Journal on Interactive Systems, 2021, 12:1, doi: 10.5753/jis.2021.2065

Fast and Robust 3D Reconstruction Solution from Permissive Open-Source Code

Victor Gouveia de M. Lyra [**Voxar Labs, Centro de Informática (UFPE)** | ***vgml@cin.ufpe.br***]
Adam H. M. Pinto [**Voxar Labs, Centro de Informática (UFPE)** | ***ahmp3@cin.ufpe.br***]
Gustavo C. R. Lima [**Voxar Labs, Centro de Informática (UFPE)** | ***gcrl@cin.ufpe.br***]
João Paulo Lima [**Departamento de Computação (UFRPE) and Voxar Labs, Centro de Informática (UFPE)** | ***joao.mlima@ufrpe.br***, ***jpsml@cin.ufpe.br***]
Veronica Teichrieb [**Voxar Labs, Centro de Informática (UFPE)** | ***vt@cin.ufpe.br***]
Jonysberg Peixoto Quintino [**Projeto de P&D CIn/Samsung (UFPE)** | ***jpq@cin.ufpe.br***]
Fabio Q. B. da Silva [**Centro de Informática (UFPE)** | ***fabio@cin.ufpe.br***]
André L M Santos [**Centro de Informática (UFPE)** | ***alms@cin.ufpe.br***]
Helder Pinho [**SiDi, Campinas** | ***helder.p@sidi.org.br***]

Abstract

With the growth of access to faster computers and more powerful cameras, the 3D reconstruction of objects has become one of the public's main topics of research and demand. This task is vigorously applied in creating virtual environments, creating object models, and other activities. One of the techniques for obtaining 3D features is photogrammetry, mapping objects and scenarios using only images. However, this process is very costly and can be time-consuming for large datasets. This paper proposes a robust, efficient reconstruction pipeline with a low runtime in batch processing and permissive code. It is even possible to commercialize it without the need to keep the code open. We mix an improved structure from motion algorithm and a recurrent multi-view stereo reconstruction. We also use the Point Cloud Library for normal estimation, surface reconstruction, and texture mapping. We compare our results with state-of-the-art techniques using benchmarks and our datasets. The results showed a decrease of 69.4% in the average execution time, with high quality but a greater need for more images to achieve complete reconstruction.

Keywords: *reconstruction, photogrammetry, permissive license, batch, texture*

1 Introduction

The creation of 3D assests is one of the main challenges in Virtual Reality (VR) and Augmented Reality (AR). These can be done by photogrammetry, mapping objetcs and scenarios using only images or video frames. This research field was already a trending topic in Seitz et al. (2006), where the authors evaluate multi-view stereo (MVS) reconstruction algorithms. The MVS is responsible for receiving the camera parameters and the image data, and match view poitns and keypoints, obtaining dense 3D correspondences. One example of MVS use is Schops et al. (2017).

For the task of 3D reconstruction through images, they are usually made from several available technologies. This combination consists of Structure for Motion (SfM) (Ullman, 1979), the aforementioned MVS, and a mesh and texturing stage. SfM allows the mapping of previously unknown environments and selects the pose information from the camera. Is in the final mesh step where the algorithm makes point cloud triangulation and gives texture to the reconstruction.

One of the main problems in 3D reconstruction is the efficiency, *i.e.*, how to obtain consistent results while keeping a low processing time. Also, the increasingly easy access to powerful devices and high resolution images, a best cost-benefit is essential to make this technology accessible to general use. Another photogrammetry problem is that many disponible solution do not have permissive licenses, with make impossible the commercial use and distribution. As already said, 3D reconstruction demands lots of different algorithms that may have different license. To be able to use it commercially and without the need to make the code open-source, which may be not desirable, it is possible to use only techniques whose source code has permissive license.

In this paper, we present the results of a photogrammetry pipeline with batch processing. We focused on getting state-of-the-art comparable results using only open-source techniques with permissive license. Our pipeline is based on using a modified COLMAP SfM for geometric features detection and applying the R-MVSNet approach allied to Point Cloud Library (PCL) to reconstruct the scene, those will be described properly in the section 3, being able to keep low execution time, making this technology more accessible for daily use. The contributions of this work are the following:

- A 3D reconstruction solution that uses only permissively licensed techniques, being suitable to commercial use;
- A pipeline using the R-MVSNet method, which achieved a good cost-benefit in terms of quality of results and runtime compared to other state-of-the-art solutions;
- Qualitative and quantitative evaluations regarding 3D reconstruction accuracy and execution time of the developed method with respect to state-of-the-art 3D reconstruction techniques;
- Experiments using real-world daily use scenarios, re-

inforcing our proposal viability, and comparison using banchmark datasets.

This paper is organized as follows: Section 2 presents the related works; Section 3 describes the method employed in this study; Section 4 presents our methodology; Section 5 discusses the results; and Section 6 concludes the paper.

2 Related Works

One of the fundamental steps (and research field) in computer vision and graphics is acquiring 3D information. With their changing parameters and colors, dynamic environments make the creation of efficient and robust models a complicated task (Schönberger et al., 2016). This task is even more complex using only permissive solutions. In this work, we combine pipelines from different applications to create a viable solution with high quality and the lowest computational cost.

Multi-View Reconstruction Environment (MVE) (Fuhrmann et al., 2014) is an end-to-end free software for multi-view geometry reconstruction. This system has inputs with several photos and uses the well-known three main reconstruction steps: SfM (Seitz et al., 2006) to reconstruct the camera parameters (intrinsic and extrinsic), MVS to obtain 3D correspondences and mesh generation, combining the dense point cloud with color information, rendering a final colored object. The SfM step uses both SIFT (Lowe, 2004) and SURF (Bay et al., 2008) for feature detection, and these features are matched between a pair of images. Every image in the dataset is matched to all other photos.

Memory consumption is one of the main limitations of MVE, even using low-resolution features to discard unmatched image pairs before performing the full-resolution match. For surface reconstruction, all the points are kept in memory, which is prohibitive for large-scale scenes or a large datasets, creating a bottleneck in execution time performance. For example, it takes 116 minutes to perform reconstruction using the 79-images Der Hass dataset (Fuhrmann et al., 2014) [1]. To reach a reasonable time performance in these large datasets, the MVS method proposed by Goesele et al. (2007) is a solution. As a depth map-based approach, there is a lot of redundancy, but it also means that only a small set of neighboring views are required for reconstruction. The Floating Scale Surface Reconstruction (FSSR) approach (Fuhrmann and Goesele, 2014) is used to perform point-based mesh reconstruction.

Using an uncontrolled environment, as random internet images or user made datasets, is an even more challenging task. In these cases, other variables influence the final result, such as variability in resolution, changes in lighting, occlusions, and increased noise. In general, these datasets generate sparse clouds, unable to be used to create the 3D model. To improve these point clouds, Schönberger et al. (2016) proposed COLMAP, a general-purpose SfM (Schönberger and Frahm, 2016) and MVS pipeline, offering a range of features for reconstruction. Starting from the optimization framework proposed by Zheng et al. (2014), COLMAP improves the PatchMatch sampling scheme, applying a pixel-wise normal estimation, introducing a multi-view geometric consistency term and a "temporal" view selection smoothness term. The inclusion of normal estimation and geometric prior allied to the optimization framework and a novel likelihood function makes the solution less memory expensive, as opposed to the Markov Random Field approach in Strecha et al. (2004). This approach was compared to state-of-the-art algorithms in low and high-resolution datasets, obtaining competitive results, and sometimes outperforming the previous results.

In terms of completeness, a few state-of-the-art implementations are performing well, since problems such as low-textured or reflexive regions make incomplete reconstructions using the dense correspondences. However, with the advent of deep learning, the use of this model for application in stereo reconstruction has grown. In Yao et al. (2018), MSVNet is proposed, decreasing memory consumption by building information from camera frustum and decoupling the MVS construction into smaller problems. This method outperformed the previous state-of-the-art, being faster in execution time. Then, in Yao et al. (2019), a scalable MVS framework called R-MVSNet is proposed based on recurrent neural networks. It improves the MVSNet implementation introducing recurrent regularization using a convolutional gated recurrent unit (GRU) (Cho et al., 2014). R-MVSNet is used to estimate the reference depth map in our approach.

After the R-MVSNet, several algorithms were proposed to enhance the results and made better 3D models. One of these projects is Point-MVSNet (Chen et al., 2019), a deep framework that generates a coarse depth map and converts it into a point cloud, predicting the deep in a coarse-to-fine manner. This approach guarantees a more efficient representation of the target scene, refining the point cloud interactively without converting to volumetric grids. The Point-MVSNet was able to produce a high-quality and relatively fast reconstruction, even without decreasing image resolution. After, Chen et al. (2020) proposed the Visibility-Aware (VAPoint-MVSNet) extends the Point-MVSNet with visibility-aware multi-view feature aggregations, which aggregates information to a better result with occlusions.

