis [Dal+12]. The addition of Marin et al. was the reliability of each depth hypothesis when evaluating the plausibility, taking the complementary nature of both techniques of the depth capturing methods into consideration. Although the results look promising, they are still working with very low spatial resolution ToF depth maps.

Other sensors Although most of the work for depth fusion is centered around ToF and passive stereo because of their complementary nature and popularity of ToF

capture there are some other fusions as well. One of the methods is still centered around ToF, but looks to improve the ToF depth map with a Structured Light (SL) depth map to correct Multi-Path Interference (MPI) errors [AZ18]. In their method they use Maximum Likelihood to determine the depth values in a winner-takes-all approach. They calculate the reliability of the depth values and use a mixture of Gaussians model to determine the likelihood of each pixel. In their algorithm they take a patch centered around the depth hypothesis to calculate the weighted sum of the Gaussian distributions taking the distance to the pixel into account. The approach performs well, but is specifically centered around the MPI error.

Conclusion In our work we do not use a ToF camera, as described in section 4.1. However a lot of the principles of the depth fusion algorithms can be utilized for our depth fusion. Most successful methods seem to use some form of reliability for each depth hypothesis based on the qualities of the capture method used. Then a probability method is used to estimate which depth hypothesis is most likely to be correct taking the reliability into account. All described methods then pick the hypothesis in a winner-takes-all style.

4 Pilot

To determine where our focus lies, we first observe what quality we can achieve with our own collected data using color images only. Therefore this section will describe a pilot to test the quality of the 3D reconstruction. We will use a depth camera for the collection of the color images, as we predict that the results using the color images and Colmap depth maps only will not achieve the desired accuracy like with the other dataset. For this reason we will first describe what RGB-D camera we will use in section 4.1, focusing on the quality of the generated depth maps. Subsection 4.2 will describe how we collected the data for the reconstruction pipeline. We will conclude with a discussion of the results of the pilot in 4.3.

4.1 Depth camera

To be able to capture the environment we need a depth sensor that can make the most accurate depth images of the environment with the highest spatial resolution. There are four leading technologies of depth sensors: Structured light, Time of Flight (ToF), Stereo and Light Detection and Ranging (LiDAR) [O M+19].

Types of depth sensing Structured light uses a laser light source which projects a known pattern. The receiver then calculates the depth based on the distortion of the reflected pattern. Structured light performs best at close range and can handle low light pretty well. It is however slower than most technologies, making it less suitable for environments with dynamic obstacles. ToF uses the time it takes for light to travel from and back to the sensor to measure the depth. Generally ToF allows for fast capture, but the spatial resolution of the depth images is relatively low. The capture range depends on what the device focuses on. Stereo sensors use two cameras and compare the two images of the two horizontally displaced cameras to calculate the depth. The principle is based on the stereo vision of creatures in nature. The process is slower, but results in high resolution depth maps. Since it does not use active illumination the performance in low light is weaker than with other technologies. The range is usually up to ten meters, which makes it longer than the structured light sensors which generally can not measure accurately beyond 3.5 meters. ToF sensors can measure far beyond ten meters and can be considered for long range applications. LiDAR is a form of ToF and is an example of long range ToF which uses a pulsed laser to measure the time. LiDAR can have a very long range and can sense in 360 degrees. It is also considered the fastest depth sensing technology. The sensor is mostly used for capturing large environments outside and is less accurate at close range.

Requirements To be able to capture our data we need to determine which sensor to use. In order to make a proper selection we look at the technologies, the requirements for the depth images and the available consumer hardware. As for the requirements, there are a few important qualities we want for the sensor. First of all we need the spatial resolution to be as high as possible. We need to match the RGB and depth image, and the higher the resolution of the depth image, the more accurate data points we can use for the final depth data. We can upsample any depth image we get, but precision would suffer if we do that. Secondly we need the range to be far

enough. We will capture a relatively small room, but depth sensors that are meant for object recognition do not have a large enough range. Since the environment will be indoors, because the robot would operate indoors as well, we have to deal with low light situations as well. Lastly we look at the frame rate of the devices, which should be high enough for we are making video recordings in the environment. If the framerate is too low the overlap between images will be too low, resulting in a poor reconstruction quality.

Available cameras Looking at the available consumer 3D cameras we refer to a list created by Giancola et al.[GVS18b]. Table 1 shows the list of the cameras they discussed and contains only the relevant attributes for our application. As for picking the best camera, structured light sensors are less suitable, because their range is low and often they are slower. However since all cameras have at least 30 fps, which is fast enough for our capture, we can ignore the fps in the table. The bigger problem is the error accumulation over distance, since this is quadratic for structured light sensors. ToF cameras do not suffer from the falloff, have a long range and can measure inside well. However their resolution is low and we want to record a high resolution. We see that the Kinect V2, the leader in the market for several years, has been used for a lot of the DIBR researches. However, since we consider a high resolution the most important requirement and because Microsoft announced that the hardware is discontinued and no longer available, we will not pick a Kinect camera. Moreover it was shown that the standard calibration of the Kinect V2 is not good enough and requires a custom calibration in order to get decent quality depth images [Vil+17]. Due to these reasons we want to look at the much newer cameras in the Intel D4 series. There is also the StereoLabs ZED camera with a high range and resolution. However in case of low texture the passive stereoscopy is unable to properly reconstruct environments. Our environments have low textures so this device is not suitable.

Selection We consider to use the D415 or D435 camera as they offer a high resolution and a high enough range for what we want to achieve. Both show the potential for what we need. The difference between the two is that the D415 can capture more details on small objects, whereas the D435 is better at handling moving objects and has a broader field of view which minimizes blind spots [Int19b]. Since we are not working on dynamic scenes, the added benefit of the D435 does not help us. Therefore D415 will be the choice, as it has the highest theoretical precision of the two [GVS18a]. The camera uses an infrared projector to assist in the stereoscopy and this means that it will still perform well on textureless areas, which is known to be a big weakness for MVS based 3D reconstructions.

4.2 Data collection

For the capture of the data there are a few things to take into consideration. First of all we need a room that has enough features in it for the reconstruction to work. We select a room of the study association at the university, as it contains some objects there besides just desks and computers.

The recording We record a video of the room with the Intel D415 RealSense camera connected to a laptop on which the recording will be saved. For the recording we

Device	Technology	Range (m)	Resolution	Frame rate (fps)
PMD CamCube 2.0™	Time-of-Flight	0-13	200 x 200	80
PMD CamBoard™	Time-of-Flight	0.1 - 0.4	224 x 171	45
MESA SR 4000™	Time-of-Flight	0.8 - 8.0	176 x 144	30
MESA SR 4500™	Time-of-Flight	0.8 - 9.0	176 x 144	30
ASUS Xtion™	Structured-light	0.8 - 4.0	640 x 480	30
Occipital™	Structured-light	0.8 - 4.0	640 x 480	30
Sense 3D scanner™	Structured-light	0.8 - 4.0	640 x 480	30
Kinect V1™	Structured-light	0.8 - 4.0	640 x 480	30
Kinect V2™	Time-of-Flight	0.5 - 4.5	512 x 424	30
Creative Senz 3D™	Time-of-Flight	0.15 - 1.0	320 x 240	60
SoftKinetic DS325™	Time-of-Flight	0.15 - 1.0	320 x 240	60
Google Tango™Phone	Time-of-Flight	-	-	-
Google Tango™Tablet	Structured-light	0.5 - 4.0	160 x 120	10
Orbbec Astra S™	Structured-light	0.4 - 2.0	640 x 480	30
Intel SR300™	Structured-light	0.2 - 1.5	640 x 480	90
Intel R200™	Active stereoscopy	0.5 - 6.0	640 x 480	90
Intel Euclid™	Active stereoscopy	0.5 - 6.0	640 x 480	90
Intel D415™	Active stereoscopy	0.16 - 10.0	1280 x 720	90
Intel D435™	Active stereoscopy	0.2 - 4.5	1280 x 720	90
StereoLabs ZED™	Passive stereoscopy	0.2 - 20	4416 x 1242	100

Table 1: Comparison of consumer 3D cameras

use the factory calibration and extract intrinsic parameters for the reconstruction. There are several settings when using the RealSense camera and there are a few visual presets in the SDK that can be used. The “standard” preset is the most standard, but for our purpose we use the “high density” preset as this tries to get the most depth data as possible to fill in most of the holes. This profile also allows us to capture as much data as possible in all ranges and it sees more objects. Since we want the entire room with as much objects as possible for object detection purposes later on this profile fits our purpose best. There also are a few specifics to keep in mind when taking the recording.

- **Textures** Having as much textures as possible is important for our reconstruction, as it will allow SIFT to detect as many features as possible. If there are not enough features found during the reconstruction process, there will not be enough matches to determine the camera poses. For that reason we choose the room of the study association.
- **Illumination** It is also important not to have different illumination conditions. The main problem will concern specularities on shiny surfaces (e.g., computer screens or metal objects) and for high dynamic ranges (e.g., pictures against the sun or through doors/windows). However, we will need light in order to have enough contrast for the feature detection. We will try to block as much of the incoming sun as possible, while still trying to get enough ambient light for our capture.
- **High visual overlap** To make sure the tracks from the detected features can be generated we need enough visual overlap between two consecutive images. This is most important in the selection process, but will also mean that we need to move the camera slow enough to get images with a low motion blur between two selected images. Our aim is to have at least 70 to 80 percent overlap between two images.
- **Different viewpoints** In order to properly capture all objects we need enough viewpoints for a good reconstruction of the object. The process is similar to the previous one, where we need a recording with images with a low motion blur that can provide us with the viewpoints. The aim is to have viewpoints for each object that differ 5 to 15 degrees.

Data selection After the recording we extract all frames and select a subset based on a motion blur metric [Cre+07]. We do not want motion blur, because the feature detection will not work or select wrong features based on the blur. These features can not be matched with the features in other images because it will not represent a point in the room, but it will represent a motion. The same problem exists for the eventual synthesized image creation with DIBR. The goal of the selection is to make the dataset smaller, as the process would be very slow if we kept all images and just removed the blurred ones. So in the first step we create batches of 20 images and select the least blurry one. After that we manually remove images that still have too much motion blur and consecutive images where the overlap is too large.

4.3 Results

In our first capture we observed significant artifacts and a general lack of quality. Figure 4 shows the generated dense point cloud from the 3D reconstruction. As can be observed this looks more like a sparse point cloud and a lot of the textureless regions lack the information. Since this quality is far from what we need for our simulation we look at how to improve this result. In the rest of this section we will analyze what can be improved.

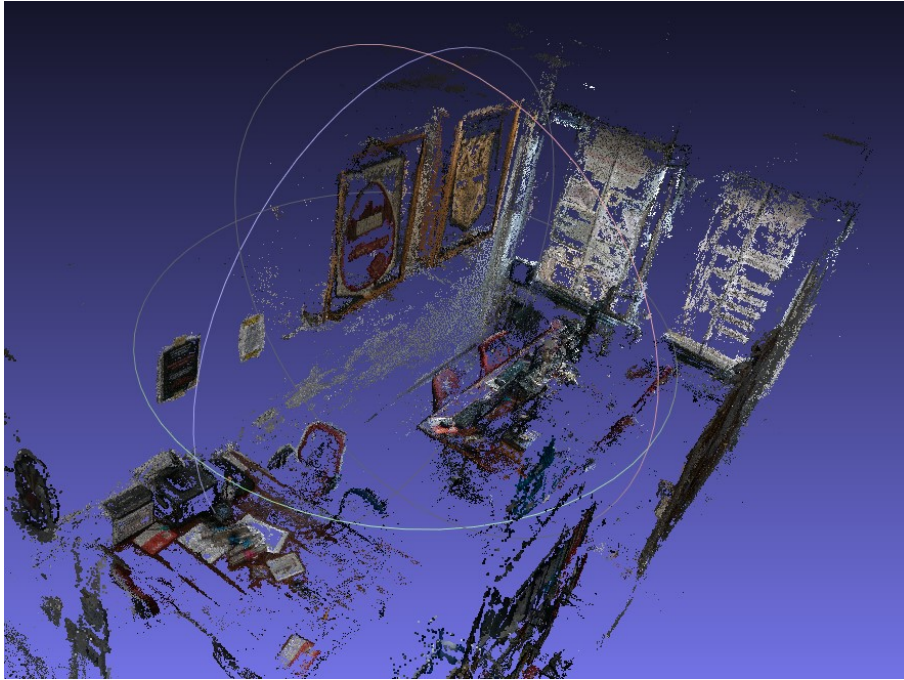


Figure 4: Resulting dense point cloud from the first reconstruction.

