Computer Vision 2: 3D Meshing

Gabriele Bani
11640758
gabriele.bani@student.uva.nl

Andrii Skliar
11636785
andrii.skliar@student.uva.nl

Andreas Hadjipieris
11730064
andreas.hadjipieri@student.uva.nl

1 Introduction

In this report, we will explore the capabilities of 3D Meshing and Watertighting. We generate a 3D mesh from registered point clouds. We work with the provided data from depth and colour images and camera poses of 8 consecutive frames. Later on we add texture on the created meshes, by using the RGB images of every frame.

2 Depth-based and Texture-based 3D Reconstruction Comparison

Comparing two approaches appear as a challenging question due to the fact that though advantages and disadvantages seem trivial to see, a lot of the issues of initially introduced depth-based and texture-based 3D reconstruction algorithms were eliminated by improved versions of those algorithms. However, there are several main differences between these two approaches, which are:

- Depth-based 3D reconstruction algorithms need to have point clouds as the input, which means, that there should be depth sensors built into camera being able to measure depth in real time. With modern increase in computing power, that is not as much of an issue as before, however, in certain cases, it might still be too computationally expensive to use depth sensors.

  However, removing the necessity for estimating depth from 2D images, this approach is much more computationally efficient during the reconstruction process, i.e. given that the depth has already been estimated. Also, despite being initially used only for rigid transformation estimation, more modern approaches can learn non-rigid transformations [1]. Improvements have also been suggested for making algorithm more robust to initialization and outliers and also adding the scaling factor in the ICP algorithm thus improving performance and accuracy [5] [6] [4] [10] [8]. Also, [3] has suggested an improvement to ICP algorithm, which searches for corresponding pairs on subsets of the entire data as this can provide structural information to benefit the registration.

  In general, this approach gives significantly better results than texture-based 3D reconstruction algorithms and also tends to be more efficient, but requires additional information in form of depth in order to work properly. Due to the fact that for this approach, it usually boils down to 3D point registration, which is much easier than doing that together with 3D reconstruction as texture-based algorithms do.

- The main advantage of texture-based algorithms is also its main curse. The fact that you don't need precomputed depth in order for this approach to work is very appealing, because the only thing we need then are 2D photos from different angles in order to reconstruct 3D surface and taking those don't need any special cameras. However, despite being very attractive, this approach performs much worse than depth-based 3D reconstruction

algorithms and also needs a lot of assumptions in order to work properly, i.e. for affine structure from motion, we assume that cameras are affine.

Also, these approaches heavily rely on feature matching between two images and therefore might not always give good results given that the feature matching might not be good enough even when using epipolar constraints and RANSAC to make it more efficient and robust to outliers. One more issue of that approach is that it's not too efficient as RANSAC might need a lot of iterations in order to converge to reasonable results. One more issue is that this approach can't really incorporate scaling factor into the algorithm itself.

However, a lot of improvements were suggested to make algorithm work better by removing additional assumptions on cameras [2] [11]; including scaling factor [12]; using bundle adjustment [14] along with conjugate gradients [7] thus making it possible to reach linear time complexity [13]; including 3D curves for better structure estimation [9]. These methods have greatly improved the performance of this approach, however, not making it better than many of simpler depth-based 3D reconstruction algorithms.

The best solution, however, would be to use these two approaches together, which comes logically from the main advantages and disadvantages of both groups of algorithms. More specifically, we can use texture-based algorithms for estimating structure (thus, depth) and then use depth-based algorithms for 3D point registration. We can, of course, expect it to perform worse than depth-based algorithms provided with depth measured by sensors. However, it gives us higher flexibility as we don't need any additional information other than 2D photos from different points of view for robust 3D estimation by allowing to incorporate advances of both depth-based and texture-based 3D reconstruction algorithms.