Another solution is the PA-MVSNet (Zhang et al., 2021), which uses a pyramidal attention module, obtaining more information from the original image and generating a significant improvement in the representation of features. This solution has improved image quality and less noise, while it has an increase the execution time for the generation of multi pyramidal views. Overall, this system was relatively better when compared to other methods. Focusing on decreasing memory requirements to process high-resolution images, the HighRes-MVSNet (Weilharter and Fraundorfer, 2021) uses a pyramid encoder-decoder structure align to a coarse-to-fine hierarchy, achieving to find depth correspondences. This approach significantly reduced the CPU and runtime requirements on challenging benchmarks, but had a intermediate result in terms of accuracy and completeness.

[1] Avaliable in https://www.gcc.tu-darmstadt.de/home/proj/mve/index.en.jsp

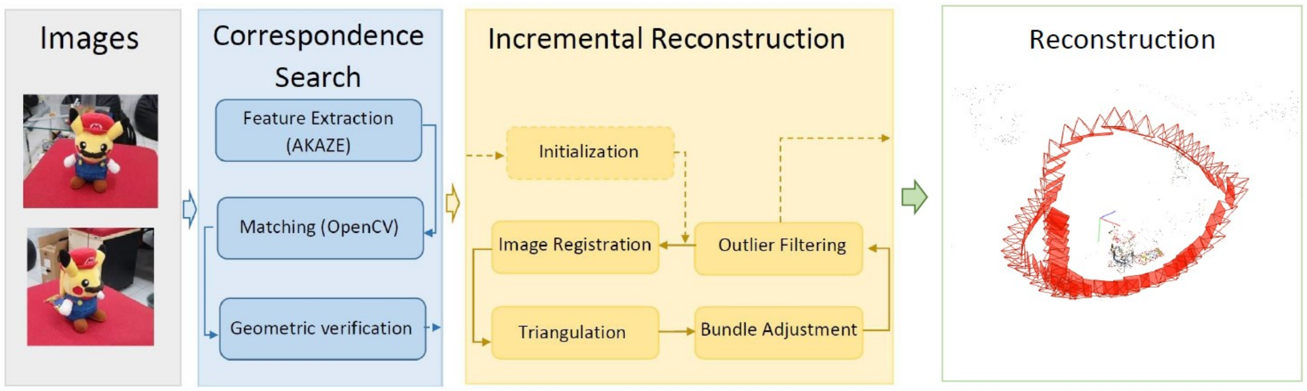

Figure 1. The modified (including permissive AKAZE and OpenCV) COLMAP SfM pipeline used. Adapted from Fuhrmann et al. (2014).

3 Methods

As mentioned in Section 2, uncontrolled environments are a challenging task in reconstruction. Most of the state-of-the-art algorithms fail to perform complete large-scale scene reconstructions from Internet images. To maximize the robustness, completeness, and accuracy of our solution and to keep the time consumption low, we use a modified version of the SfM method proposed in Schönberger and Frahm (2016), the depth estimation of Yao et al. (2019) and PCL Rusu and Cousins (2011) for meshing and texturing.

3.1 COLMAP Structure from Motion

Although COLMAP has a permissive license, some dependencies such as QT, FreeImage, and SIFT Lowe (2004) cannot be used commercially. To guarantee the use of COLMAP SfM in our pipeline, we had to remove the QT Library, replace FreeImage by OpenCV Bradski (2000) and SIFT by A-KAZE Alcantarilla and Solutions (2011). The SfM algorithm works as follows: considering the I_i input image in $\mathcal{I} = \{I_i \mid i = 1 \cdots N_I\}$, we want to detect sets of local features $\mathcal{F}_i = \{(x_j, f_j) \mid j = 1 \cdots N_{F_i}\}$ at location $x_j \in R$ represented by f_j descriptor. To be recognized in multiple images, these features have to be geometric invariant, so A-KAZE is a good descriptor in terms of robustness. Using these F_i features, SfM can identify the images that catch sight of the same scene. This matching has to be scalable and efficient, as the brute force approach is prohibitive for extensive image collections. The output is the overlapping image pairs $\mathcal{C} = \{\{I_a, I_b\} \mid I_a, I_b \in \mathcal{I}, a < b\}$.

With this $\mathcal{C}$ set, it is necessary to verify the geometric relation, as the matching in the previous step is only based on appearance. This process starts with the estimation of the fundamental matrix (for uncalibrated images) and essential matrix (for calibrated images), finding the number of inliers, and then the homography inliers. If it is found that $N_{inliers} > th$, being th a threshold, the image is geometrically verified. By triangulating points from essential matrix decomposition, it is possible to find the triangulation angle α_m, which is used to distinguish pure rotation and planar scenes. This process, allied to a similarity transformation check to remove common Internet photos problems (watermarks, timestamps, and frames), enables an optimal initialization for a robust reconstruction.

For the reconstruction step, the inputs are scene graphs, the outputs are estimated poses $\mathcal{P} = \{P_c \in SE(3) \mid c = 1, \cdots, N_p\}$, and the scene is reconstructed as a set of points $\mathcal{X} = \{X_k \in R^3 \mid k = 1, \cdots, N_X\}$. The model is initialized from dense image graph location, as the overlapped cameras can result in more accurate reconstruction. The intrinsic parameters and the P_c pose are estimated by solving the Perspective-n-Point (PnP) problem, based on already found 2D-3D correspondences. As these correspondences are typically contaminated by outliers, Schönberger and Frahm (2016) proposed a better image selection, enhancing the uncertainty-driven solution of Haner and Heyden (2012) and giving a score considering how visible and uniform the points distribution is. After the triangulation, a step that is crucial and high computational, the reconstructed scene can be added to $\mathcal{X}$. The recursive RANSAC multi-view triangulation handles the outlier contamination, reducing the cost of this step. Figure 1 shows the SfM pipeline used.

3.2 R-MVSNet

MVSNet (Yao et al., 2018) (Figure 2) showed state-of-the-art results on the DTU datasetJensen et al. (2014)[2], but it is prohibitive for large-scale scenes. To solve this problem, Yao et al. (2019) proposed a recurrent MVSNet implementation. This approach starts with a recurrent regularization scheme with sequential processing of a C cost volume, based on GRU. C is viewed as D cost maps $\{C(i)\}_{i=1}^{D}$ concatenated in depth direction. That means the $C_r(t)$ step is dependent on cost maps of $C(t)$ as well as all previous $\{C(i)\}_{i=1}^{t-1}$ steps. This temporal context information is modeled as a GRU convolutional variant, formulated as

$$C_r(t) = (1 - U(t)) \odot C_r(t-1) + U(t) \odot C_u(t), \quad (1)$$

being $\odot$ the element-wise multiplication, $U(t)$ the update gate map to decide, and $C_r(t-1)$ the regularized cost map of late step. $C_u(t)$ could be viewed as the updated cost map in current step, defined as

$$C_u(t) = \sigma_c(W_c * [C(t), R(t) \odot C_r(t-1)] + b_c), \quad (2)$$

where $*$ is the convolution operation, $R(t)$ is the reset gate map deciding how much the previous $C_r(t-1)$ affects the

[2]http://roboimagedata.compute.dtu.dk/?page_id=36

current update, $\sigma(\cdot)$ is the sigmoid nonlinear mapping and $[]$ is the concatenation. The last output and current input feeds both update and reset gates. Being W and b the learning rates and σ_g the hyberbolic tangent function, the gates formula are

$$R(t) = \sigma_g(W_r * [C(t), C_r(t-1)] + b_r), \tag{3}$$

$$U(t) = \sigma_g(W_u * [C(t), C_r(t-1)] + b_u). \tag{4}$$

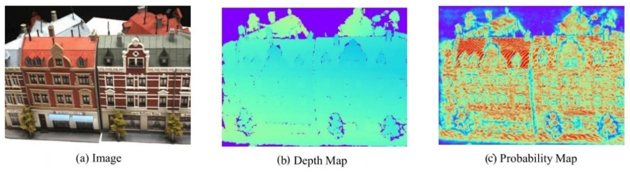

Figure 2. MVS reconstruction steps: (a) The reference image. (b) Final depth map. (c) Probability estimation for the obtained depth map. Adapted from Yao et al. (2019).