Utilize depth data The first optimization we look at is utilizing the depth data. Although we put some work in selecting a suitable depth camera, the current pipeline does not utilize this data yet. Since the captured data is a lot better at capturing textureless regions we expect we can use the data for the refinement of the generated depth maps. In figure 5 you see an example of a synthesized view made with the DIBR algorithm and it can be observed that a lot of depth data is missing (the black pixels). The view was generated with an RGB image at that exact location in combination with a second image with the most overlap. The generated depth data that belonged to those two RGB images were used for the novel view. Figure 5c shows the captured depth image for the shown RGB image. It can be observed that there is a lot of depth data present in this depth map that is missing in the novel view. Therefore we think that combining the synthesized depth map with the captured depth images can enhance the results.

Feature detection We can also look at the SIFT feature detection to see if we can improve the algorithm to detect more features on textureless areas. An example could be generating extra features by triangulating some of the detected features to generate new features that improve the matching process. We will try to increase the number of features by tuning some parameters in the Colmap feature detection

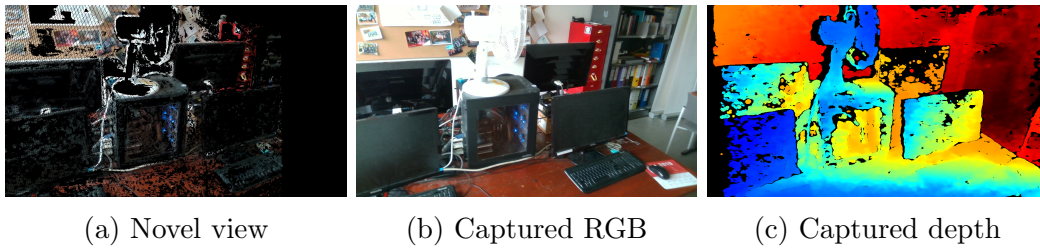


Figure 5: A novel view generated by the pilot (a), one of the captured RGB images used for the generation of the novel view (b) and the captured depth image matching the RGB image (c). The depth image has not been used yet, but has a lot of information the novel view is lacking.

algorithm.

Mesh Generating a global mesh from the point cloud will provide us with a more accurate and more compact description of the scene. The overall speed and quality of the novel view generation will improve, provided the accuracy of the dense point cloud is high enough. This also allows us to use some OpenGL functions for generating light conditions so the realism is increased.

Conclusion After picking the Intel D415 RealSense camera we captured a pilot at the study association room at the UU. After making a selection of the data and creating a reconstruction we can see that the quality of the results are too low for an accurate simulation. We presented several solutions for the collection of data, utilization of the data and changes in the reconstruction process. We will continue by looking at steps we will take to enhance the results.

5 Objectives

For this research we made a selection of topics we want to focus on. First we will create a dataset of a real world environment. We will use this data to reconstruct the room and use it in the simulated environment. We want to optimize the reconstruction process for better initial results. Secondly we want to utilize the depth data captured to improve the results of the DIBR algorithm.

Data collection Since the pilot was our first time capturing RGB-D data we still think we can improve that process. As part of the research we want to see if we can improve the initial results by improving the data we collect during the capture and the selection process. Our assumption is that we can generate better data that will yield a more accurate initial reconstruction. During the gathering of the selected data we also need to make sure the depth data is correctly aligned with the color data for the fusion later on.

O1: Create an RGB-D dataset that maximizes the initial 3D reconstruction quality with the Colmap software, as well as aligned depth maps to utilize in the depth fusion.

To find the best way to collect the RGB data we capture a few more RGB-D datasets with the D415 RealSense camera in the same room as before. This time we only capture a portion of the room, focusing on the large table near the windows. We evaluate different light conditions and look into parameter tuning within the Colmap software. We also look at the selection process and the amount of images needed for the best initial results. We use the point cloud generated by Colmap to evaluate the results of the different reconstructions, as the point cloud is the acquired depth data projected to 3D space. This way we can evaluate the average results of the depth maps more efficiently.

Combining depth data After the collection of correctly aligned depth and color data we can use the captured depth data to fill the holes and enhance the quality of the depth maps generated by Colmap. The depth maps generated by Colmap have a lot of missing data, which will be the primary focus of the refinement. We evaluate if we can use the captured depth data to enhance the accuracy of the depth image, which leads to a more accurate generation of the synthesized views. This leads to our main objective:

O2: Improve the quality of the synthesized views by using a fusion of the depth maps generated by Colmap and captured depth data as input for DIBR.

By using the measures taken by the depth camera we will evaluate if the results of the synthesized views can be improved in quality as opposed to just using the Colmap generated depth maps. Although the depth maps generated by Colmap can provide

decent synthesized images, it heavily depends on how well the 3D reconstruction in Colmap succeeds in synthesizing the required depth maps. By adding the extra measured data enhance the results of the synthesized images as well as making the quality less dependant on the results of Colmap when generating our own dataset.

6 Approach

In this section we describe what the plan is for determining the best way to gather the best data for the DIBR and depth fusion. The algorithm for the depth fusion is explained in subsection 6.2. In the next subsection we explain our method for evaluating the depth fusion results.

6.1 Gathering data

To find the way to collect the best data we look at the four guidelines for the optimal results: textures, illumination, high visual overlap and different viewpoints. When analyzing the pilot dataset we reconstructed we concluded that there is enough visual overlap and different viewpoints. With a total number of 404 images used, the size of the dataset is larger than the one used from Hedman et al. [Hed+16], which only has 226 high resolution images. As mentioned by the developers of Colmap, increasing the amount of images will not necessarily increase the quality of the reconstruction. We will look to reduce the number of images to speed up the reconstruction process, while still having enough overlap and viewpoints. As for the textureless areas, we want to fix that problem by using the measured depth data. The depth camera we chose is able to capture depth on textureless area a lot better, because of the IR pattern it uses for capturing depth data. For this reason we will not look to change anything in the room we capture, but instead observe if fusing the measured depth data with the depth data generated by Colmap can provide the needed data for textureless areas.

Illumination This leaves the illumination conditions that we can improve. To see what conditions would work best for us we record the same room as before, but to speed up the process we only record one of the tables to see if the results would improve for that part of the room. We tested three different illumination conditions. For the first one we let the sun in, although it is said this does not work very well for the reconstruction, the depth camera would be able to capture more precise depth data. If these results are similar to the other illumination conditions, we could try to make the full reconstruction in these conditions to have better measured depth data. For the second test we block as much sunlight as possible and rely on the artificial lights in the room. The last test will be performed without the artificial light, as that light can cause a lot of reflections which has a negative effect on the quality of the reconstruction. This means we purely rely on the ambient light that still enters the room after blocking away the sunlight. In short we will test three lighting conditions in a part of the room the pilot was captured in. These conditions are:

- Sunlight
- Artificial and ambient light
- Ambient light

Colmap Another optimization is tweaking the Colmap parameters. Their web-

site mentions¹ a few suggestions that can improve the results of the reconstruction. One of the suggestions could improve the number of matches. This includes setting the `-SiftExtraction.estimate_affine_shape`, `-SiftExtraction.domain_size_pooling` and `-SiftMatching.guided_matching` parameters to true. Another improvement mentioned is to account for weakly textured surfaces. In this case we increase the value of `PatchMatchStereo.window_radius` and decrease the value of `PatchMatchStereo.filter_min_ncc`. We will run a few reconstructions with different settings and empirically determine what the best result is based on the point cloud, making sure to alter the `StereoFusion.min_num_pixels` variable to account for noise as we should be able to reduce the noise with our depth fusion algorithm.

6.2 Depth fusion algorithm

Algorithm overview For our fusion algorithm we will be fusing passive and active stereo. Although different than fusing with ToF, we will be using the same principles as they remain the same. The first step is making sure the depth data is representing the same color pixel and depth scale. Since we captured the color images with our depth camera and use the SDK of the camera to align the depth maps real time we already have correctly aligned depth maps. However, there is a difference between the scale of the Colmap generated depth maps and the captured depth maps. Colmap is not able to accurately estimate distances in meters where the depth maps actually contain a scale conversion to mm. This means the same depth value represents a different distance in meters. Therefore the first step of the algorithm will be to determine the scale. After that we will be calculating the reliability of each depth hypothesis similar to previous works [Zhu+10; MZM16; AZ18]. Then we will fuse the depth data using the reliability maps and the probability of the correctness of the depth value by looking at neighboring values as well. As a last post-processing step we will use the global depth refining method to refine the resulting fused depth maps.

Scale In our baseline we use the mean scale between the two depth maps. Although the results are decent, there is a lot of noise in both depth maps. We will be using RANSAC to determine a more accurate scale, as it can filter out the outliers. For each depth image pair we calculate the scale using RANSAC and then take the mean of the resulting scales to get our final scale value. We then use this scale value to scale all the depth values of the captured depth maps to the scale of the Colmap depth map.

Fusion For the fusion of the two depth maps we will find the most likely depth value for each pixel based on the reliability and the probability the value is correct. We will be using the maximum likelihood to determine the fused depth map. This means the fusion algorithm looks to solve the following equation:

$$\hat{Z} = \arg \max_z ((R_s P_s)(R_c P_c)) \quad (1)$$

where R_s is the reliability map and P_s the probability map of the depth map synthesized by Colmap. R_c and P_c represent the reliability and probability map of the

¹<https://colmap.github.io/faq.html>

captured depth map and \hat{Z} is the resulting depth map that maximizes the reliability and probability of the depth maps for each depth value z . In the rest of this subsection we show the definition of the reliability and probability maps.

Captured reliability We define the reliability of the captured depth data R_c by two factors. The first part is the quadratic falloff of the depth measurement used. In the second part we add the confidence for textureless areas as this is the strength of the camera as opposed to the Colmap depth maps. Our formulation for the reliability of the captured depth data looks like:

$$R_c = \gamma R_d + (1 - \gamma) R_{tl} \quad (2)$$

where R_d is the confidence map of each depth value based on the distance, R_{tl} is the confidence map based on the textureless areas and γ is a constant defining the weight of the two factors. The equation calculates the reliability of each depth value in the depth map where 1 is the most reliable and 0 shows the depth value is not reliable at all.

Distance reliability For the distance factor we take the quadratic falloff of the depth sensor into account. We also look at the minimum and maximum distance the device is able to measure. According to the document by Intel [Int19a] about the depth camera, the minimum measured depth can be defined as:

$$MinZ(mm) = focallength(pixels) * baseline(mm) / 126 \quad (3)$$

The maximum theoretical distance is 60 meters, which is way beyond the size of the room. Instead we will be using a maximum distance based on the size of the room. Using this we map the estimated reliability of the distance quadratically between the minimum and maximum distance as:

$$r_d = \begin{cases} \frac{MaxZ^2 - z^2}{MaxZ^2 - MinZ^2}, & \text{if } MinZ < z < MaxZ \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where z is the depth value. To get R_d we calculate r_d for each depth value z in the captured depth map. The resulting reliability map tells us how reliable the depth values in the captured depth map are based on the distance measured.

Textureless reliability The camera is able to capture textureless areas where the Colmap method has a hard time in getting accurate values in these areas. This is why we included R_{tl} in the calculation of the reliability of the captured depth map. We calculate R_{tl} by first making range images based on the color values. After that we use OpenCV's contour detection algorithm to calculate the contours of each range image. Then we calculate the number of pixels in that area and if it is above a threshold α we define the whole area as textureless and assign each pixel with the value 1. All pixels that did not get a 1 value after evaluating each range image, are textureless and will get a 0 assigned. The algorithm is based on feature detection methods looking for features in locations with high color contrasts, making low color differences featureless. This is why textureless areas are hard for stereo algorithms as large regions with color similarity are hard to match.

Colmap reliability For the reliability of the stereo algorithm we look at the photometric and geometric differences. The MVS algorithm first determines the depth

hypothesis with photo consistency costs and outputs the depth maps based on those costs. It then also looks at the geometric consistency of all images, resulting in a second depth map. Comparing the two depth maps will provide the confidence of the resulting depth value. So we linearly map the difference, giving us the confidence of the Colmap depth map:

$$r_s = \begin{cases} \frac{\beta - |z_p - z_g|}{\beta}, & \text{if } |z_p - z_g| < \beta \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where z_p and z_g are the photometric and geometric depth values respectively and β is the maximum accepted difference. We use a maximum accepted difference as in some cases the difference is very large and the depth value is most likely noise. The result r_s tells us the reliability of a depth value based on the geometric and photometric difference in the location of z in the Colmap depth map. We get the reliability map R_s by calculating r_s for every z in the Colmap depth map.