3 Marching Cubes

Marching Cubes is a very peculiar algorithm. This is a perfect example why the simplest approaches are not always the best. The algorithm iteratively "walks" through the 3D field taking 8 neighbor locations at a time (forming a cube) then determining the polygons needed to represent the part of the surface that passes through this cube. Later on the polygons merged together to form a surface that approximates the desired surface. The only advantage of this algorithm is that its easy to grasp. Experimenting with it seemed very unpleasant since not only the results were very rough (cubed) but the algorithm took a lot of unnecessary time in contrast to the poisson approach.

The parameter that influences most the result of the algorithm is the resolution, which also affects heavily the running time, as much more computations are needed in order to achieve a greater resolution. The resolution parameter acts as the number of blocks each dimension is splitted. The higher the resolution the clearer the image becomes since the building cubes are very small in size. We have experimented with multiple values for this hyper parameter, and in fig. 3 we show visualizations for resolution 25, 50, 100, and 200. Notice that in order to meaningfully show results, we apply coloring here otherwise it would be much harder to grasp the 3D shape of the image, especially in 2D as we show in the figure. The first immediate result is the quality of the reconstruction, which is much lower to the one of Poisson algorithm. Although we have no quantitative measure to express the quality of these results, it can be clearly seen that there are many pieces of the 3D mesh that did not belong to the original model. Also, we can see that the lower part of the chest is not reconstructed at all in many parts, especially on the back. This prevents the hole filling procedure to fill holes.

Regarding the resolution, we can clearly see its effects on the reconstructed meshes. With low resolution, only a small number of polygons is generated, and the result is not sharp enough. With higher resolutions, such as 100 or more, the fine grained details are reconstructed successfully, so we can distinguish, for example, the change in shape happening between the neck and the t-shirt. However, we can see that increasing the resolution after a certain points does not give any more benefits, as the high frequency details look the same for resolutions 150 and 200. Also, we can notice by comparing with Poisson algorithm, that

We also show results for different setPercentageExtendGrid values, which define how much free space should be left inside the grid between the bounding box of the point cloud and the grid limits. This parameter clearly affects the result, generating very large noise objects for high values of the parameter, as seen in fig. 2. So, for all other experiments, we keep the default value of 0.0.

