Merging Overlapping Depth Maps into a Nonredundant Point Cloud

Tomi Kyöstilä, Daniel Herrera C., Juho Kannala, and Janne Heikkilä

University of Oulu, Oulu, Finland
tomikyos@paju.oulu.fi,
{dherrera,jkannala,jth}@ee.oulu.fi

Abstract. Combining long sequences of overlapping depth maps without simplification results in a huge number of redundant points, which slows down further processing. In this paper, a novel method is presented for incrementally creating a nonredundant point cloud with varying levels of detail without limiting the captured volume or requiring any parameters from the user. Overlapping measurements are used to refine point estimates by reducing their directional variance. The algorithm was evaluated with plane and cube fitting residuals, which were improved considerably over redundant point clouds.

Keywords: point cloud simplification, surface modeling.

1 Introduction

Creating a point cloud from large indoor environments using a depth camera (such as the Kinect from Microsoft) without any simplification results in a lot of redundant points. This is caused by the sequence of depth maps having overlap, which is necessary for view registration. The redundancy unnecessarily increases memory requirements and computation time when the cloud is further processed.

This problem could be approached by first combining all depth maps into a single redundant point cloud and then simplifying it. This would require storing the whole cloud in memory and applying the simplification as a post-process.

Instead, we present a method for incrementally creating a point cloud from sequential depth maps without adding redundant points. The measurements from the depth sensor have directional variance, and this is taken into account by having a covariance matrix for each point. New measurements reduce the variances of existing nearby points instead of adding redundant points. Existing points are never removed, which allows for varying the level of detail by varying the distance from the sensor to the measured surface. Thus, it is not necessary for the user to predecide the point density, point count, or any such parameter for the resulting cloud. Moreover, our method does not impose a limit on the size of the captured volume.

J.-K. Kämäräinen and M. Koskela (Eds.): SCIA 2013, LNCS 7944, pp. 567–578, 2013.

2 Related Work

Most existing methods of point cloud simplification require the user to supply parameters such as the desired point density [1], point count [2, 3], or a threshold for clustering [4]. These methods concentrate on capturing individual items, and when capturing large varied environments, picking optimal parameters can be time consuming or even impossible.

Merrell et. al [5] investigated the 3D reconstruction of entire building exteriors from video in real time. They fused adjacent depth maps to reduce the error from their stereo algorithm and finally triangulated a mesh from the fused maps. The algorithm of [5] is mainly designed for rejecting outliers, which are common in depth maps obtained by passive stereo imaging, and it does not utilize uncertainties of different depth estimates in the merging process in a statistically justified manner like we do. Also, in such cases where a surface mesh is the desired output of depth map fusion, our nonredundant point cloud can be transformed into a mesh, as in [5] or [6].

Recently, due to the emergence of inexpensive depth cameras, such as the Kinect, there has been a lot of interest for 3D-modeling techniques that use image sequences captured with a moving depth camera. Many of these recent approaches are real-time mapping methods which represent the world with a voxel grid [7–10]. These methods are limited by a fixed resolution imposed by the grid and are not very convenient for large environments, although there have been some efforts to increase their applicability [9, 10].

Henry et. al [11] modeled large indoor environments using surfels (or surface elements) [12] instead of a point cloud. Surfels are discs with position, normal, and size properties. They also included with each surfel a measure of confidence, which increased when a surfel was seen from multiple angles. However, such a heuristic measure of confidence does not have a sound statistical justification and may imply suboptimal merging of different measurements.

In our method, the confidence of the position of a point is described with a covariance matrix which appropriately takes into account the measurement uncertainty of each point. For example, points originating from the same physical location of a surface may have widely varying measurement uncertainties in different depth maps due to the varying camera locations with respect to the surface. That is, the accuracy of depth cameras depends heavily on the observation distance [13] and our approach takes this into account in the merging process. To the best of our knowledge, such an approach has not been previously used for integrating depth maps into a point cloud.

3 Merging Depth Maps

This section describes our approach. A general overview of the algorithm is outlined first and the details are explained in subsections 3.2 and 3.3.