Fusion All that is left is calculating probability maps P_s and P_c . To calculate the probability of a depth value z in either the captured or Colmap depth map we will look at the neighboring depth values. We define a support region S_f centered around the location f of depth value z . We assume the scene consists of flat surfaces, therefore we define a plane based on the value z and its neighbours. The probability is calculated by checking for each value $g \in S_f$ if g is on the plane. The further away the depth value at g is from the plane the less support g can give to the correctness of z . Since the scene will not consist of a single plane we also assume that the further away g is from f , the more likely it is that this value is on another plane. Combining these assumptions we define the probability of g supporting the depth value z at f as $P(E_{f,g}|\Delta_{f,g}^p)$ where $E_{f,g}$ is the event that encodes the distance between f and g and $\Delta_{f,g}^p$ is the distance of g from the plane. Since we assume the distance to be zero we can write according to Bayes theorem:

$$P(E_{f,g}|\Delta_{f,g}^p) \propto P_p(E_{f,g}) * P_l(\Delta_{f,g}^p|E_{f,g}) \quad (6)$$

where $P_p(E_{f,g})$ is the prior and $P_l(\Delta_{f,g}^p|E_{f,g})$ the likelihood. Since we assumed that points closer to f are more relevant we define the prior as:

$$P_p(E_{f,g}) = e^{-\frac{\Delta_{f,g}}{\gamma_d}} \quad (7)$$

where $\Delta_{f,g}$ is the euclidean distance between points f and g and γ_d is a constant. In the equation we map the prior exponentially, based on the euclidean distance between f and g , where 1 would mean f and g are the same and zero means that g is infinitely far away from f .

Plane To determine the likelihood P_l for g supporting depth value z at f we first need to determine the plane defined by f and its neighbours. For the definition of the plane we use the plane equation:

$$0 = \vec{n} \cdot (\vec{r} - \vec{r}_0) \quad (8)$$

where in our case $\vec{r} = \{g.x, g.y, g.value\}$ represents the point g in 3D space with the depth value at g as the z component and $\vec{r}_0 = \{f.x, f.y, z\}$ the point f in 3D space.

The normal is unknown and needs to be calculated. We determine the normal by looking at neighboring pixels and selecting two pixels h_1 and h_2 that will make two vectors perpendicular in x and y if the vectors are drawn from f to both points. We refer to figure ?? for a visual explanation. The normal is then calculated as:

$$\vec{n}_1 = (h_1 - r_0) \times (h_2 - r_0) \quad (9)$$

where h_1 and h_2 are the selected neighboring points in 3D space and the resulting normal is the normal for the plane with points h_1 , h_2 and r_0 on them. We calculate eight normals in total by rotating h_1 and h_2 clockwise to get the next combination of neighbours to calculate a new normal. We refer to figure ?? for a visual representation. For the calculation of the normal \vec{n} we take the mean of the resulting normals \vec{n}_1 from equation 9 and normalize the result.

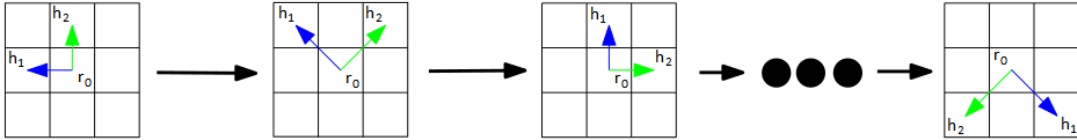


Figure 6: The calculation of the normal for the plane visualized. The normal is calculated with the blue and green vector, which are 3D vectors for the depth values of r_0 , h_1 and h_2 are taken into account as the z component. In the second image we select a new h_1 and h_2 resulting in new vectors from r_0 to h_1 and h_2 and thus a new normal. The other images show the rotation that results in new values for the h_1 and h_2 , which how we got eight normals after 8 rotations.

With the normal for the plane equation we can go back to the calculation of the likelihood $P_l(\Delta_{f,g}^p | E_{f,g})$. We model $\Delta_{f,g}^p$ as the distance between g and the plane calculated for f . We then get:

$$\Delta_{f,g}^p = |\vec{n} \cdot (\vec{r} - \vec{r}_0)| \quad (10)$$

where \vec{n} is the calculated normal using f and its neighbours, \vec{r} is the 3D points in g and \vec{r}_0 is the 3D point in f . This results in the distance of the depth value in g from the plane calculated for f . Assuming that (10) is Gaussian distributed we combine the equations to get the posterior defining the probability of g belonging to the support of f as:

$$P(E_{f,g} | \Delta_{f,g}^p) \propto e^{-\frac{\Delta_{f,g}}{\gamma_d}} * e^{-\frac{\Delta_{f,g}^p}{\gamma_z}} \quad (11)$$

where γ_d and γ_z are used to control the equation. This equation gives us the probability that point g supports the depth value at f , meaning that if the probability is high for a g , the probability that z at f is correct is also high. To determine the final probability that z is correct we accumulate all probabilities $P(E_{f,g} | \Delta_{f,g}^p)$ for each $g \in S_f$ giving the following equation:

$$P(f) = \sum_{g \in S_f} P(E_{f,g} | \Delta_{f,g}^p) \quad (12)$$

The final equation maps the support of each point g in the support region where a value closer to 1 means that the probability of f being correct is very high. We use $P(f)$ to calculate the probability for every depth value in the Colmap and captured depth map to get the probability maps P_s and P_c . This means we now have all four maps needed in equation 1 and after solving that we get our final resulting depth map. For an overview of the algorithm to fuse a full set of depth maps we refer to algorithm 1.

Algorithm 1 Depth fusion

Input: $Z_1^{col} \dots Z_N^{col}$, $Z_1^{cap} \dots Z_N^{cap}$, $Z_1^p \dots Z_N^p$, $Z_1^g \dots Z_N^g$, M as # pixels

Output: Set of N depth maps \hat{Z}

```

1: function DEPTHFUSION( $\gamma$ )
2:    $scale \leftarrow 0$ 
3:   for  $i \leftarrow 1$  to  $N$  do
4:      $scale \leftarrow scale + RANSACscale(Z_i^{col}, Z_i^{cap})$ 
5:   end for
6:    $scale \leftarrow scale/N$ 
7:   for  $i \leftarrow 1$  to  $N$  do
8:      $Z_i^{cap} \leftarrow Z_i^{cap} \times scale$ 
9:      $R_c \leftarrow \gamma \times QuadraticDist(Z_i^{cap}) + (1 - \gamma) \times Textureless(Z_i^{cap})$ 
10:     $R_s \leftarrow |Z_i^g - Z_i^p|$ 
11:     $P_c \leftarrow 0$ 
12:     $P_s \leftarrow 0$ 
13:    for  $k \leftarrow 1$  to  $M$  do
14:       $P_c \leftarrow P_c + PlaneSupport(Z_i^{cap}(k))$ 
15:       $P_s \leftarrow P_s + PlaneSupport(Z_i^{col}(k))$ 
16:    end for
17:     $\hat{Z}_i \leftarrow Max(R_s \times P_s, R_c \times P_c)$ 
18:  end for
19:  for  $i \leftarrow 1$  to  $N$  do
20:     $\hat{Z}_i \leftarrow Refine(\hat{Z}_i)$ 
21:  end for
22:  return  $\hat{Z}$ 
23: end function

```

6.3 Depth fusion evaluation

We would like to compare the results of the depth fusion algorithm with several ground truth depth maps to evaluate the results. A comparison with ground truth depth images is the easiest way to analyze the depth maps generated by our algorithm. However, there are some problems in obtaining ground truth for our experiment. In order to properly evaluate our algorithm we either need a dataset contain-

ing ground truth or we have to generate our own ground truth. Unfortunately, we cannot use existing ground truth datasets as they do not contain a combination of:

- Ground truth depth maps
- Depth images captured by an active stereo camera
- Depth maps generated by the Colmap software

The problem is that we need a very specific dataset that matches the dataset we capture ourselves, as the algorithm is designed for specifically Colmap generated depth maps and depth data captured by an active stereo camera. So it does not make sense to use another dataset to evaluate our results, meaning we cannot use ground truth captured in other datasets. The other option would be generating our own ground truth. Setting up such ground truth would require some precise measurements to be taken with a specific setup of a new color camera and the depth camera. Additionally we would have to create an algorithm like in the Middlebury dataset [Sch+14]. The combination of accurately capturing the required data and creating the algorithm to have reliable ground truth would cost a lot of time. Another problem with this approach is that some of the recorded data would no longer be useful. We could return to the room to capture the ground truth, but the whole room will have changed since it is used daily. This means that the room changed so much that even in the static situation only the walls remain the same and new images cannot be added to the reconstruction without creating a new dataset. Since we want to focus on creating the fusion algorithm we decided that we will evaluate our results a different way.

Evaluation through IQA The goal is to see if we can improve the quality of the novel views by the fusion of the Colmap and captured depth images. Without ground truth we will evaluate the depth image by assessing the image quality of the novel view with an IQA algorithm. We can compare trajectories of 30 consecutive moves from the Pepper robot in the simulation. Each move in the trajectory will generate a new novel view generated from the virtual camera pose of Pepper. Calculating the mean IQA score of such a trajectory provides a value we can compare. We will take three trajectories for each dataset. In the comparison the set of color images will stay constant for each dataset. The DIBR algorithm will stay the same for the whole experiment. The IQA algorithm in the next subsection works by detecting geometric distortions introduced by inaccurate and noisy depth maps. This means that the best way to improve the quality scores is to improve the depth input of the DIBR algorithm and that is why we use IQA as the evaluation method.

Comparison With the DSQM algorithm we can evaluate the results of our fusion algorithm. For the evaluation we would have a constant DIBR and evaluation algorithm as well as a constant set of camera poses and color input for each trajectory for each set of depth images. This leaves us with the sets of depth images to compare in the simulation. In table 2 we show an example of what the evaluation would look like. Our goal is to improve the Colmap depth maps. We will use the depth refinement to fill in the missing information and perform the comparison on the following depth datasets:

- Colmap depth maps

		DSQM mean scores for each depth map set		
		Colmap refined	Baseline refined	Fusion refined
Study room	Trajectory 1	0-1	0-1	0-1
	Trajectory 2	0-1	0-1	0-1
	Trajectory 3	0-1	0-1	0-1
ING room	Trajectory 1	0-1	0-1	0-1
	Trajectory 2	0-1	0-1	0-1
	Trajectory 3	0-1	0-1	0-1

Table 2: Example of the evaluation table. The trajectories consist of 30 unique novel views with 30 unique camera poses for Pepper. The 0-1 represents the mean DSQM score for those 30 novel views, generated for each depth map set.

- Baseline of the fusion
- Algorithm results

For the baseline we will use an algorithm which fuses the depth maps in the simplest way possible. This means scaling the captured depth image values to the corresponding scale of the Colmap values calculated by taking the mean of each pair of relevant values in the complete set of depth images. Relevant values would be values greater than zero, which is depth data where no value was found and greater than 20.000. We choose 20.000 because that would translate to 20 meters for the captured data. Since the room is not larger than 20 meters every value above is considered noise. By scaling the captured image down to the Colmap scale we can fill in every zero value in the Colmap depth map with a value from the captured depth map. As a last step we use the depth refinement to fill in the last missing information and improve the quality as we did with the Colmap images. The last part of the comparison is the set of depth maps that was generated by our fusion algorithm.

7 Experiments

In this section we describe how we conducted the experiments. In the first subsection we discuss the datasets captured and how we obtain the best results to complete **O1**. In the next subsection we discuss the parameters used in our depth fusion algorithm and the different kind of trajectories we chose to evaluate.