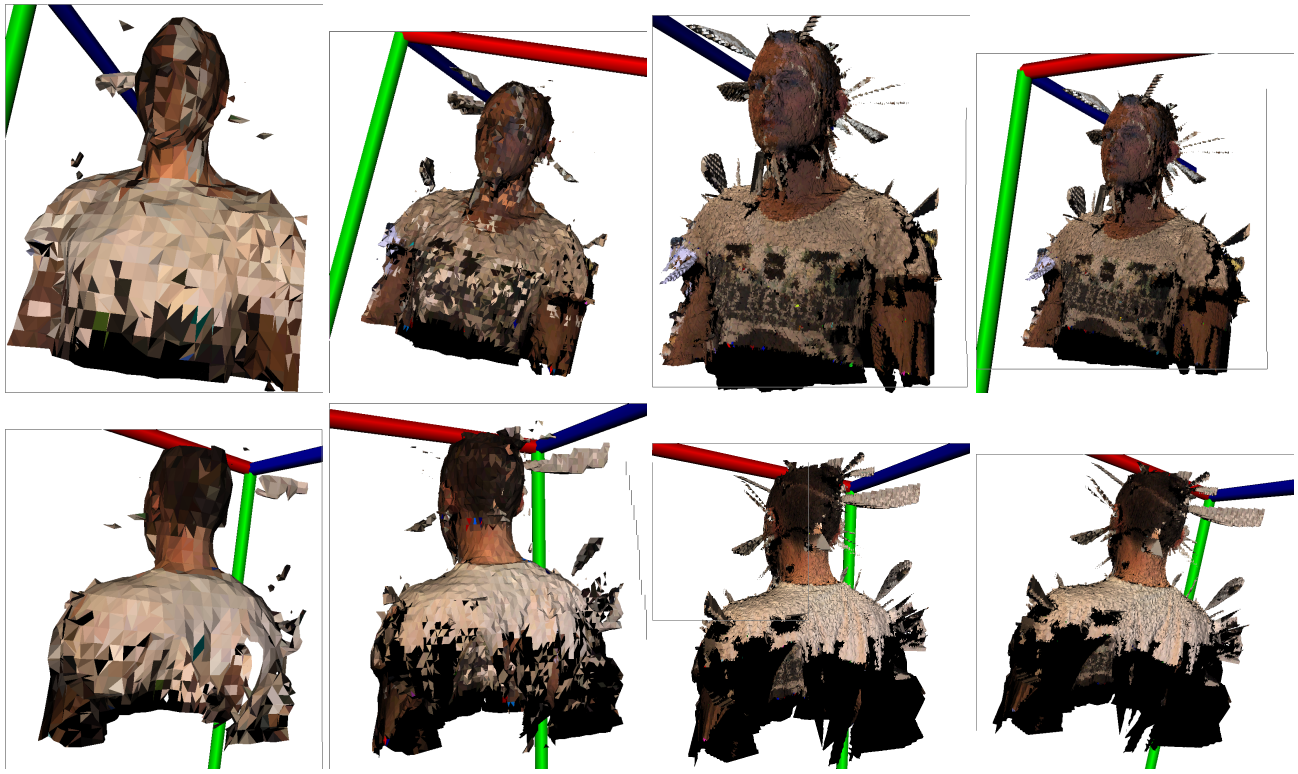

Figure 1: Marching cubes applied with resolutions 25, 50, 100, 200. It can be clearly seen that higher resolutions achieve more realistic reconstructions. Note that the wrong coloring of the mesh, as discussed in the coloring section, is due to the fact that we use our first baseline method for coloring.

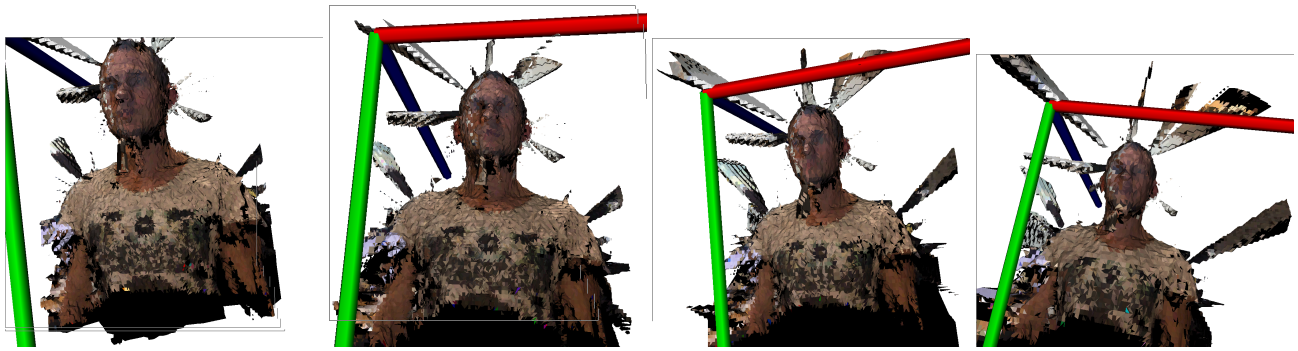

Figure 2: Marching cubes applied for PercentageExtendGrid values 0.1, 0.25, 0.5, 0.9. (resolution 100). For high values of the parameter the outlier objects increase in size, making the reconstruction much worse.

4 Poisson

Poisson algorithm is a much more complicated algorithm, than Marching Cubes, however, it also tends to produce much better results and is much more efficient. Poisson surface reconstruction algorithm depends on many parameters, however, throughout our experiments, seen in fig. 3, we have found, that the main parameter, that influences the final result is depth parameter.