This model uses a singular layer GRU model that can be stacked to create deeper networks (Figure 3). The base pipeline starts with a 2D convolutional layer to map the C cost map, which is an input to the GRU layer. Next, a softmax layer generates the probability volume P from the regularized maps to calculate the training loss. The R-MVSNet model applies the inverse depth, treating the problem as a multi-class classification, with cross-entropy loss:

$$Loss = \sum_P \left(\sum_{i=1}^{D} -P(i,p) \cdot \log Q(i,p) \right), \tag{5}$$

where p is the spatial image coordinate, and $P(i,p)$ is a voxel in the probability volume P. Q is the ground truth binary occupancy volume generated by the one-hot encoding of the ground truth depth map. $Q(i,p)$ is the voxel corresponding to $P(i,p)$.

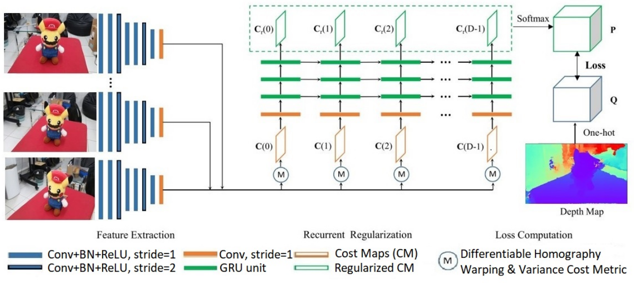

Figure 3. The R-MVSNet architecture. Features are extracted from input images and the cost maps are obtained at different depths, being then regularized by the convolutional GRU. Adapted from Yao et al. (2019).

3.3 PCL Library

PCL is a standalone, open-source library for point cloud processing tasks and geometric processing written in C++. It contains algorithms for filtering, feature extraction, segmentation, surface estimation, object recognition, visualization, among others. We selected some of these techniques to complement the proposed reconstruction. For more details on the algorithms check Rusu (2011).

3.3.1 Filtering

During the point cloud estimation, measurement errors can lead to sparse outliers, causing corruption errors and erroneous reconstruction. Statistical analysis can be performed to solve some of these irregularities, as the distribution of point to input neighbors distance. That can be done assuming a Gaussian distribution of the mean of all point distances to neighbors, and applying the mean and standard deviation to remove all data of this interval. Also, we downsampled the point cloud using the Voxel Grid filter algorithm, which can be seen as a grid of small 3D boxes in space, where all points inside these boxes are approximated to their centroid, taking a spatial average of the points in each voxel. Figure 4 shows an example of a dense point cloud passing through these filters.

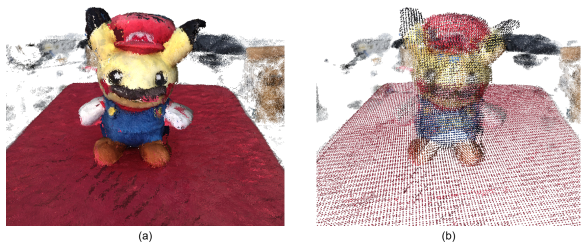

Figure 4. Outlier removal and point density reduction in the Pikachu dataset using PCL. (a) Dense point cloud; (b) Point cloud after filtering.

3.3.2 Resampling

Resampling algorithms can be used in case of occlusion or noisy data, when is impossible to do an additional scan to recreate missing parts of the surface. The Moving Least Squares (MLS) method was used to reconstruct the surface from the estimated set of points and also to remove some small artifacts, *i.e.,* double walls. MLS provides an interpolated surface for this set, fitting higher order polynomials to each point. This method was selected because it has the advantage that the resultant fitted surface passes through the original data points. Figure 5 shows an example of a point cloud before and after resampling.

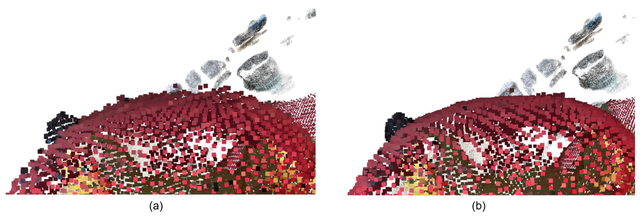

Figure 5. Smoothing of the Pikachu dataset point cloud using the MLS algorithm of PCL. (a) Point cloud with noise; (b) Smoothed point cloud after resampling.

3.3.3 Normal estimation

One critical step for high-quality visual effects is normal estimation. With this, we can access essential properties of geometric surfaces, being possible to know object orientation, for example, used to define light, shade, and other visual effects. With the point cloud model, normal estimation is usually formulated as the problem of estimating the normal of a plane tangent to the surface, turning the problem into a fitting estimation of the least-square plane. Therefore, normal estimation is reduced to an analysis of the eigenvectors and eigenvalues (or Principal Component Analysis - PCA) of a covariance matrix created from the nearest neighbors of the query point. This information is the base to create a mesh for further texture mapping and to improve lighting effects when visualizing the 3D model. Figure 6 shows a representation of the normals in a point cloud.

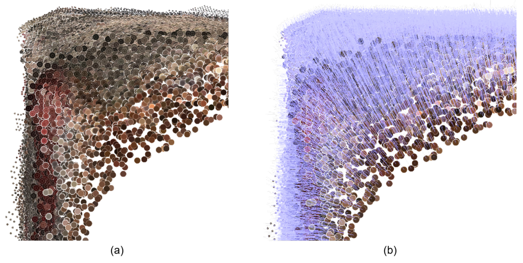

Figure 6. Results of normal estimation applied to the DTU set6 point cloud. (a) Set of points without normals; (b) Set of points with normals, where the blue lines represent the normal of each point.

3.3.4 Surface reconstruction

PCL provides some different functions to transform a set of 3D points with oriented normal into a mesh. Between them are the greedy projection triangulation, Poisson and Grid projection surface reconstruction, which are based on Marton et al. (2009), Kazhdan et al. (2006) and Li et al. (2010), respectively.

The first performs a local triangulation by projecting the local neighborhood of a point along with the point's normal and connecting unconnected points. It can deal with unorganized points coming from one or multiple scans and having multiple connected parts. But it works best if the surface is locally smooth and there are smooth transitions between areas with different point densities, which is not always the case.

On the other hand, the goal of the second is to reconstruct a watertight, triangulated approximation of the surface. It derives a relationship between the gradient of the indicator function and an integral of the surface normals field. Then, it reconstructs the indicator function from this gradient field as a Poisson problem. The output of the scalar function, represented in an adaptive octree, is then iso-contoured using adaptive marching cubes to obtain the reconstructed surface.

Finally the third uses a pair of scalar and unoriented vector functions along with a spatial grid over the domain, to make the surface reconstruction. First, it will identify the grid edges where the derivative of the scalar and unoriented vector is zero. Then the algorithm can make a polyline crossing all the grid edges, creating the curves through that vector grid.

Figure 7 shows these three surface reconstruction algorithms on the same input point cloud. We chose to use the Poisson algorithm for its smoother surfaces and ability to fill holes.

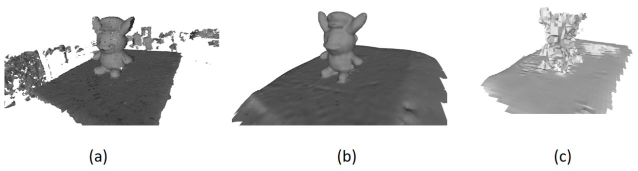

Figure 7. Surface reconstructions algorithms on the same input cloud. (a) Greedy projection triangulation; (b) Poisson surface reconstruction; (c) Grid Projection surface reconstruction

3.3.5 Texture mapping

The final step in our pipeline is to perform texture mapping on the reconstructed surface. It consists of transforming image data projected in a 3D surface, and matching with the point cloud structure of the mapped scene or object, in our case the Poisson surface. For that, every vertex in a polygon found in the dense point cloud is assigned to a texture coordinate. The final result is a realistic reconstruction of the desired object. Figure 8 shows an example of a texture being applied to the object surface.

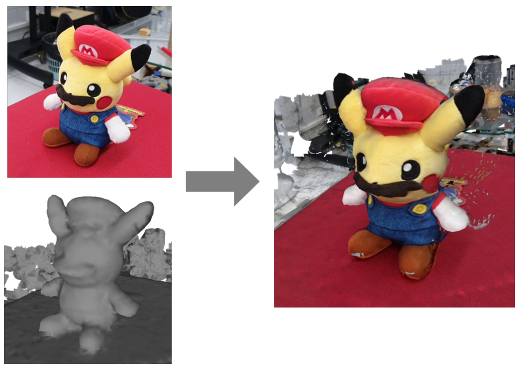

Figure 8. Final result. Texture application on the obtained Pikachu surface.

4 Solution

To maintain the application only with permissive dependencies, low execution time, and robustness, we connected the methods described in Section 3, creating the 3D reconstruction pipeline illustrated in Figure 9.