7.1 Dataset

To get the best data we took the recording of the table in the first room under the different light conditions:

- Sunlight
- Artificial and ambient light
- Ambient light

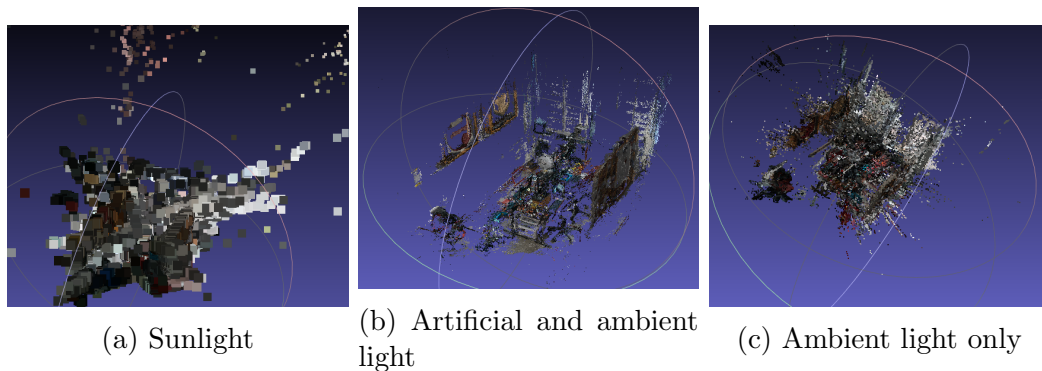


Figure 7: The resulting point clouds under the three different light conditions.

Results illumination In figure 7 the resulting point cloud under all three conditions is shown. The first observation is that using the sunlight results in an inaccurate reconstruction that we can not use in combination with the captured depth data. The camera poses and synthesized depth data are wrong in most cases which is shown in the resulting point cloud. The results of using artificial light compared to the ambient light are very similar. However the ambient light seems to have a lot more outliers which can be explained by the dark corners in the room. These dark corners make it hard for the reconstruction to calculate the depth, causing these stretched corners in the room. Based on these results we decide to create a full reconstruction of the room with a combination of artificial and ambient light. We create the balance of enough light, reducing reflections and making sure the whole room has enough light to be captured.

Full capture For capture of the full room we took videos of the whole room under the light conditions that worked best. Another point of attention was to reduce the amount of images, to speed up our reconstruction. We focus more on shots that show the whole room and less on shots with too much overlap. As a last optimization step we try to tweak some of the parameters in Colmap. As a result we

increased the *window_radius* to 7 for the stereo patch match to better handle textureless surfaces, which results in a small increase in quality. We also try to increase the number of matches by activating *estimate_affine_shape*, *domain_size_pooling* and *guided_matching*. This does not result in a higher quality reconstruction which means that the number of matches in this room is high enough already. The results of this final capture is depicted in figure 8. The results are a little better than the initial results, but as expected the quality is still lacking. In appendix A all Colmap parameters used can be found.

Second room We create a second dataset that keeps object detection and navigation a little better in mind. For this dataset we use a room at ING which is larger than the previously recorded room. In this room we put two tables in the center with several objects that can be used for object detection algorithms. The room size also allows for more possibilities regarding navigation algorithms, as there is more space to move around. Since the room is larger the amount of images has increased as well, making sure that there is enough coverage for the entire room to perform DIBR. We will use the room for the evaluation for the depth fusion algorithm as well. The resulting point cloud of this recording can be found in figure 9.

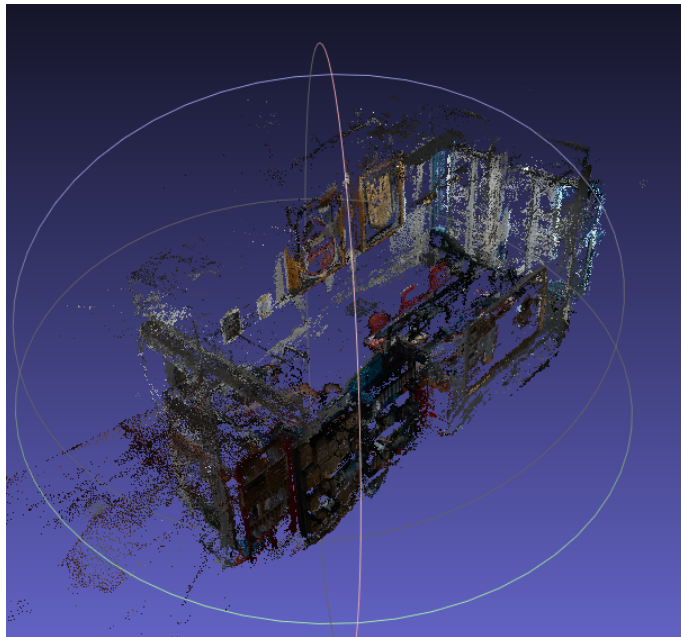


Figure 8: The resulting point cloud of the final reconstruction of the first room at the university.

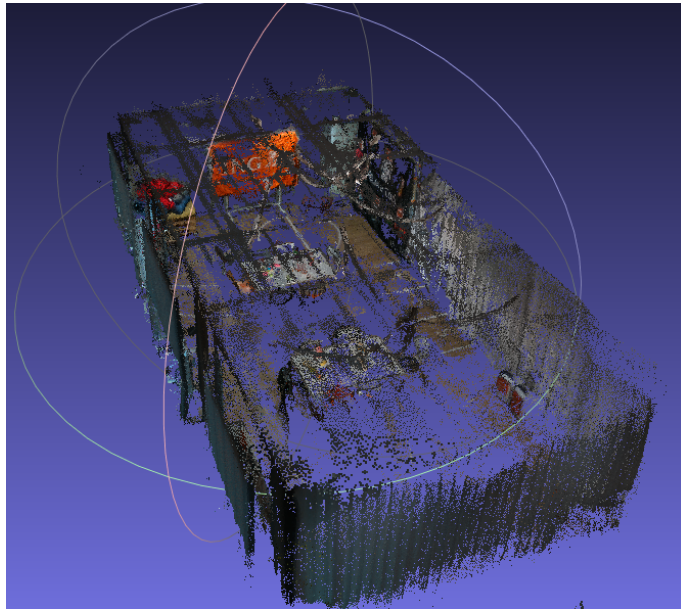


Figure 9: The resulting point cloud of the reconstruction of the second room at ING.

7.2 Depth evaluation

Fusion parameters To evaluate our depth fusion algorithm we first run the depth fusion on all depth pairs in both datasets created. We empirically determined the parameter values needed for the fusion algorithm. We set γ at 0,5 and define $\max Z$ as 20.000 which translates to 20 meters. For the control parameters in equation 11 we choose γ_d 4 and γ_z 1. With these parameters we run the fusion algorithm on each pair of depth maps and used those results in our evaluation.

Trajectories The next step is to create three trajectories for both datasets with at least 30 camera poses for the robot in each trajectory. We evaluate three different trajectories in the room where the first trajectory is turning around in a full circle in the center of the room to get some variation in what the robot will see. We will be referring to that trajectory as “*circle trajectory*” from there on. The circle trajectory in the middle of the room should give a general idea of how well the fused depth images perform overall. The second trajectory will start in the corner of the room and will consist of images with large distances, from now one referred to as “*distance trajectory*”. For this trajectory we aim to evaluate the effect of larger distances, as the depth camera would measure depth values less accurate. We will evaluate the influence of the quadratic falloff in our algorithm. The last trajectory will look at textured areas from a relatively close distance, from now on referred to as “*texture trajectory*”. We expect the images with a lot of texture to yield better initial results in Colmap.

8 Results

In this section we will discuss the results of the DSQM scores of the trajectories. Table 3 shows the mean scores of the DSQM algorithm for each set of depth maps for each trajectory. We start the evaluation by comparing the differences in the mean values for the trajectories to see if the fusion algorithm does improve the quality of the synthesized images. We will also look at some of the differences between the trajectories and datasets and give an explanation for the potential reasons for these differences.

Mean value differences In table 4 we show the differences between the trajectories to compare the sets of depth maps. The comparisons include the comparison between:

- Colmap and baseline
- Colmap and fusion
- Baseline and fusion

For every comparison the mean value of the second depth map set is subtracted from the mean value of the first depth map set. So for the first column that means that we subtracted the mean value of the baseline from the Colmap mean value. As can be seen in the table the baseline and fusion get better results with the exception of the distance trajectory in the ING dataset. We will discuss the reason for this later. First we will check whether these differences are meaningful, or created by chance. In other words, if the difference actually means the depth map set improves the quality of the synthesized images or whether it was a coincidence that the mean values are better.

		Colmap	Baseline	Fusion
Study room	Circle	0.017055	0.015843	0.014961
	Distance	0.019276	0.018812	0.016708
	Textures	0.012229	0.011641	0.011529
ING room	Circle	0.017723	0.017087	0.016912
	Distance	0.011958	0.012052	0.0126
	Textures	0.011342	0.010955	0.010442

Table 3: DSQM mean scores for each depth map set

		Colmap - baseline	Colmap - fusion	Baseline - fusion
Study room	Circle	0.001211911	0.002093544	0.000881633
	Distance	0.000463494	0.002569035	0.002105541
	Textures	0.000588558	0.000700041	0.000111483
ING room	Circle	0.000636	0.000811	0.000175
	Distance	-9.4E-05	-0.000642	-0.000548
	Textures	0.000387	0.0009	0.000513

Table 4: Differences in mean DSQM score for all trajectories

		Colmap - baseline	Colmap - fusion	Baseline - fusion
Study room	Circle	0.009141346	0.000497099	0.308076608
	Distance	0.160954614	1.09924E-07	2.51971E-08
	Textures	0.000762134	0.817337769	0.899482382
ING room	Circle	0.074945916	0.176916029	0.330334411
	Distance	0.182959485	1.42713E-06	0.247359525
	Textures	0.131757819	0.008866781	0.00852439

Table 5: P values for each trajectory performing the D’Agostino-Pearson test to determine whether the data is normally distributed. P values lower than alpha 0.05 are considered to be distributed non-normally.

8.1 Statistical analysis

t-test To verify the differences in the means we run a paired sample t-test, a statistical procedure that determines whether the mean value difference between two sets of observations is zero. If the difference turns out to not be zero we can say the difference is not caused by chance. We choose the paired sample t-test, because all scores are evaluated with the same virtual camera pose for the robot and with the same algorithmic procedure. The only difference between the measured observations is the depth input data used. Since we use a continuous scale for our measurements we can use the test. There are two assumptions for the t-test we should look at before we can run the t-test though. We will check whether the data is normally distributed and whether there are no outliers. The t-test is said to be quite robust, but we want to verify these assumptions.

Normality We will start by checking whether the data is normally distributed. The first step is to visualize the data to check for normality. For the visualization we choose to create a Q-Q plot of every sample. In the plots the points should be close to or on the line if they represents a normal distribution, since the line represents the normal distribution. The resulting graph for each sample can be found in figure 10 for the trajectories in the study room dataset and in figure 11 for the trajectories in the ING room. Analyzing the plots does not give us a clear answer whether the data is normally distributed or not. For example, the plots representing the circle trajectory in the ING dataset seem like they are normally distributed, but the plot for the baseline in the distance trajectory for the ING dataset deviates relatively far from the line. To verify the Q-Q plots we did a normality test based on the kurtosis and skew factors. We ran a D’Agostino-Pearson test which calculates how much the values differ from the expected values for a normal distribution. The test computes a P value that sums up the differences and compares it with a chosen alpha of 0.05. This means that if the P value is lower than 0.05, the distribution is significantly different from the normal distribution. All P values calculated with the D’Agostino-Pearson normality test are shown in table 5. Most samples from the ING dataset are normally distributed, but five samples from the study room and three samples from the ING dataset are not normally distributed. We will first look at the potential outliers before we conclude whether the data is normally distributed or not.

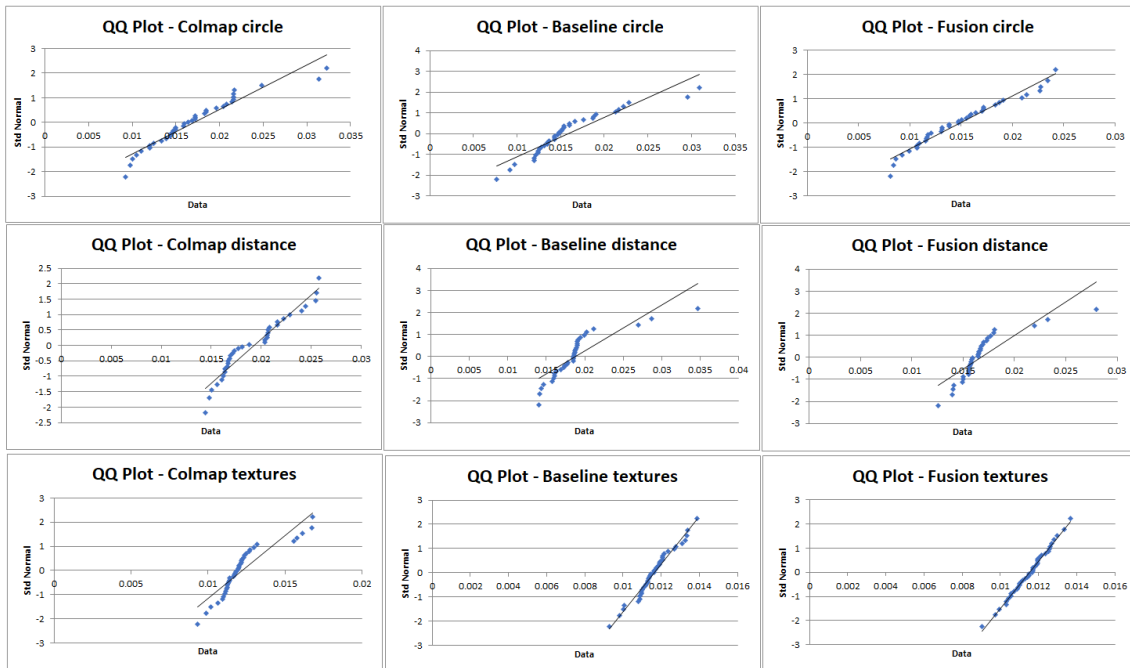


Figure 10: The Q-Q plots representing the collected data in the three different trajectories of the study room dataset. The line represents the perfect normal distribution and points represent the measured data points.