Of course, this is due to the fact, that we are observing a much simpler case than the ones you might encounter in real-life scenario. Namely, there is no need to set solverDivide and isoDivide parameters, specifying the depth at which a block Gauss-Seidel solver is used to solve the Laplacian equation, because they are used in order to reduce memory overhead, which is not an issue in our

case due to a lower number of points. Also, our samples are noise-free and therefore we can use default low value for samplesPerNode while still getting smooth reconstruction. We have also tried to modify values for scale factor, however, we have found out that careful tuning of this parameter is not necessary as the filter produced with default scale, outputs results, which are smooth enough and are not over-smoothed at the same time.

The reason, that depth greatly influences the result is that while depth of the tree increases, reconstructions are becoming capable of capturing finer details. However, as the depth increases, it can overfit to the data thus not smoothing out the noise present in the initial data, which might also lead to poor results. Therefore, with small values for the depth such as 3, the number of resulting polygons is very low and thus we can only see a very approximated reconstruction of the original figure. With a value of 6, we have a more defined result, which however clearly misses the fine grained details of the original figure. With a value of 9 we obtain a very detailed 3D model, which however also incorporates some noise, which we did not manage to distinguish from the details of the original model.

Another remarkable difference in the reconstructions with different depth parameters can be seen by looking at the lower part of the 3D model. We can see in fact that higher depth values cause the lower part of the chest to be smoothed out and not get reconstructed. This also results for example in the arm and the chest to be separated for depth 9, with the torso being very small, which is not correct as the original images clearly show a full torso. The reason for that might be that functions, found for the octree, overfit too much to the data and are not providing good representation of the indicator function anymore thus outputting the results, which are not meaningful anymore. Also, we would like to note that higher depth causes the algorithm to have longer running times.

The other parameter that most influences results of poisson reconstruction is the number of samples per node. As suggested from pcl documentation, values in the range from 1 to 5 can be used when no noise is assumed, while for noisy data we can use values of 15 - 20. In fig. 4 we show results for different parameter values. As we can see, indeed low values of the number of samples per node lead to little change in the mesh (where we set the depth to 9 to have a very sharp reconstruction), while higher values indeed produce smoother results. However, we can see that in our case using a small values of such parameter is useful to filer out the noise due to the high depth.

We also fill holes, and do so using the library VTK, in particular using the class vtkFillHolesFilter. This class, as described in the documentation, works by first detecting boundary edges, linking them together into loops, and then triangulating the resulting loops. The parameter holeSize for the hole filling procedure produces almost always the same results in our case, as long as it is greater than 1.

5 Colouring

In this section we explore the problem of coloring the 3D mesh with meaningful values. Since we have as input images with RGB color representations, we can exploit this information in order to map these colors to the reconstructed mesh. We have used the RGB images of every frame to extract the colour for every point. We extract U and V coordinates of the RGB image for points that are seen by the camera for every frame, then assign the colour of the point based on the those U and V coordinates on the real image. In our baseline approach, we map every point to the 3D space of our mesh and override the colour if we encounter the same point again. This approach seemed to work well and its very intuitive. The problem with this approach was that in one of the frames the lighting from the windows was too strong resulting in change of the colour of the persons left face. In particular, in our case, we have that the left side of his face is more bright than the other. One way to solve the problem could be changing the order of the frames for the coloring, in order to overwrite the bright points. Although this improves the coloring, as shown in fig. 5, this solution is not general enough, and can work pretty bad on other situations.

So, we first tried to average the colors of every vertex between frames, which strangely resulted in the loss of color of many vertices. We argue that this result is due to a bug in our code, as it makes no sense that the result misses so many points. Furthermore, since the mean is sensitive to outliers, especially in the case with such low number of frames, we implemented a median assigning of color, which simply collects color for every frame at all the vertices, and takes the median color over frames as the final color. This, as shown in fig. 5, actually produces more visually coherent results in the head of the person. However, it produces bad results on the lower part of the torso and on the right