As a first step, a succession of overlapping photos was taken, to be used as the $\mathcal{I}$ dataset. Then, these images were used in our modified COLMAP SfM to find the local features, which gives: the sparse reconstruction of the scene, and intrinsic and extrinsic camera parameters. That sparse reconstruction helps to find the N most similar images for each image in the dataset.

In the inference phase, these found groups, set of images, and camera parameters was used as input to the pre-trained R-MVSNet network. This network outputs a depth map and a probability map for each image. With these maps, it is possible to use the latter to filter the former by excluding pixels depth values in which the probability does not pass a certain threshold.

After that, each filtered depth map was transformed into a point cloud to have their normals estimated using PCL. After, we fused them into one point cloud by using the extrinsic parameters of each camera to transform the points, which are in camera coordinates, to world coordinates.

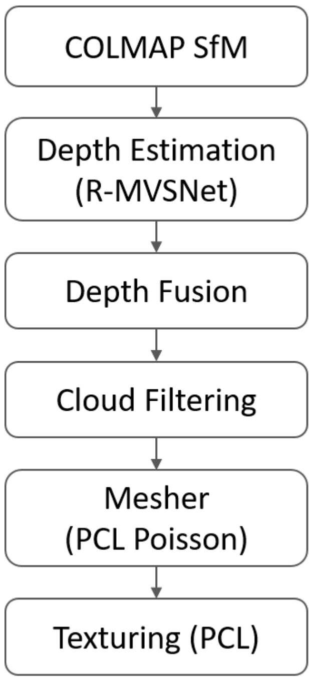

Figure 9. Our proposed pipeline. Input goes into the COLMAP SfM algorithm and follows the flow until the texturing step, where it outputs a texturized mesh of the scene.

Next, using PCL, we filtered this point cloud by removing outliers, lowering the density of points, and resampling the cloud to make it smoother. After that, a Poisson surface reconstruction algorithm is applied in the filtered cloud to obtain a mesh of the scene. Finally, this mesh goes in a texturization process, also using PCL, giving a texturized mesh that represents the complete object.

5 Results and Discussion

The proposed 3D reconstruction pipeline was evaluated through both qualitative and quantitative comparison of the reconstructed scenes with COLMAP (regarding the modifications mentioned in Section 3) and MVE pipeline (using A-KAZE as feature extractor instead SIFT/SURF algorithms, using a fully permissive application). We also made an execution time analysis for all methods. The details of these tests are presented in the next subsections[3].

We used OpenCV in our pipeline for image handling, mainly running in C++. However, the R-MVSNet is an exception, running in Python and using TensorFlow to handle the deep neural network. We managed to run the inference in C ++ using ONNX, but it was not a stable solution, and some adjustments are still needed for its conclusion. COLMAP and our pipeline use GPU acceleration in the feature matching and depth estimation steps while MVE can only run entirely in CPU. The machine used to run the test has these specifications: Intel Core i7-7700HQ 2.80GHz, 16 GB RAM, and an NVIDIA GeForce GTX 1060 6GB graphics card.

[3]A video demonstrating the quantitative results can be found at https://youtu.be/i5rlWxVzp60

5.1 Experiments

We tested our pipeline using several image sets from the DTU MVS dataset (Jensen et al., 2014). Each set has a total of 49 photos, with a 1600×1200 resolution. Figure 10 shows some examples images of the scan 6. With all scans, a ground truth point cloud is also available (Fig. 11), so we made a quantitative evaluation of the methods following the process described in Knapitsch et al. (2017).

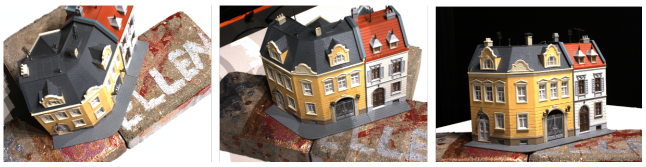
Figure 10. Example images from DTU MVS set6 used in the first experiment.

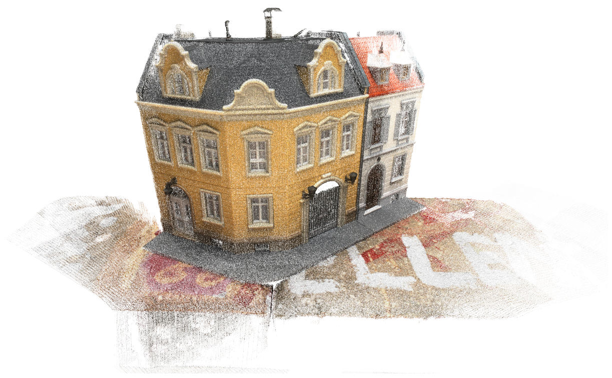
Figure 11. Ground truth point cloud of the set6 from the DTU MVS dataset.

To make this evaluation, first the point clouds were automatically aligned using the reconstructed camera poses, and then this alignment was refined using a RANSAC algorithm. The point clouds were resampled using a voxel grid filter of size $\tau/2$, being τ the distance threshold of a point being valid, to avoid bias in the evaluation by maintaining a uniform sampling on the reconstructed surface. Then, having $\mathcal{G}$ as the ground truth and $\mathcal{R}$ as the reconstructed cloud, we can calculate the distance of the reconstructed points $\mathbf{r} \in \mathcal{R}$ to the ground truth and use them to define the precision of the reconstruction $\mathcal{R}$ for any threshold τ as:

$$e_{\mathbf{r}\to\mathcal{G}} = \underset{\mathbf{g}\in\mathcal{G}}{min} \|\mathbf{r} - \mathbf{g}\|, \quad (6)$$

$$P(\tau) = \frac{100}{|\mathcal{R}|} \sum_{\mathbf{r}\in\mathcal{R}} [e_{\mathbf{r}\to\mathcal{G}} < \tau]. \quad (7)$$

Similarly, we can calculate the distance of the ground truth points $\mathbf{g} \in \mathcal{G}$ to the reconstruction and use them to define the recall of the reconstruction $\mathcal{R}$ for any threshold τ as:

$$e_{\mathbf{g}\to\mathcal{R}} = \underset{\mathbf{r}\in\mathcal{R}}{min} \|\mathbf{g} - \mathbf{r}\|, \quad (8)$$

$$R(\tau) = \frac{100}{|\mathcal{G}|} \sum_{\mathbf{g}\in\mathcal{G}} [e_{\mathbf{g}\to\mathcal{R}} < \tau]. \quad (9)$$

Table 1. Execution times of several DTU sets in first experiment

	SFM	Depth	Fusion	Mesh	Texture	TOTAL
COLMAP						
	SFM	**Depth**	**Fusion**	**Mesh**	**Texture**	**TOTAL**
Set 1	6m42s	49m5s	8m46s	1m50s	-	1h06m23s
Set 4	6m24s	55m35s	7m41s	1m55s	-	1h11m35s
Set 6	5m31s	48m48s	7m53s	2m5s	-	1h4m17s
Set 12	7m59s	44m33s	5m27s	1m45s	-	59m44s
Set 13	16m22s	46m48s	6m20s	2m42s	-	1h12m12s
Set 29	10m21s	53m56s	7m57s	3m53s	-	1h16m07s
Set 33	21m18s	57m11s	3m19s	1m8s	-	1h22m56s
Set 114	12m57s	44m23s	7m22s	2m54s	-	1h07m36s
Set 118	6m1s	43m22s	9m29s	3m37s	-	1h02m29s
Average	10m24s	49m18s	07m08s	02m25s	-	1h09m15s
MVE						
	SFM	**Depth**	**Fusion**	**Mesh**	**Texture**	**TOTAL**
Set 1	18m15s	57m24s	28s	16m14s	17s	1h32m38s
Set 4	33m29s	52m57s	17s	16m48s	14s	1h43m45s
Set 6	21m19s	45m45	21s	20m01s	15s	1:27:41s
Set 12	10m40s	34m9s	15s	10m2s	6s	55m16s
Set 13	14m13s	26m27s	16s	13m45s	10s	54m51s
Set 29	14m55s	29m58s	14s	14m41s	8s	59m56s
Set 33	52m7s	34m40s	13s	10m23s	12s	1h37m35s
Set 114	10m6s	39m42s	17s	9m28s	7s	59m40s
Set 118	3m39s	27m20s	13s	9m27s	9s	40m48s
Average	19m51s	38m42s	17s	13m25s	11s	1h12m28s
Ours						
	SFM	**Depth**	**Fusion**	**Mesh**	**Texture**	**TOTAL**
Set 1	6m42s	6m44s	54s	3m28s	0m37s	18m25s
Set 4	6m24s	6m25s	51s	3m36s	0m41s	17m57s
Set 6	5m18s	6m38s	52s	3m18s	0m31s	16m37s
Set 12	7m59s	5m43s	39s	1m27s	0m25s	16m13s
Set 13	16m22s	5m13s	34s	1m32s	0m55s	24m36s
Set 29	10m21s	6m4s	44s	7m28s	2m23s	27m00s
Set 33	21m18s	5m31s	34s	0m50s	0m16s	28m29s
Set 114	12m57s	5m41s	42s	2m5s	0m28s	21m53s
Set 118	6m1s	5m36s	39s	3m2s	0m46s	16m04s
Average	10m22s	5m57s	43s	2m58s	47s	20m48s

Finally, the precision and recall can be combined using a harmonic mean, giving us the F-score:

$$F(\tau) = \frac{2P(\tau)R(\tau)}{P(\tau)+R(\tau)}. \qquad (10)$$

The precision quantifies the accuracy of the reconstruction, and the recall quantifies the reconstruction completeness. So a high F-score can only be achieved by a reconstruction that is both accurate and complete.