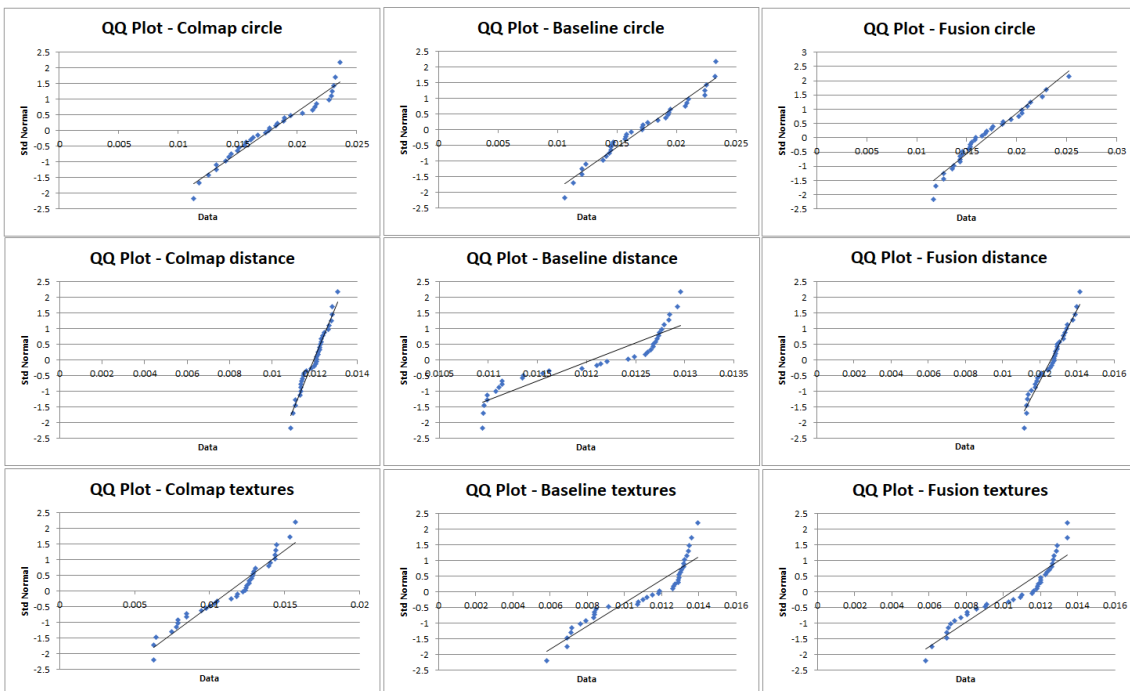


Figure 11: The Q-Q plots representing the collected data in the three different trajectories of the ING room dataset. The line represents the perfect normal distribution and points represent the measured data points.

Outliers To determine whether we have outliers we first visualize the data in box

plots that include outliers and analyze those results. The box plots can be found in figure 12 for the study room dataset and in figure 13 for the ING dataset. As can be seen, the results from the ING dataset do not contain any outliers, whereas the dataset from the study room contains several in some of the samples. In order to evaluate what causes the outliers we look at the depth map of the first image selected for the DIBR. This image will fill most of the synthesized image and has the highest influence on the reconstruction of the synthesized image. In all three trajectories the outliers are generated with the same first color and depth image so we analyze that depth image.

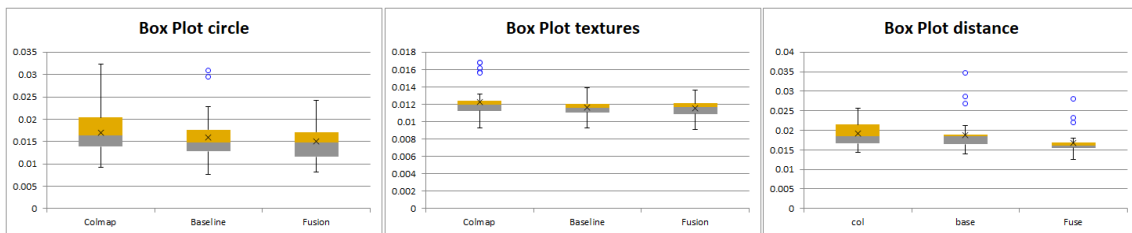


Figure 12: The box plots with outliers representing the data in the study room. Each plot represents all three samples in a trajectory.

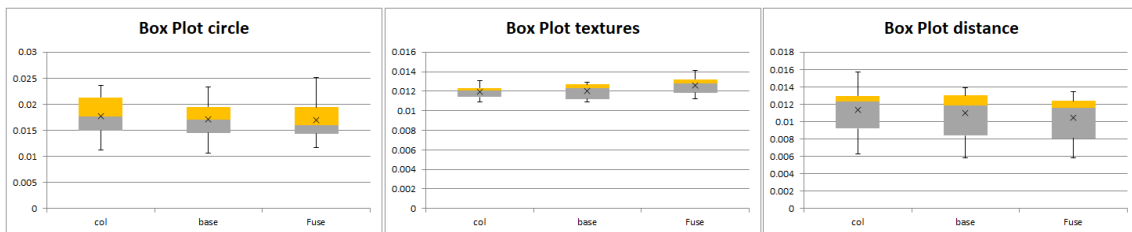


Figure 13: The box plots with outliers representing the data in the ING dataset. Each plot represents all three samples in a trajectory.

Circle trajectory The depth maps used for the reconstruction in the circle trajectory of the image reported as outliers can be found in figure 14. The outlier is reported for the baseline, so we compare the baseline depth map to the Colmap and fusion depth map to see what caused the outlier. As can be seen the depth map generated by the baseline contains a lot of noise and is likely to be the reason for the outlier. The score of the synthesized images using the Colmap depth map is even worse, but since the overall scores for these results are higher as well, it is not regarded an outlier. This means that the baseline does not improve either in the rest of the sample, but since this is a result of the used data we will not remove the outlier as the result is still valid.

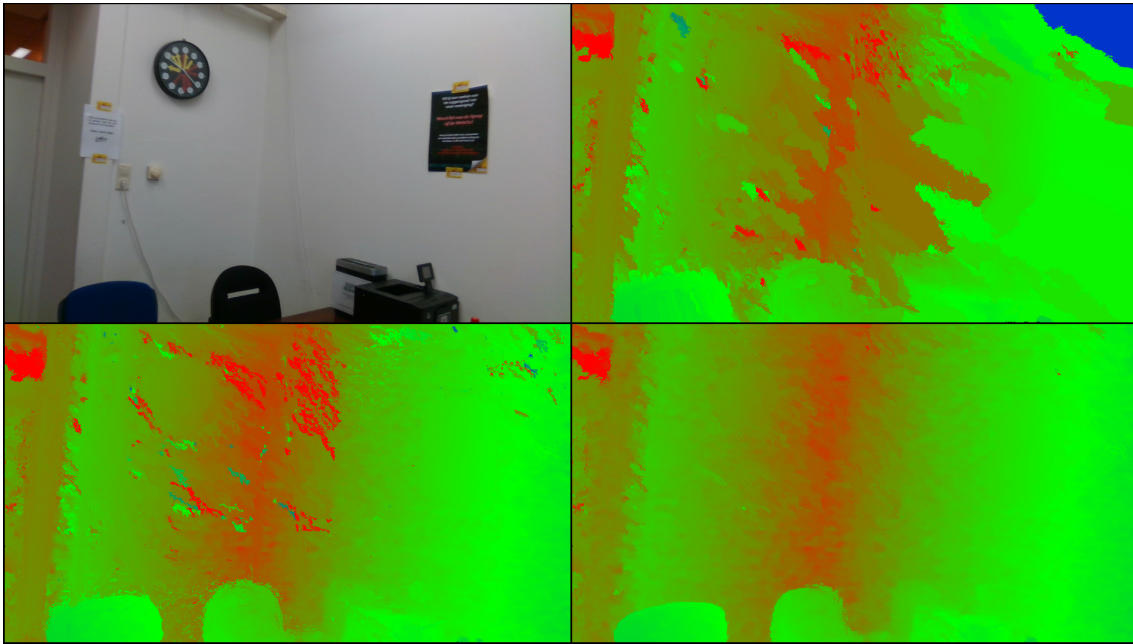


Figure 14: Depth maps underlying the synthesized images that are considered outliers in the circle trajectory of the study room dataset. From top left to bottom right: color image, Colmap depth image, baseline depth map and fused depth map.

Texture trajectory As for the texture trajectory, we want to see where the Colmap depth set outliers come from. Analyzing the depth maps in figure 15 shows that the Colmap depth map used for the synthesized image that resulted in outliers is relatively bad. It should be taken into consideration that the scores for the texture trajectory are lower overall, so in the entire population this is not considered an outlier. It shows that when a part of the image is textureless (the left side), the quality decreases a lot.

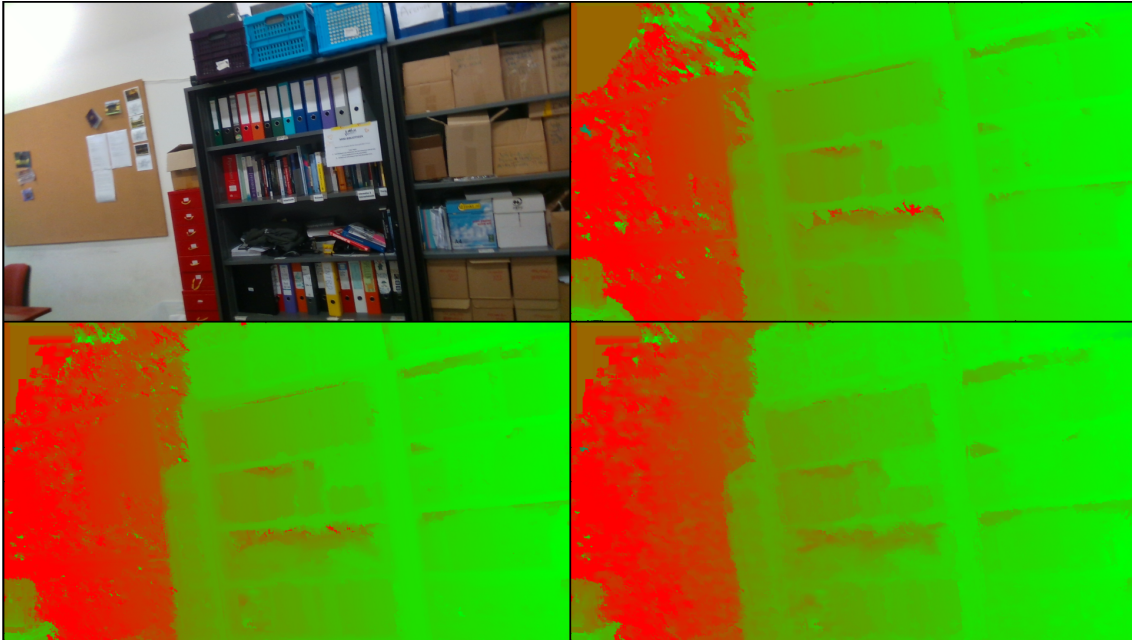


Figure 15: Depth maps underlying for the synthesized images that are considered outliers in the texture trajectory of the study room dataset. From top left to bottom right: color image, Colmap depth image, baseline depth map and fused depth map.