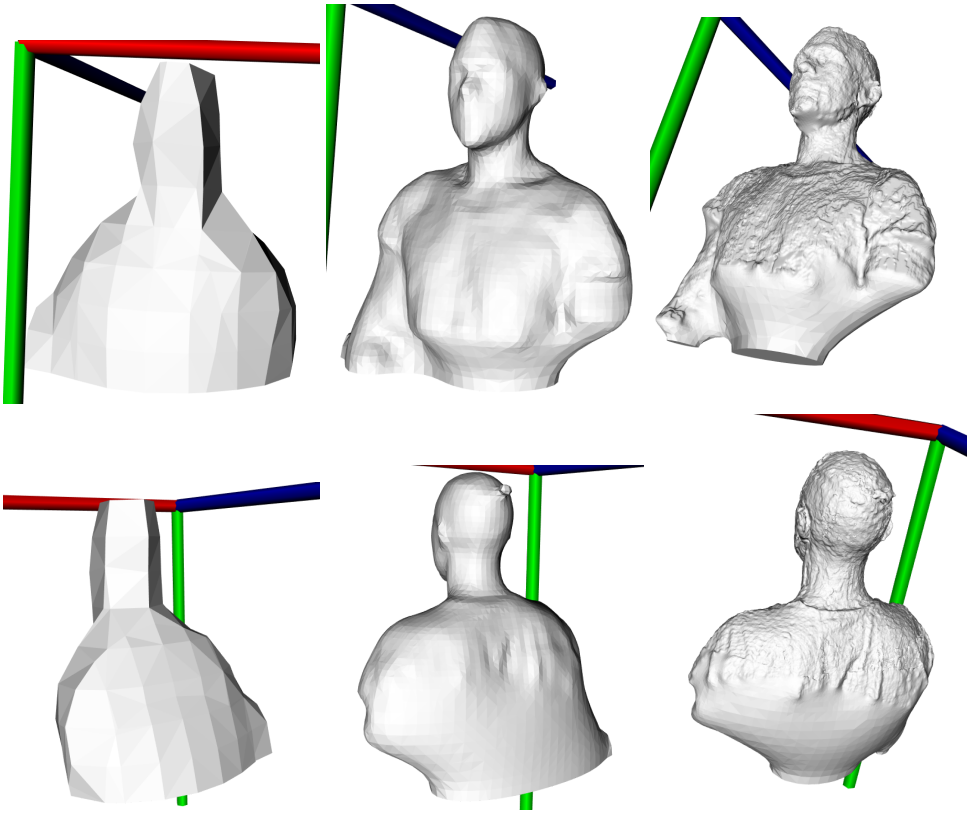

Figure 3: Poisoon reconstruction of the point cloud with changing the depth parameter. (3-6-9, left-center-right)

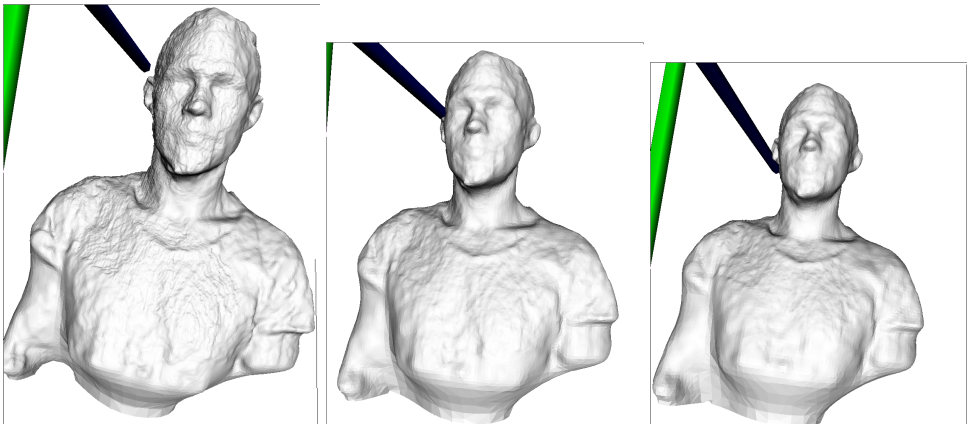

Figure 4: Poisoon reconstruction with setSamplesPerNode of 5, 15, and 20 (depth = 9)

arm. The reason this might happen is that for those parts, only vertices with non coherent coloring are seen from only two frames, which makes the result unreliable. With higher number of frames however, we expect the median approach to be more robust to outliers than the previous approach. The main drawback of the median approach is however that it needs to store all the colors of the points, while the average approach can replace averages on the fly. This leads to higher memory consumption, and causes the program to crash with too high values of depth parameter for Poisson, as a large number of vertices is generated. For this reason, we show results for depth 7.