Besides the benchmark datasets, we also experimented with real-world scenarios. For this test, we captured a video surrounding a highly textured mug using a Samsung Galaxy Note 10 mobile device and utilized 78 equally spaced frames with a resolution of 1080×1920. Some of these images can be seen in Fig. 12.

Finally, in the last experiment, we made a video surrounding a stuffed toy, using the Samsung Galaxy Note 10 device. We followed the same experiment, extracting different numbers of equally spaced frames from it to analyze the impact that the number of input images causes in quality and computational time. We constructed three different sets that had 46, 86, and 125 images with the same 1920×1080 resolution. Some of these images can be seen in Fig. 13.

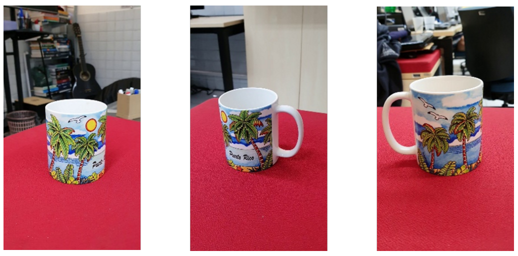
Figure 12. Example images from the Mug dataset, created by the authors, for the second experiment.

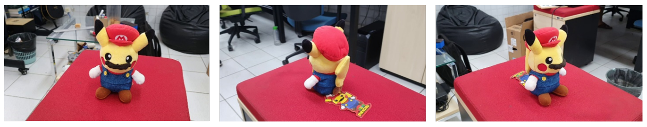
Figure 13. Example images from the Pikachu dataset, built by the authors, for the third experiment.

It is important to reaffirm that both datasets created by the authors (mug and Pikachu) followed the same protocol for analysis. The frames used were obtained automatically through the filming of the camera, being limited by the software. This methodology was proposed to compare the reaction of the solutions with reduced numbers of images and in uncontrolled environments, compared in terms of robustness, completeness, and execution time.

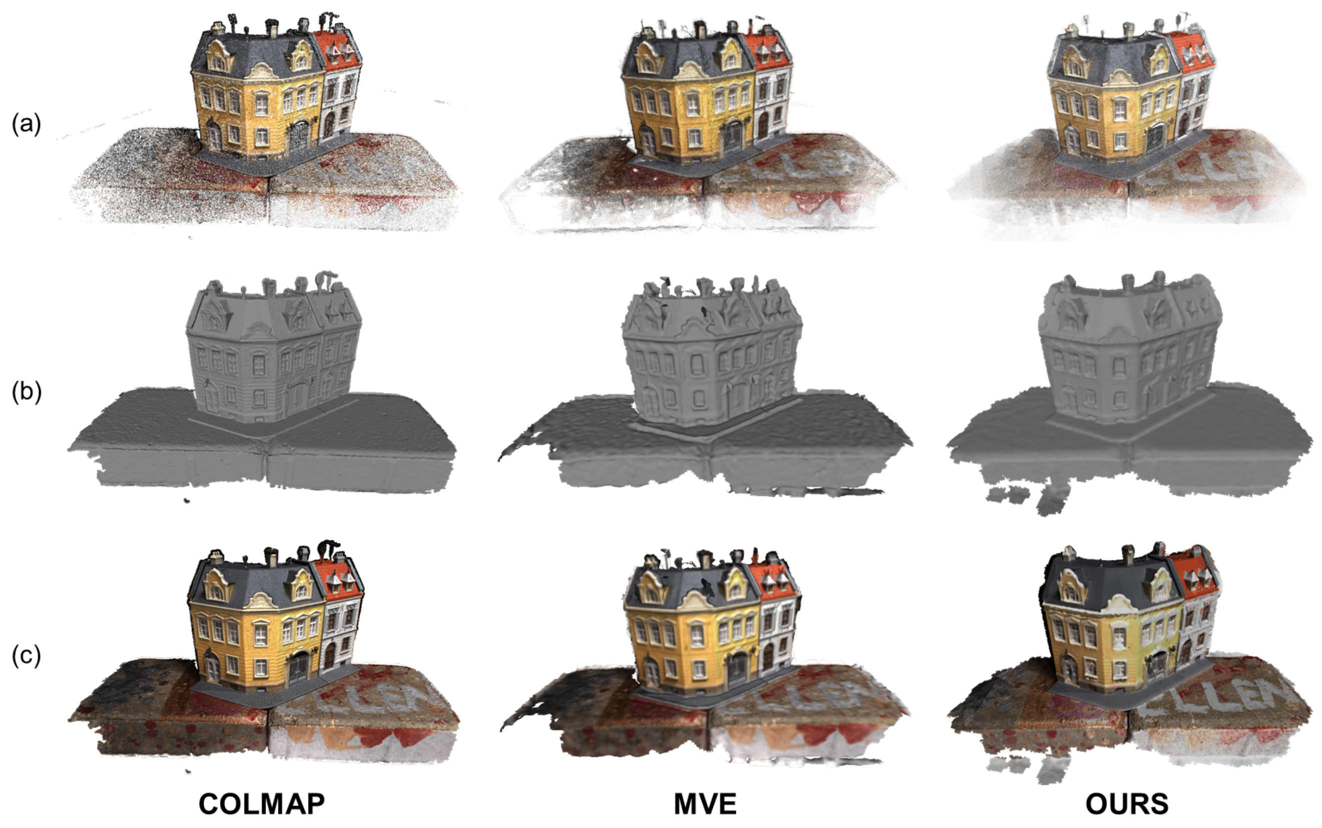

Figure 14. First experiment results: (a) Point cloud; (b) Mesh; (c) Colored mesh. The amount of noise from MVE leaves the reconstruction with more errors, which are more noticeable in the chimney. In the proposed solution, there is a loss of detail when compared to COLMAP, which is visible in the yellow wall texture.