Distance trajectory For the final trajectory the outliers are located in the baseline and the fusion algorithm depth map sets. The first depth maps used for synthesizing the outlier images are located in figure 16. Although the baseline and the fused depth map both look better at first, it can be seen that the left side of the image contains a lot of noise. This is caused by the captured depth map, where values concerning the left of the image are unreliable and often zero because of the nature of the stereo algorithm of the camera. Analyzing one of the synthesized images considered as an outlier in all three depth maps we see that the left side is indeed worse (shown in figure 17). If we compare that to the results in the same trajectory (shown in figure 18) which lies inside the box plot we can see that there is no noise on the left, so we can confirm that the left side of the depth map is the cause of the outlier. Since this again is caused by the depth maps it should be evaluated as well.

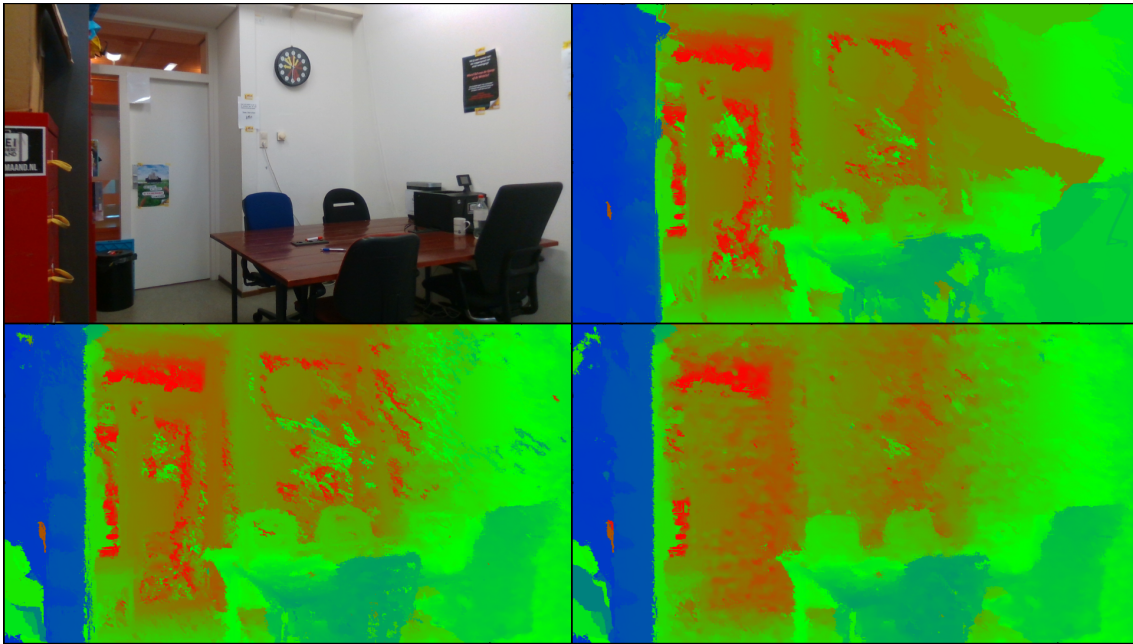
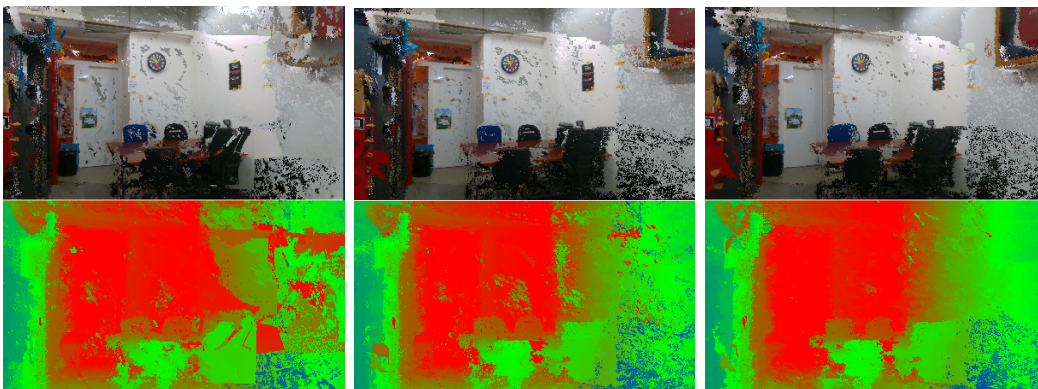
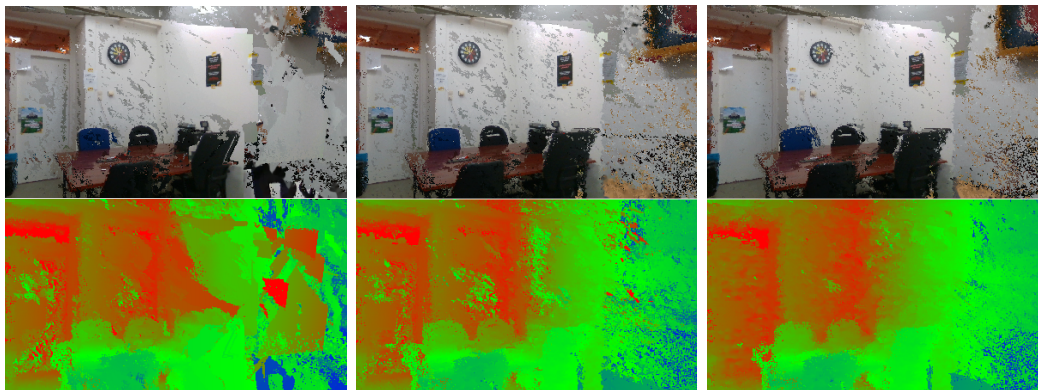


Figure 16: Depth maps underlying the synthesized images that are considered outliers in the distance trajectory of the study room dataset. From top left to bottom right: color image, Colmap depth image, baseline depth map and fused depth map.



(a) Synthesized image with the Colmap depth maps (b) Synthesized image with the baseline depth maps (c) Synthesized image with the fused depth maps

Figure 17: Synthesized image from the distance trajectory in the study room dataset. The images made with the baseline and fused depth maps were flagged as an outlier when analyzing the box plot.



(a) Synthesized image with the Colmap depth maps (b) Synthesized image with the baseline depth maps (c) Synthesized image with the fused depth maps

Figure 18: Synthesized image from the distance trajectory in the study room dataset. These images are used to compare to the outliers of figure 17

Significance Since the samples from the study room show a lot of non-normal distributions and contain some outliers we do not want to remove we assume the population to be not normally distributed. The likely reason for this is the high variance introduced by some of the depth maps that produce some bad results in this dataset for all depth map sets. As a result we will not use the paired t-test for the study room dataset. Instead we will use the non-parametric substitute, the Wilcoxon Signed-Rank test, to evaluate whether the mean values are significantly different. We therefore run the test on each trajectory on each combination like with the D’Agostino-Pearson test. We set the null hypothesis as a difference of zero in the mean values and we set alpha at 0.05. The resulting P values of all tests can be found in table 6 where any value lower than 0.05 means that the difference between the two mean values was not a coincidence. In the ING dataset we see that the data is closer to the normal distribution and it does not contain outliers. Therefore we assume the population in the ING dataset to be normal distributed and in this case we will evaluate the differences in the mean values with the paired t-test. Again with the null hypothesis as a difference of zero and we set the value of alpha at 0.05 and show the resulting P values in table 7. We will analyze the results of the tests of each trajectory in the next subsection.

		Colmap - baseline	Colmap - fusion	Baseline - fusion
Study room	Circle	0.000776991	9.92687E-05	0.002991924
	Distance	0.143003028	8.78535E-06	1.05239E-07
	Textures	0.000188566	0.003595216	0.419708094

Table 6: P value results from the Wilcoxon Signed-Rank test for every comparison in the study room dataset. P values lower than alpha 0.05 show that the difference in mean values is not a coincidence.

		Colmap - baseline	Colmap - fusion	Baseline - fusion
ING	Circle	0.002142632	0.043725015	0.658508348
	Distance	0.138166896	3.12E-07	5.05E-07
	Textures	0.030923837	6.82E-06	9.14E-09

Table 7: P value results from the paired t-test for every comparison in the ING dataset. P values lower than alpha 0.05 show that the difference in means is not a coincidence.

8.2 Per-trajectory analysis

In this subsection we will interpret the P values of the statistical tests described in the previous subsection and we will evaluate the results of a trajectory overall.

Study room circle For the circle trajectory in the study room we see that every difference analyzed has a P value of lower than our alpha of 0.05. This means each of the mean values are significantly different and since the mean value of the baseline is lower than the mean value of Colmap and the mean value of the Fusion is the lowest of all we can conclude that the quality of the images improved in this trajectory. The circle was meant to evaluate all kind of situations and thus we can see that in this dataset the fusion algorithm shows an overall increase of synthesized image quality.

Study room distance For the distance trajectory we want to see what the effect of larger distances would be for the generation of the synthesized images with our fusion algorithm. Larger distances are a weakness of the depth camera and result in less precise depth data. In the results we see that the fusion algorithm outperforms both the baseline and the Colmap depth maps. The difference between the Colmap depth maps and the baseline is not significant (see table 6) however. This can be explained by the amount of noise the Colmap images introduce on textureless areas. The effect can be seen in figure 18 on the white wall. Overall there seems to be less noise in the synthesized images using the fusion algorithm, meaning that for this dataset the larger distances are handled well by our algorithm. A problem that we do notice however are the color differences on the white wall. This is taken into consideration in the DSQM scores, but is not caused by depth maps. The problem are the color images that captured the white wall in different shades of white. It could be interesting to see if that problem can be fixed in future work.

Study room textured The last trajectory in the study room analyzes the book cases and contains the most textures of all six trajectories. The first thing to notice is the immediate increase in quality for all three depth map sets regarding the mean score. This shows that Colmap performs a lot better if there are more textures present and explains the higher quality of the synthesized views (a lower score means a higher quality). Even though the Colmap depth maps perform relatively well initially, when comparing them to the baseline and fusion algorithm we can still see a significant increase in quality for the synthesized images. However, the difference between the baseline and fusion algorithm is not significant (table 6) this time. In figure 19 we show an example of the depth maps used in this trajectory. When taking a closer look, it can be seen that the baseline has a surprisingly low rate of

noise. The depth map generated by the fusion algorithm looks like it has a little less noise, but as shown the difference is not big enough to impact the quality of the resulting synthesized image. The reason for the worse performance of the Colmap depth map is the trajectory showing some of the wall like in the shown image, where the addition of the depth values of the camera is needed. Since there are no depth values initially for the textureless area we see that the baseline has a good initial depth map explaining why the difference between the fusion and the baseline is not big.