Figure 5: Reconstruction with Poisson with texture. From first to last row: Baseline coloring, inverted frames, average colors, median colors.

6 Conclusion

We have implemented two 3D mesh generation algorithms and experiment their usage on a sample 3D model representing the upper part of a person. Overall, we managed to to reconstruct a human shape for both Poisson and Marching Cubes, although from our results, it is clear that Poisson is much more precise than Marching Cubes. The latter in fact creates many 3d small objects that were not in the original images, and fails to reconstruct parts of the chest of the person. Poisson however can be tuned to reconstruct a very fine 3d representation of the input images, while also allowing to smooth the result with specific parameters. Thus, for real life applications, based on our experiments we would choose Poisson algorithm, as it gives better results in visual terms, is easier to tune, and generally requires less execution time than marching cubes.

During our experiments, we also deal with aspects of 3D reconstructions which are not considered in the two algorithms. In particular, we filled the holes that resulted from the reconstructions when possible, and also implemented a method for using color information of images of the person in order to correctly visualize colors also in the reconstructed surface. Since colors were very different in some of the images, we found out that using colors from just one image for points which were appearing in more images lead to not coherent colors. To achieve better results, we have implemented averaging of colors by point and median calculation, which allows us to obtain a more coherent coloring of the 3D mesh.

7 Self Evaluation

- Andrii Skliar: Core implementation, report
- Gabriele Bani: Coloring 3D Model, experiments
- Andreas Hadjipieris: Coloring 3D Model, experiments

References

[1] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[2] Avinash Bhaskaran and Anusha Sridhar Rao. Structure from motion using uncalibrated cameras. 2016.

[3] Junfen Chen and Bahari Belaton. An improved iterative closest point algorithm for rigid point registration. In *International Conference on Machine Learning and Cybernetics*, pages 255–263. Springer, 2014.

[4] Shaoyi Du, Nanning Zheng, Lei Xiong, Shihui Ying, and Jianru Xue. Scaling iterative closest point algorithm for registration of m–d point sets. *Journal of Visual Communication and Image Representation*, 21(5-6):442–452, 2010.

[5] Shaoyi Du, Nanning Zheng, Shihui Ying, Qubo You, and Yang Wu. An extension of the icp algorithm considering scale factor. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V–193. IEEE, 2007.

[6] SY Du, JH Zhu, NN Zheng, JZ Zhao, and C Li. Isotropic scaling iterative closest point algorithm for partial registration. *Electronics letters*, 47(14):799–800, 2011.

[7] Michal Havlena. Incremental structure from motion for large ordered and unordered sets of images. 2012.

[8] Christian Langis, Michael Greenspan, and Guy Godin. The parallel iterative closest point algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 195–202. IEEE, 2001.

[9] I. Nurutdinova and A. Fitzgibbon. Towards pointless structure from motion: 3d reconstruction and camera parameters from general 3d curves. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2363–2371, Dec 2015.

[10] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. 2001, 10 2001.

[11] M. Sainz, N. Bagherzadeh, and A. Susin. Recovering 3d metric structure and motion from multiple uncalibrated cameras. In *Proceedings. International Conference on Information Technology: Coding and Computing*, pages 268–273, April 2002.

[12] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1413–1419, Sept 2009.

[13] C. Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 127–134, June 2013.

[14] J. Zhang, M. Boutin, and D. G. Aliaga. Robust bundle adjustment for structure from motion. In *2006 International Conference on Image Processing*, pages 2185–2188, Oct 2006.