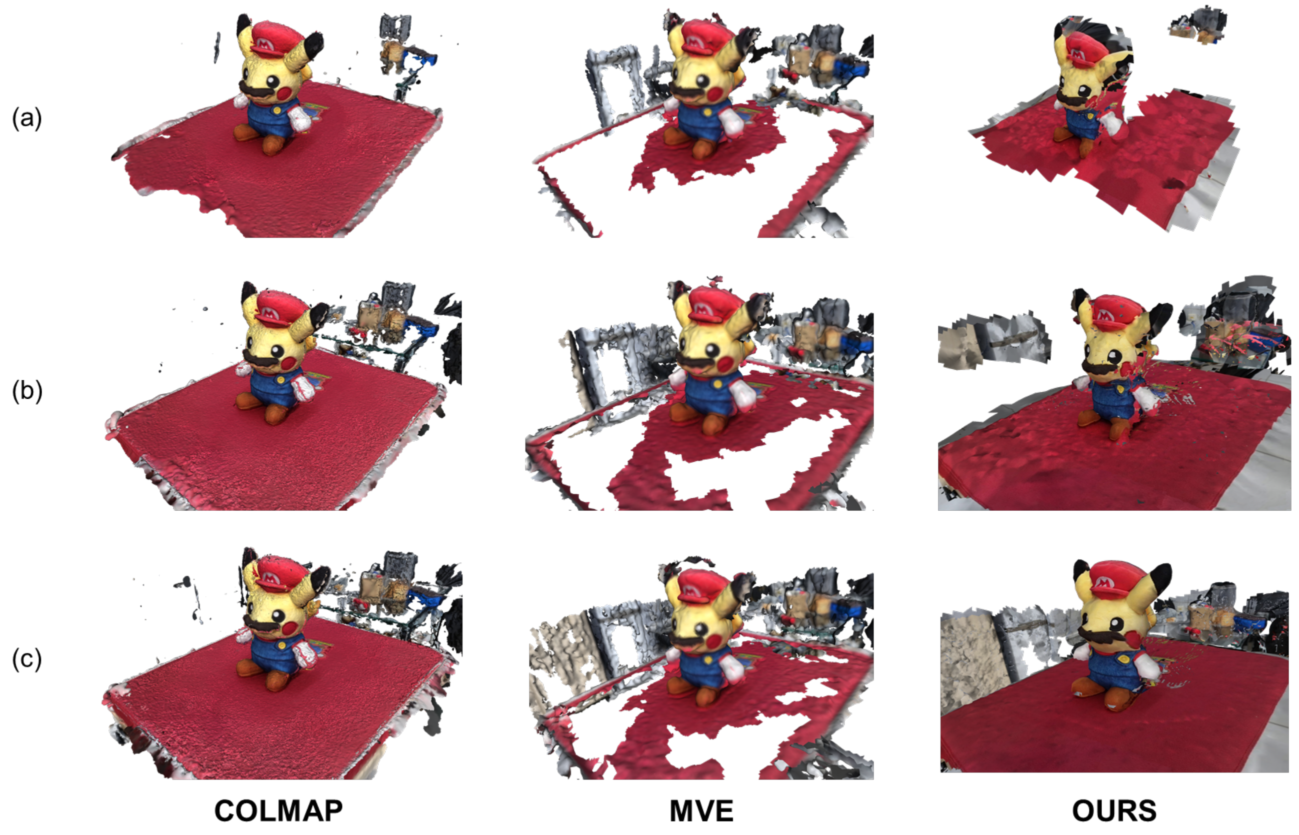

Figure 15. Comparing qualitative results using created Pikachu Dataset with (a) 46, (b) 86 and (c) 124 images.

5.2 Results

We conducted a qualitative comparison of the resulting point cloud, mesh, and colored mesh in all chosen datasets with COLMAP, MVE, and our proposed solution. Figure 14 shows the result of the experiment in DTU's scan 6. We also calculated the execution time for each step of the experiment (SfM, MVS, Depth Fusion, and Mesher) and total time for all scans. Table 1 show some examples of DTU dataset times in several experiments. The Mesher execution time includes the meshing and texturing steps.

Table 2. Precision, recall and F-score of the first experiment ($\tau = 2mm$).

	Precision	Recall	F-score
Scan 1			
COLMAP	31.62	47.14	37.85
MVE	**53.58**	**91.96**	**67.71**
Ours	33.57	50.40	43.30
Scan 4			
COLMAP	72.32	58.98	60.52
MVE	56.95	64.55	64.98
Ours	**76.08**	**62.91**	**68.87**
Scan 6			
COLMAP	70.87	78.31	74.40
MVE	55.70	80.67	65.90
Ours	**76.38**	**78.47**	**77.41**
Scan 12			
COLMAP	49.82	21.57	30.10
MVE	**49.75**	**31.00**	**38.19**
Ours	45.61	20.33	28.13
Scan 13			
COLMAP	76.56	36.02	48.99
MVE	55.10	33.92	41.99
Ours	**86.00**	**36.21**	**50.96**
Scan 15			
COLMAP	67.27	54.49	60.21
MVE	**59.23**	**67.29**	**63.01**
Ours	67.44	56.65	61.58
Scan 29			
COLMAP	53.44	31.75	39.84
MVE	**51.23**	**42.65**	**46.55**
Ours	47.97	26.79	34.38
Scan 118			
COLMAP	39.31	76.51	51.94
MVE	38.75	64.31	48.36
Ours	**90.02**	**83.40**	**86.58**
Average			
COLMAP	57.65	50.59	50.48
MVE	52.53	**59.58**	54.58
Ours	**65.38**	51.89	**56.40**

We also conducted a quantitative comparison of the reconstructed point cloud in the first experiment and the ground truth point cloud provided by the DTU dataset. We evaluated all pipelines in terms of precision, recall, and F-score, as defined in Section 5.1, following the traditional methodology present in the literature. Table 2 shows the results of this comparison with a distance threshold of 2mm. We show the results of eight different scans, in which four were the best result of our pipeline and the other four were the worst. The values in bold indicate the best results in each of the evaluated sets.

In the stuffed toy experiment (Pikachu Dataset), we used three sets with different amounts of photos to analyze the effects of the input size on the 3D reconstruction pipelines. That was a relatively more complex task, as it involved a reasonably textured image, a chaotic background, and a small number of images to create the correspondences. Figure 15 show the comparison of the colored mesh for all three pipelines with the different amounts of images used for the reconstruction and Table 4 show the entire execution time for each set for all compared pipelines.

For the mug dataset experiment, we show the reaction of our pipeline in less controlled environments compared to other implementations. The Figure 16 shows the qualitative comparison of the point cloud and mesh results. It is possible to notice that there is an influence on the background of the image. This influence happened even using a well-textured mug. Execution times have also been compared and are shown in the Table 3.

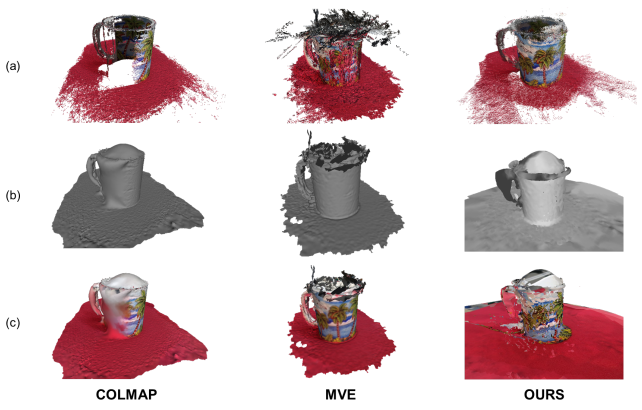

Figure 16. Qualitative results from Mug Dataset. (a) Point cloud, (b) Mesh, (c) Colored mesh.

5.3 Discussion

The qualitative assessment shows a good response from our solution compared to the other pipelines. COLMAP is complete, including using fewer images, which is more noticeable mainly at the edges of the image, when checking the mesh without coloring, in Figure 14. This problem is perceived primarily in the texturing stage using the PCL. Textures like some windows or points with an inevitable overlap can cause a certain inconsistency in the results. The reconstructions generated from the DTU dataset using our pipeline can be seen in Figure 17, and these problems are shown in Figure 18.

However, we perceive a better result of our approach for the point cloud, which influences the colored mesh, being possible to perceive a better texture in Figure 14(c). These results were noticeable in several other objects in the DTU dataset. This characteristic, combined with the much shorter reconstruction time, proves a positive result of our approach. Both COLMAP and our solution are better in terms of robustness, completeness, and runtime than MVE.

Although the times for COLMAP's SfM stage and our solution are practically the same, the average time for the other steps is a clear advantage of the proposed solution. In the average of the DTU results, the standard of our pipeline is

Table 3. Execution times of the second experiment.

	SfM	MVS	Fusion	Mesher	TOTAL
COLMAP	12min37s	34min42s	2min42s	31s	50min32s
MVE	8min8s	38min22s	14s	16min45s	63min29s
Ours	12min23 s	6min1s	39s	33s	19min36s