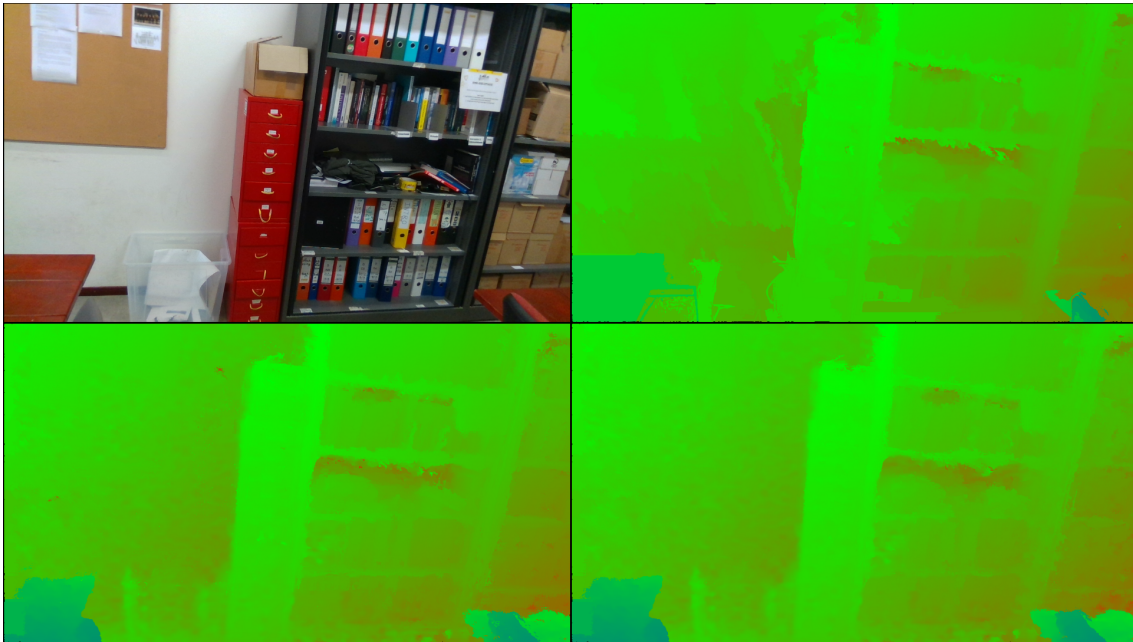
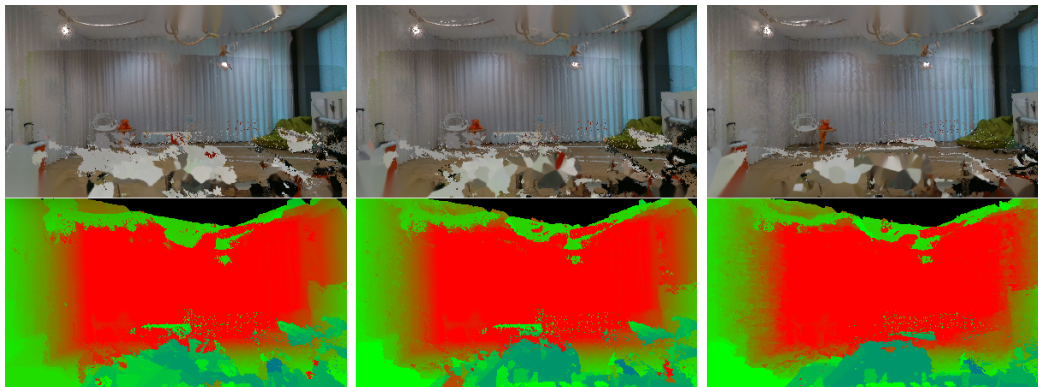


Figure 19: An example of the depth maps used in the textured trajectory in the study room. From top left to bottom right: color image, Colmap depth image, baseline depth map and fused depth map.

Study room conclusion Overall the quality of the synthesized images increased with our fusion algorithm over the Colmap depth maps as well as over the baseline. The only exception are areas with clear differences between textured and textureless areas, where the baseline can easily pick the important values like our fusion algorithm does. The fusion algorithm still outperforms the Colmap set and there is no difference in the case the baseline performs well. Therefore we conclude that the fusion algorithm does give us the best quality for synthesized images in the studyroom dataset.

ING circle The first thing to notice for this trajectory is that the quality is relatively low compared to the other trajectories in this dataset. Looking at figure 20 we can see that the DIBR algorithm has trouble reconstructing the tables very close to the Pepper robot. The likely explanation for this is that the algorithm is lacking the correct views to reconstruct the table or that the camera poses for the images showing the table are slightly different from their actual values. In the latter case this causes the most problems at close range, as the same point on the table will get a very different position in the synthesized image.



(a) Synthesized image with the Colmap depth maps (b) Synthesized image with the baseline depth maps (c) Synthesized image with the fused depth maps

Figure 20: Synthesized image from the circle trajectory in the ING room dataset. It can be seen that the table in the front does not have enough coverage to be synthesized correctly.

Regarding the differences in the mean values we can see that both the fusion and baseline outperform the Colmap depth map set with regards to quality. However, like in the textured trajectory of the study room, the difference between the baseline and the fusion algorithm is not significant. When looking at an example of the depth maps used in figure 21 we can see that even though the depth at close range seems a lot better, the depth further away seems to be better for the Colmap image, making the basic combination as good as the fusion algorithm. We could improve this by changing the gamma value in equation 2 such that distance is more important than the textureless areas. The distance is large but the back of the room is textureless so the values of the camera are still chosen even though they are far away and less precise. The problem with changing the gamma value would be that the other results might change as well, so we will provide a possible different solution later.

ING distance A more extreme example of what happened in the circle trajectory appears in the ING distance trajectory. Here the Colmap depth maps score best out of all three sets. See table 4 to observe that the Colmap scores are lowest. The significant difference means it scores best. The baseline does not score significantly worse, since it favours all Colmap values. In this case it is even more obvious that the quadratic falloff is hurting the quality of the captured depth map and the fusion algorithm does not handle this well enough to increase the quality of the synthesized image. We will discuss how to remedy this after taking a look at the last trajectory.

ING texture For the texture trajectory of the ING room we see that the Colmap is outperformed by both the baseline and the fusion algorithm and that the fusion algorithm scores significantly better than the baseline. The results here are similar to the study room trajectories and here the fusion algorithm outperforms the baseline as opposed to the study room textured trajectory. The reason for this is that there are less textures in this trajectory than in the study room circle trajectory. This is because the ING room does not have an area with as much texture as the bookcase in the study room, so the initial Colmap results are a little worse. This means that the fusion algorithm can do more to properly remove noise and to pick the correct

values.

ING dataset conclusion For the ING dataset we see that overall the fusion algorithm does not work very well. The main reasons for this are the larger distances in this dataset and the increased number of images helped Colmap to make a better initial reconstruction. To be able to better handle the larger distances in a larger room we can make two different kind of recordings. For this research we used high density profile in the camera settings. We can change the settings to better record the very large distances. This means that the minimal measured distance becomes larger, which is why we did not do this initially. There would be two different depth map sets with two different values for $minZ$ and $maxZ$ in equation 3. This would require the two sets to undergo the fusion algorithm with different parameters. We assume that we could get good results on rooms like the ING one as well if we do this. Another option is to increase the gamma value in equation 2 to make sure that values with a large distance are less likely to be chosen. This will however influence the other results, which is why we do not recommend this approach.

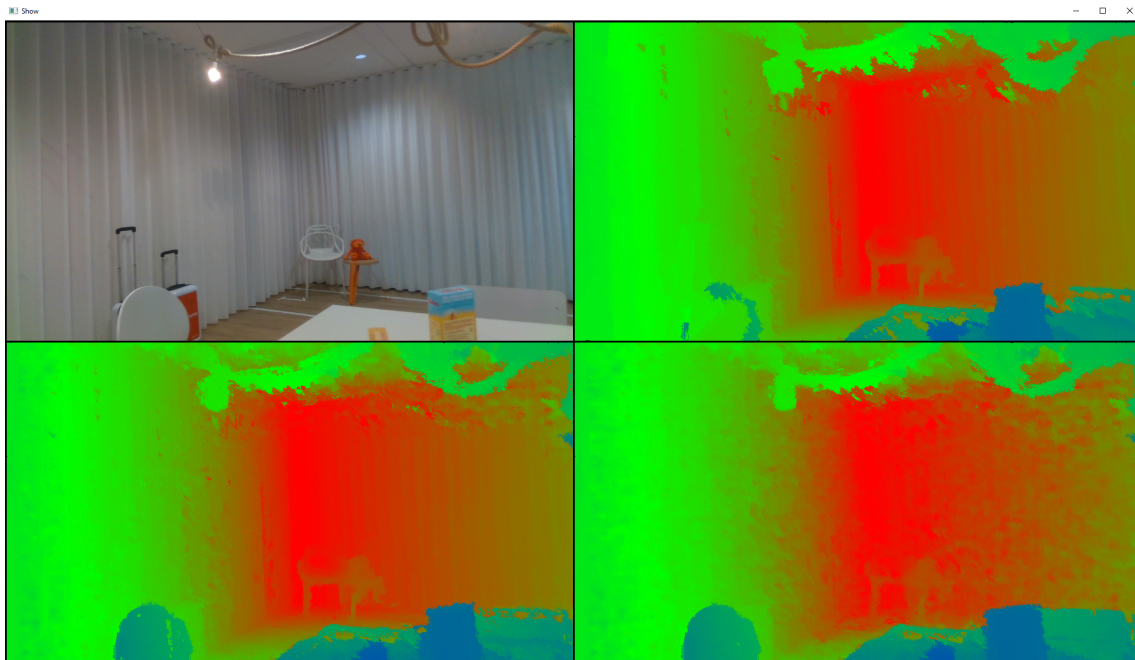


Figure 21: An example of the depth maps used in the circle trajectory in the ING room. From top left to bottom right: color image, Colmap depth image, baseline depth map and fused depth map.

9 Conclusion & future work

In this section we will discuss the conclusion of this research and potential future work that could improve the recordings, reconstruction, DIBR results and the fusion algorithm.

9.1 Conclusion

For this research we first conducted a pilot to see the current status of the simulation and the results of the DIBR algorithm. We concluded that the simulation and synthesized images did not have the desired quality to continue developing object detection and navigation algorithms. For that reason we decided to create our own dataset and we wanted to improve the quality of the reconstruction. That led to our two objectives:

- **O1:** Create an RGB-D dataset that maximizes the initial 3D reconstruction quality with the Colmap software, as well as aligned depth maps to utilize in the depth fusion.
- **O2:** Improve the quality of the synthesized views by using a fusion of the depth maps generated by Colmap and captured depth data as input for DIBR.

O1 To complete the first objective we performed several recordings in the study room to see which light conditions work best for Colmap and for our camera. After reconstructing the table in the room and analyzing the point cloud it was clear we needed to block as much sunlight as possible and had to work with the artificial lights in the room. Even though this casts a lot of shadows it seemed to help the Colmap software to produce the best reconstruction. After experimenting with some different Colmap parameters we reached the conclusion that the parameters found in appendix A provided a decent initial reconstruction. We used the same settings and method for a second recording in a room at ING where we selected more initial images to get a better coverage for the DIBR. This reconstruction looked better and we can conclude that we succeeded in producing better initial reconstructions for our own dataset. We used the RealSense SDK to align the depth images to the color images real time. Even though the alignment is not pixel perfect we were able to use the depth images for our second objective and therefore we conclude that we successfully completed the first objective.

O2 For our second objective we fused the depth maps captured by our depth camera with the synthesized depth maps created by Colmap. We created a fusion algorithm that combined the strengths of both depth maps and determined pixel wise which depth value was most likely to be correct. In order to evaluate our results we implemented the DSQM algorithm as an IQA for synthesized views, as we want to know whether the fused depth maps could improve the quality of the synthesized views. We then generated three trajectories in both datasets created for objective 1 to analyze the effectiveness of the fusion algorithm. We showed that for a smaller room with relatively many textures the fusion algorithm does improve the results of the synthesized views. As for the ING dataset we saw a drop-off in the quality when the distance gets too large. The quality of the depth maps of the camera can

no longer remedy the wrong values in the Colmap depth map. However at close range the fusion algorithm still outperformed the baseline and the Colmap depth maps. We hypothesize that if a separate recording is made that focuses more on capturing large distances we can get captured depth maps that would help improve the quality of the synthesized views for large distances as well. Overall we conclude that our fusion algorithm does improve the quality of the synthesized views. When the algorithm is used it is important to think about the best way the depth maps are captured and adjust the parameters according to the data captured.

9.2 Future work

We were able to increase the quality of the simulation, but there are still some things that could be improved. One of the main problems with the synthesized images with DIBR is the noise present in textureless areas. This can best be seen in figure 17. The problem is caused by color images projected into the synthesized views. The color camera of our chosen capture device performs relatively poorly so you can see the color differences when the same white areas are captured. A better color camera could remedy this, but would pose a new problem with aligning the depth maps and color images. A setup with a better color camera that captures less different colors mounted with the depth camera could solve the problem. It would be interesting to see whether we could align the depth maps with those color images correctly.

Alignment For the alignment of the color and depth maps we used the SDK, but we noticed that the alignment is not pixel perfect. We did use the global refinement to alleviate part of the problem, but it still introduces some noise. We would recommend to improve the alignment when recording or do some offline pre-processing. Another option would be to look at the global refinement and see if we can solve the problem from that perspective.

Ground truth Another thing we looked at is to compare the resulting depth maps to their ground truth. We focused on improving the quality of the synthesized image rather than looking at the ground truth depth values for the depth maps. It would be interesting to see how much closer to the ground truth our depth maps are and if possible compare it to some other depth fusion method. If we look at results shown in section 8 we assume that the fused depth maps would prove better than the depth maps created by Colmap and the baseline if compared to a ground truth. It will be interesting to see if we could compare it to some state of the art global fusion algorithms to evaluate the effectiveness of our fusion algorithm.

Quality map We also created a way to assess the quality of the synthesized images. In section 2 we mentioned a quality map. With the quality assessment we can make such a map. A useful addition would be to include the coverage in the quality map, as we saw in both our datasets that even though there are a lot of images, getting full coverage of the entire scene is nearly impossible. So generating a quality map that can show the expected quality and coverage would be a valuable addition to the simulation to assist in real world and algorithmic tasks.

Object detection and navigation We worked on the simulation to create a testing environment for object detection and navigation algorithms. It will be interesting to see how much the simulation contributes to these tasks. This would also mean

adding annotations to our dataset so that object detection algorithms can be tested.

Real world We also mentioned we want robots like Pepper to use the environment built in the simulation to assist in their real world tasks. This will be a difficult task, as we noticed that even the static environments we record change a lot by moving the small objects. That does not yet include dynamic movement of persons present in the room. It would be an interesting study to see how the simulation can be used to assist in real world tasks such as object detection and navigation and how we could deal with these problems.

Scenes An addition to these real world tasks is the preparation for a real world environment, to ensure that more texture is present. We did not do any preparation in the study room as it had plenty of textures. However, most rooms do not have this much texture, which is why we had to add several objects to the second room. For robots like Pepper to be able to perform the tasks in the real world a solution for this problem should be found. Our fusion algorithm is the first step, but it still needs a good initial reconstruction.

Conclusion In conclusion, even though we were able to improve the simulation, some work can still be done to further enhance the simulation. A lot of options are now open for the use of the simulation and it will be interesting to see where this will get us.

A Colmap parameters

```

[ImageReader]
single_camera          true
single_camera_per_folder  false
existing_camera_id     -1
default_focal_length_factor  1.2
camera_model          SIMPLE_RADIAL
[SiftExtraction]
use_gpu                true
estimate_affine_shape  false
upright                false
domain_size_pooling   false
num_threads            -1
max_image_size         3200
max_num_features        8192
first_octave            -1
num_octaves             4
octave_resolution       3
max_num_orientations    2
dsp_num_scales          10
peak_threshold          0.00666666666666666671
edge_threshold          10
dsp_min_scale           0.166666666666666666
dsp_max_scale           3
gpu_index               -1
[SiftMatching]
use_gpu                true
cross_check             true
multiple_models         false
guided_matching         false
num_threads            -1
max_num_matches         32768
max_num_trials          10000
min_num_inliers         15
max_ratio                0.800000000000000004
max_distance            0.699999999999999996
max_error                4
confidence              0.999
min_inlier_ratio        0.25
gpu_index               -1

```

[SequentialMatching]	
quadratic_overlap	true
loop_detection	false
overlap	10
loop_detection_period	10
loop_detection_num_images	50
loop_detection_num_nearest_neighbors	1
loop_detection_num_checks	256
loop_detection_num_images_after_verification	0
loop_detection_max_num_features	-1
[SpatialMatching]	
is_gps	true
ignore_z	true
max_num_neighbors	50
max_distance	100
[BundleAdjustment]	
refine_focal_length	true
refine_principal_point	false
refine_extra_params	true
refine_extrinsics	true
max_num_iterations	100
max_linear_solver_iterations	200
function_tolerance	0
gradient_tolerance	0
parameter_tolerance	0
[Mapper]	
ignore_watermarks	false
multiple_models	true
extract_colors	true
ba_refine_focal_length	true
ba_refine_principal_point	true
ba_refine_extra_params	true
ba_global_use_pba	true
tri_ignore_two_view_tracks	true
min_num_matches	15
max_num_models	50
max_model_overlap	20
min_model_size	10
init_image_id1	-1
init_image_id2	-1
init_num_trials	200
num_threads	-1
ba_local_num_images	6
ba_local_max_num_iterations	25
ba_global_pba_gpu_index	-1
ba_global_images_freq	500
ba_global_points_freq	250000

ba_global_max_num_iterations	50
ba_global_max_refinements	5
ba_local_max_refinements	2
snapshot_images_freq	0
init_min_num_inliers	100
init_max_reg_trials	2
abs_pose_min_num_inliers	30
max_reg_trials	3
tri_max_transitivity	1
tri_complete_max_transitivity	5
tri_re_max_trials	1
min_focal_length_ratio	0.10000000000000001
max_focal_length_ratio	10
max_extra_param	1
ba_global_images_ratio	1.1000000000000001
ba_global_points_ratio	1.1000000000000001
ba_global_max_refinement_change	0.0005000000000000001
ba_local_max_refinement_change	0.001
init_max_error	4
init_max_forward_motion	0.9499999999999996
init_min_tri_angle	16
abs_pose_max_error	12
abs_pose_min_inlier_ratio	0.25
filter_max_reproj_error	4
filter_min_tri_angle	1.5
tri_create_max_angle_error	2
tri_continue_max_angle_error	2
tri_merge_max_reproj_error	4
tri_complete_max_reproj_error	4
tri_re_max_angle_error	5
tri_re_min_ratio	0.20000000000000001
tri_min_angle	1.5
[PatchMatchStereo]	
geom_consistency	true
filter	true
write_consistency_graph	false
max_image_size	2000
window_radius	7
window_step	1
num_samples	15
num_iterations	5
filter_min_num_consistent	2
depth_min	-1
depth_max	-1
sigma_spatial	-1
sigma_color	0.20000000298023224
ncc_sigma	0.60000002384185791
min_triangulation_angle	1
incident_angle_sigma	0.89999997615814209

geom_consistency_regularizer	0.30000001192092896
geom_consistency_max_cost	3
filter_min_ncc	0.10000000149011612
filter_min_triangulation_angle	3
filter_geom_consistency_max_cost	1
cache_size	32
gpu_index	-1
[Render]	
adapt_refresh_rate	true
image_connections	false
min_track_len	3
refresh_rate	1
projection_type	0
max_error	2
[ExhaustiveMatching]	
block_size	50
[StereoFusion]	
max_image_size	-1
min_num_pixels	25
max_num_pixels	10000
max_traversal_depth	100
check_num_images	50
max_reproj_error	2
max_depth_error	0.0099999997764825821
max_normal_error	10
cache_size	32

References

- [AZ18] Gianluca Agresti and Pietro Zanuttigh. “Combination of spatially-modulated ToF and structured light for MPI-free depth estimation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 0–0.
- [Arm+17] Iro Armeni et al. “Joint 2d-3d-semantic data for indoor scene understanding”. In: *arXiv preprint arXiv:1702.01105* (2017).
- [Bos+11] Emilie Bosc et al. “Towards a new quality metric for 3-D synthesized view assessment”. In: *IEEE Journal of Selected Topics in Signal Processing* 5.7 (2011), pp. 1332–1343.
- [Cha13] Damon M Chandler. “Seven challenges in image quality assessment: past, present, and future research”. In: *ISRN Signal Processing 2013* (2013).
- [Cha+17] Angel Chang et al. “Matterport3d: Learning from rgb-d data in indoor environments”. In: *arXiv preprint arXiv:1709.06158* (2017).
- [CM92] Yang Chen and Gérard Medioni. “Object modelling by registration of multiple range images”. In: *Image and vision computing* 10.3 (1992), pp. 145–155.
- [CZK15] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. “Robust reconstruction of indoor scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5556–5565.
- [Cre+07] Frederique Crete et al. “The blur effect: perception and estimation with a new no-reference perceptual blur metric”. In: *Human vision and electronic imaging XII*. Vol. 6492. International Society for Optics and Photonics. 2007, p. 64920I.
- [Dai+17] Angela Dai et al. “Scannet: Richly-annotated 3d reconstructions of indoor scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5828–5839.
- [Dal+12] Carlo Dal Mutto et al. “Locally consistent tof and stereo data fusion”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 598–607.
- [FLG17] Muhammad Shahid Farid, Maurizio Lucenteforte, and Marco Grangetto. “Perceptual quality assessment of 3D synthesized images”. In: *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2017, pp. 505–510.
- [FB81] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: <http://doi.acm.org/10.1145/358669.358692>.

- [F+15] Yasutaka Furukawa, Carlos Hernández, et al. “Multi-view stereo: A tutorial”. In: *Foundations and Trends® in Computer Graphics and Vision* 9.1-2 (2015), pp. 1–148.
- [GLS15] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. “Massively parallel multiview stereopsis by surface normal diffusion”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 873–881.
- [GVS18a] Silvio Giancola, Matteo Valenti, and Remo Sala. “Metrological Qualification of the Intel D400™ Active Stereoscopic Cameras”. In: *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopic Technologies*. Springer, 2018, pp. 71–85.
- [GVS18b] Silvio Giancola, Matteo Valenti, and Remo Sala. “State-of-the-Art Devices Comparison”. In: *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopic Technologies*. Cham: Springer International Publishing, 2018, pp. 29–39.
- [Gro18] Robert Groot. “Autonomous Exploration and Navigation with the Pepper robot”. MA thesis. University Utrecht, Aug. 2018.
- [Hed+16] Peter Hedman et al. “Scalable inside-out image-based rendering”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 231.
- [Hen+12] Peter Henry et al. “RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments”. In: *The International Journal of Robotics Research* 31.5 (2012), pp. 647–663.
- [Hua+16] Binh-Son Hua et al. “Scenenn: A scene meshes dataset with annotations”. In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 92–101.
- [Int19a] Intel, ed. *Best-Known-Methods for Tuning Intel® RealSense™ D400 Depth Cameras for Best Performance*. https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/BKMs_Tuning_RealSense_D4xx_Cam.pdf. Sept. 2019.
- [Int19b] Intel, ed. *Intel® RealSense™ Depth Camera D400-Series*. <https://software.intel.com/en-us/realsense/d400>. May 2019.
- [KH13] Michael Kazhdan and Hugues Hoppe. “Screened poisson surface reconstruction”. In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), p. 29.
- [Kov99] Peter Kovesi. “Image Features from Phase Congruency”. In: *ISRN Signal Processing* vol. 1, no. 3 (1999), pp. 1–26.
- [Lai+11] Kevin Lai et al. “A large-scale hierarchical multi-view rgb-d object dataset”. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 1817–1824.
- [MZM16] Giulio Marin, Pietro Zanuttigh, and Stefano Mattoccia. “Reliable fusion of tof and stereo depth driven by confidence measures”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 386–401.

- [Mat09] Stefano Mattoccia. “A locally global approach to stereo correspondence”. In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE. 2009, pp. 1763–1770.
- [MMM12] Pierre Moulon, Pascal Monasse, and Renaud Marlet. “Adaptive structure from motion with a contrario model estimation”. In: *Asian Conference on Computer Vision*. Springer. 2012, pp. 257–270.
- [Nai+13] Rahul Nair et al. “A survey on time-of-flight stereo fusion”. In: *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 2013, pp. 105–127.
- [New+11] Richard A Newcombe et al. “Kinectfusion: Real-time dense surface mapping and tracking.” In: *ISMAR*. Vol. 11. 2011, pp. 127–136.
- [O M+19] Niall O’ Mahony et al. “Computer Vision for 3D Perception”. In: *Intelligent Systems and Applications*. Ed. by Kohei Arai, Supriya Kapoor, and Rahul Bhatia. Cham: Springer International Publishing, 2019, pp. 788–804. ISBN: 978-3-030-01057-7.
- [Sch+14] Daniel Scharstein et al. “High-resolution stereo datasets with subpixel-accurate ground truth”. In: *German conference on pattern recognition*. Springer. 2014, pp. 31–42.
- [Sei+06] Steven M Seitz et al. “A comparison and evaluation of multi-view stereo reconstruction algorithms”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 1. IEEE. 2006, pp. 519–528.
- [SK00] Harry Shum and Sing Bing Kang. “Review of image-based rendering techniques”. In: *Visual Communications and Image Processing 2000*. Vol. 4067. International Society for Optics and Photonics. 2000, pp. 2–14.
- [SF11] Nathan Silberman and Rob Fergus. “Indoor scene segmentation using a structured light sensor”. In: *2011 IEEE international conference on computer vision workshops (ICCV workshops)*. IEEE. 2011, pp. 601–608.
- [Sil+12] Nathan Silberman et al. “Indoor segmentation and support inference from rgb-d images”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 746–760.
- [SLX15] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576.
- [Vil+17] Víctor Villena-Martínez et al. “A quantitative comparison of calibration methods for RGB-D sensors using different technologies”. In: *Sensors* 17.2 (2017), p. 243.

-
- [XOT13] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. “Sun3d: A database of big spaces reconstructed using sfm and object labels”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1625–1632.
- [Zhu+10] Jiejie Zhu et al. “Reliability fusion of time-of-flight depth and stereo geometry for high quality depth maps”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.7 (2010), pp. 1400–1414.