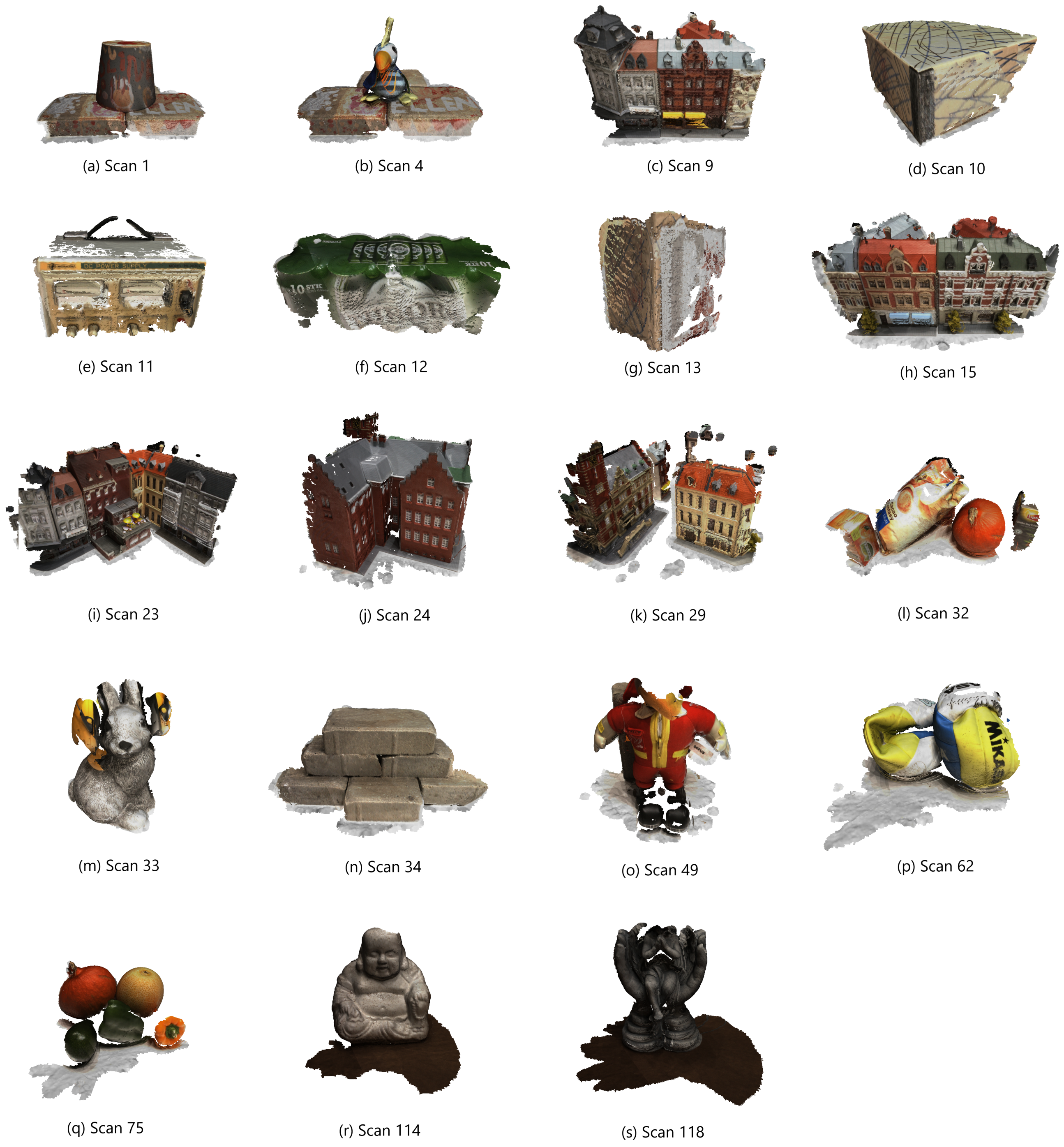

Figure 17. Qualitative results from our pipeline with the DTU dataset.

approximately 21 minutes, while others' solution best result is 69 minutes, which represents a gain of 69.56% of the time. This result is even more expressive compared to MVE, which runs on the CPU, while our solution uses the GPU (as well as COLMAP).

The analysis of the F-score was surprising. Although COLMAP has better qualitative results when compared to MVE, in none of our tests was it's F-score better. In the

Table 4. Total execution time for each set in the third experiment.

	46 images	86 images	124 images
COLMAP	51min 29s	120min 46s	174min 4s
MVE	48min 49s	110min 21s	176min 59s
Ours	13min 4s	32min 51s	62min 7s

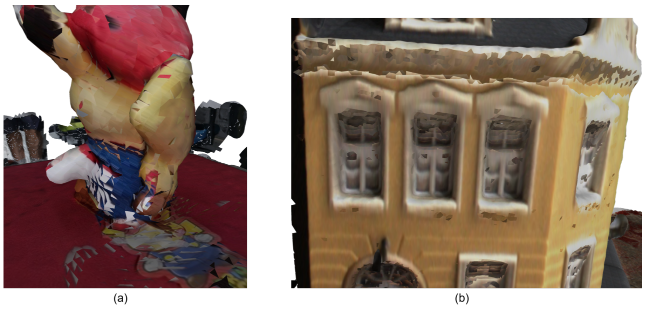

Figure 18. (a) The back of the reconstructed object from the third experiment with 124 pictures. (b) Windows of the reconstructed object from the first experiment.

cases we selected, our solution had a marginally better result on average (56.40%, against 54.58% for MVE and 50.48% for COLMAP), showing that the pipelines are quantitatively equivalent. One of the possible causes of this result is the low number of images used (49), which generated the need for the third experiment, which will be better discussed below. Noteworthy, however, is the significantly higher result in scan 118, which was the best result in the entire dataset.

Table 5. Execution time decrease (in %) of our pipeline with respect to COLMAP and MVE in different datasets

	COLMAP	MVE
DTU MVS Average	74.15%	81.05%
Mug	61.21%	69.13%
Pikachu 46	74.62%	73.23%
Pikachu86	72.80%	70.23%
Pikachu 124	64.31%	64.90%
Average	69.42%	71.71%

The mug dataset was the one that presented the worst results for all solutions. We consider this to be due to some characteristics of the dataset itself, causing inconsistencies in the point cloud, mainly in the handle and the hole. These inconsistencies generate an incomplete, blurry, and very noisy point cloud, and none of the solutions could reproduce the object satisfactorily. The point cloud of our solution is better, but the mesh generated a problem in the hole, causing a malformed object. In terms of time, our solution remained 61.52% faster than COLMAP and 69.41% faster than MVE.

Finally, in our third experiment, we verified the influence of the number of images on the final result of the approaches. Qualitative analysis shows a poor result from our system with only 46 photos, but the result improves with 86 and 124 images. This result highlights one of the flaws in our solution, which suffers a more significant impact from the number of images. However, the result with 124 photos proves to be denser and with higher quality compared to the best results of the other techniques. Even the addition of more images causes noise in the COLMAP and MVE reconstructions, with the background color bleeding from the object, as in the stuffed toy gloves (Figure 18).

Regarding the execution time, our solution kept the aver-

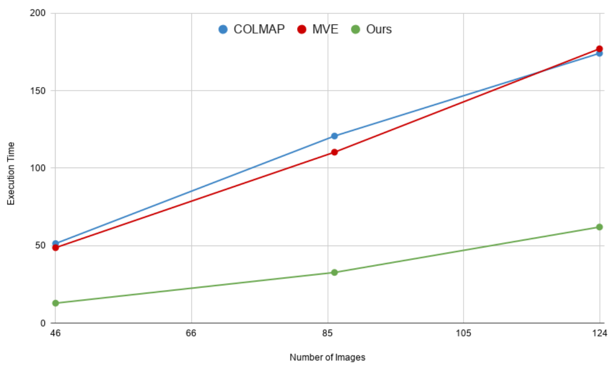

Figure 19. Relation between the number of inputs and execution time.

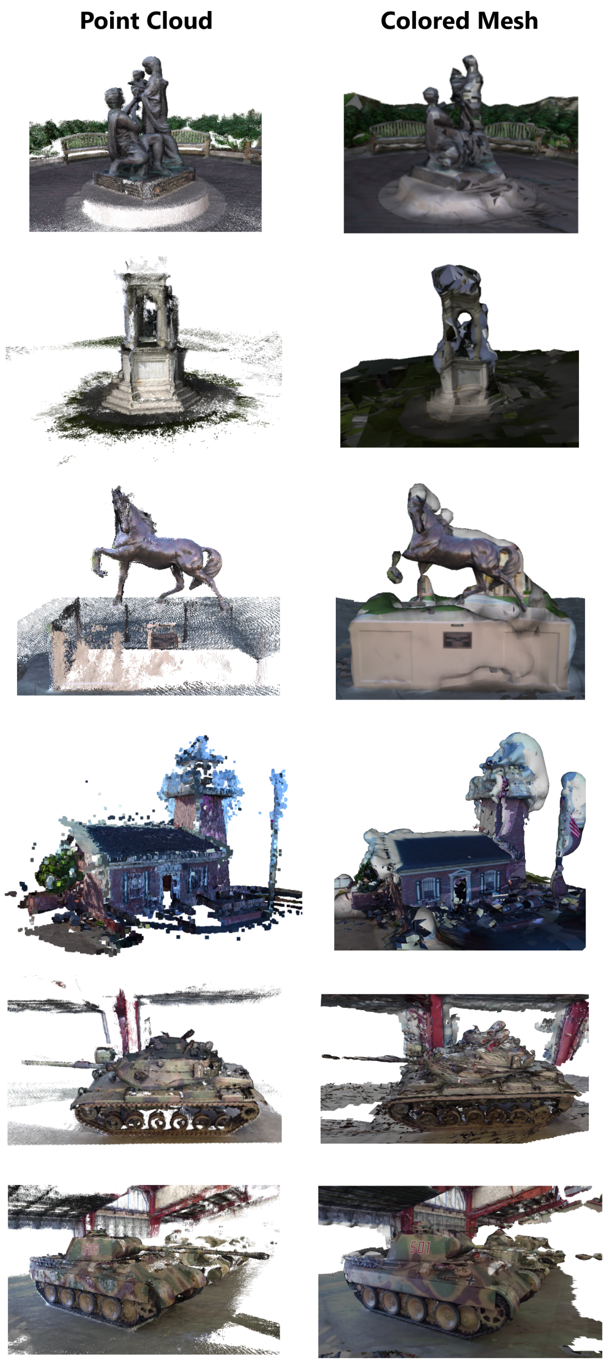

Figure 20. Qualitative results for Tanks and Temples dataset.

age speed of the other experiments. The speed was particularly higher precisely in the dataset with only 46 images but with a much worse result. With 86 pictures, and consequently with higher quality results, our solution was still approximately 20 minutes faster than COLMAP with 46 images, proving to be a valid alternative both in processing time and in image quality to the two pipelines compared. Table 5 shows the speed gain of our solution in each of the different datasets used.

Figure 17 shows the qualitative results of our dataset in several scans of the DTU dataset. Even in some cases where the F-score result was poor, the model generated was satisfactory, as seen in Figures 17(a) and 17(h). However, in some cases, our technique had problems doing the reconstruction, as in Figures 17(f) and 17(o). For these cases, a more significant number of images could be sufficient for a better reconstruction, following the other results presented here.

Because the number of input images in a photogrammetry application depends on the scene's size, for large environments like a museum or a residential area, the number of images can easily exceed the thousands. Figure 19 shows the relation between time and the number of input images for all three pipelines. In this plot, we see that our implementation has a less steep curve than COLMAP or MVE, which means that our pipeline may run much faster in a scenario that requires more images as inputs.

Figure 20 shows the qualitative results of the Tanks and Temples dataset (Knapitsch et al., 2017) for a large scale test. Each image has a resolution of 1920×1056 and we used 150, sometimes 300 images as input depending on the scene. We achieved comparable results with state-of-the-art methods, the point cloud generated preserves most of the scene, and since the scenes of this dataset are large, it makes difficult for the mesh to fully reconstruct the surface making a few artifacts in some spots, but our solution trades it with a faster execution time compared to other solutions. This results shows that our method is reliable for different datasets, without further training by the R-MVSNET model.

As for the surface reconstruction algorithms, it is important to discuss why the Poisson method was our choice, when in our experiments the greedy projection triangulation was more stable, as can be seen in Figure 21. As shown in Table 6 the execution time of the Poisson compared to the greedy and grid algorithms was significantly lower. So even with better overall results, the greedy method has a high cost in runtime, making the Poisson method more efficient for our solution.

Table 6. RunTime comparison of surface reconstruction

	Pikachu	Mug
Greedy Projection Triangulation	120 sec	9 sec
Grid Projection	180 sec	120 sec
Poisson	18 sec	3.6 sec

6 Conclusion

This paper presented a 3D reconstruction method with batch processing, with only permissive dependencies, making it

(a)

(b)

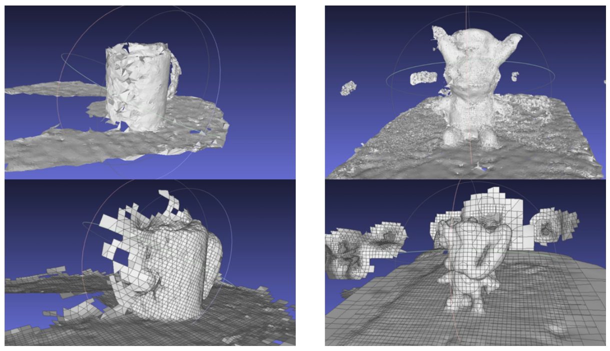

Figure 21. Surface reconstruction using the mug and pikachu dataset with different algorithms. (a) Greedy projection triangulation; (b) Poisson surface reconstruction

possible for commercial use. To achieve this goal, we combined COLMAP SfM, R-MVSNet as a depth estimation step, and PCL for mesh reconstruction, obtaining good quality results with an expressive gain of time, making it possible to create a fully merchantable technology. Even when compared to state-of-the-art methods like COLMAP and MVE, this solution was approximately 3 to 4 times quicker in runtime. The technique also showed good scalability with the increase in the number of images used for reconstruction, which allows its users to reconstruct large objects or complete scenes in less time since these approaches require a significantly large amount of information to create your point clouds.

For the 3D reconstruction task, this kind of application has the potential to make 3D scans of some artifacts, building a virtual collection for exhibition, 3D models to real objects repairing (also using virtual reality) rebuilding, in case, for example, of loss of historical material in some disaster. Another possible application is in architecture, where the professional can work with a 3D model instead of the actual scene, reducing costs. And the speed of our model is also a significant improvement. The user can do a quick scan, apply modifications to the object, becoming a solution for everyday applications with 3D models.

In future work, we intend to improve some flaws found in our solution, such as recognizing some of the dataset scans. This problem can be solved by changing the deep estimation stage, generating more accurate point clouds. Another improvement is to better understand the reaction of our solution to holes in images, as in the case of the mug, to generate more consistent results. We have already done some tests with textures, but the results have not been satisfactory, requiring further investigation.

Acknowledgements

This is an extended version of a conference paper (Lyra et al., 2020). The authors would like to thank Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (process 425401/2018-9) for partially funding this research.The results presented in this paper have been developed as part of a collaborative project between SiDi and the Center of Informatics at the Federal University of Pernambuco (CIn/UFPE), financed by Samsung Eletrônica da Amazonia Ltda., under the auspices of the Brazilian Federal Law of Informatics no. 8248/91. Professor Fabio Q. B. da Silva holds a research grant from the Brazilian National Research Council (CNPq), process #314523/2009-0.

References

Alcantarilla, P. and Solutions, T. (2011). Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, 34(7):1281–1298.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Chen, R., Han, S., Xu, J., et al. (2020). Visibility-aware point-based multi-view stereo network. *IEEE transactions on pattern analysis and machine intelligence*.

Chen, R., Han, S., Xu, J., and Su, H. (2019). Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1538–1547.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Fuhrmann, S. and Goesele, M. (2014). Floating scale surface reconstruction. *ACM Transactions on Graphics (ToG)*, 33(4):1–11.

Fuhrmann, S., Langguth, F., and Goesele, M. (2014). Mve-a multi-view reconstruction environment. In *GCH*, pages 11–18. Citeseer.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. (2007). Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

Haner, S. and Heyden, A. (2012). Covariance propagation and next best view planning for 3d reconstruction. In *European Conference on Computer Vision*, pages 545–556. Springer.

Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., and Aanæs, H. (2014). Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE.

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7.

Knapitsch, A., Park, J., Zhou, Q.-Y., and Koltun, V. (2017). Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13.

Li, R., Liu, L., Phan, L., Abeysinghe, S., Grimm, C., and Ju, T. (2010). Polygonizing extremal surfaces with manifold guarantees. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, pages 189–194.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Lyra, V. G. d. M., Pinto, A. H., Lima, G. C., Lima, J. P., Teichrieb, V., Quintino, J. P., da Silva, F. Q., Santos, A. L., and Pinho, H. (2020). Development of an efficient 3d reconstruction solution from permissive open-source code. In *2020 22nd Symposium on Virtual and Augmented Reality (SVR)*, pages 232–241. IEEE.

Marton, Z. C., Rusu, R. B., and Beetz, M. (2009). On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan.

Rusu, R. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Rusu, Radu Bogdan, S. C. (2011). Point cloud library. `https://pointclouds.org/`.

Schönberger, J. and Frahm, J. (2016). Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schönberger, J., Zheng, E., Frahm, J.-M., and Pollefeys, M. (2016). Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer.

Schops, T., Schönberger, J., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., and Geiger, A. (2017). A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 519–528. IEEE.

Strecha, C., Fransens, R., and Van Gool, L. (2004). Wide-baseline stereo from multiple views: a probabilistic account. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE.

Ullman, S. (1979). The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426.

Weilharter, R. and Fraundorfer, F. (2021). Highres-mvsnet: A fast multi-view stereo network for dense 3d reconstruction from high-resolution images. *IEEE Access*, 9:11306–11315.

Yao, Y., Luo, Z., Li, S., Fang, T., and Quan, L. (2018). Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783.

Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T., and Quan, L. (2019). Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5525–5534.

Zhang, K., Liu, M., Zhang, J., and Dong, Z. (2021). Pa-mvsnet: Sparse-to-dense multi-view stereo with pyramid attention. *IEEE Access*, 9:27908–27915.

Zheng, E., Dunn, E., Jojic, V., and Frahm, J.-M. (2014). Patchmatch based joint view selection and depthmap estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1517.