3.1 Overview

Given a sequence of depth maps with known camera poses, we represent the localization uncertainty of each point of each depth map using a 3×3 covariance matrix, which corresponds to an ellipsoid in the 3D world frame. We then utilize these covariance matrices in a sequential merging process, which produces a nonredundant point cloud by incrementally adding new depth maps to the existing point cloud so that redundant new depth measurements near previously added points are used to refine the location of existing points, i.e. without increasing the point count. New points are added to the point cloud only if they represent previously uncovered parts of the scene. The refinement process is based on a recursive implementation of the so called *best linear unbiased estimator* (BLUE) [14] as detailed in subsection 3.3.

Hence, as described above, the output of the algorithm is a nonredundant point cloud and the input is a sequence of registered *views*, which consist of a depth map, a color image, and extrinsic parameters of the camera. In addition, we have connectivity information for the views. That is, for each view we know its *connected views* whose fields of view overlap and may hence contain common points. This implies that for each incremental addition of a depth map one only needs to consider overlap with those previously added points which are visible in the views connected to the current view. Since the number of connected views is bounded in practice and usually small even for long image sequences, our approach is scalable to large environments. The connectivity information for the views can be inferred from the camera poses (see e.g. [15]) or it may be directly obtained as a by-product of the registration process, which is typically carried out by solving the simultaneous localization and mapping problem (SLAM). Nevertheless, if the number of views in the sequence is small and there is no need for optimizing the performance, one may also assume that all the views are directly connected with each other.

An overview of our approach is described in pseudocode in Algorithm 1. In this algorithm all the points (i.e. those already added to the point cloud and those back-projected from the depth maps) consist of a 3D position estimate and its covariance matrix which characterizes the uncertainty of the estimate. The form of the covariance matrix for each individual depth measurement is described in subsection 3.2 below and the procedure for updating the position estimate and its covariance matrix during the merging process (procedure *RefinePoints* in Algorithm 1) is described in subsection 3.3.

3.2 Uncertainty of Individual Depth Measurements

A depth camera can be considered as a mapping f from the three-dimensional camera coordinate frame to measurements, i.e.

$$\tilde{\boldsymbol{m}} = f(\tilde{\boldsymbol{p}}), \tag{1}$$

where $\tilde{\boldsymbol{m}} = (u, v, w)^\top$ is the measurement point corresponding to scene point $\tilde{\boldsymbol{p}} = (x, y, z)^\top$ so that (u, v) are the pixel coordinates and w is the observed

Algorithm 1. The depth map merging algorithm

```
function MergeViews(views)
    cloud ← ∅
    processed_views ← ∅
    for all curr_view ∈ views do
        used_measurements ← ∅
        for all proc_view ∈ processed_views do
            if curr_view and proc_view are connected then
                RefinePoints(cloud, proc_view, curr_view, used_measurements)
            end if
        end for
        for all measurement ∈ curr_view do
            if measurement ∉ used_measurements then
                insert measurement into cloud
            end if
        end for
        insert curr_view into processed_views
    end for
    return cloud
end function

procedure RefinePoints(cloud, proc_view, curr_view, used_measurements)
    for all $\hat{p}$ ∈ (points inserted into cloud from proc_view) do
        if $\hat{p}$ projects onto the pixel grid of curr_view then
            pixel_coordinates ← the position where $\hat{p}$ projects onto the pixel grid
            $\tilde{p}$ ← the measurement point from curr_view at pixel_coordinates
            $\hat{p}'$ ← a refined point created from $\hat{p}$ and $\tilde{p}$
            if MahalanobisDistance($\hat{p}'$, $\hat{p}$) < 3 and
               MahalanobisDistance($\hat{p}'$, $\tilde{p}$) < 3 then
                replace $\hat{p}$ in cloud with $\hat{p}'$
                insert $\tilde{p}$ into used_measurements
            end if
        end if
    end for
end procedure
```

disparity (or depth) value in the depth map. The back-projection of a point $\tilde{\boldsymbol{m}}$ is obtained by the inverse mapping $\tilde{\boldsymbol{p}} = f^{-1}(\tilde{\boldsymbol{m}})$. Given an estimate of the measurement noise, represented by the covariance matrix $\boldsymbol{D}$ of $\tilde{\boldsymbol{m}}$, one may compute the first-order approximation of the covariance matrix $\boldsymbol{C}$ of the back-projected point $\tilde{\boldsymbol{p}}$ by using the so-called backward transport of covariance [16] as follows

$$\boldsymbol{C} = (\boldsymbol{J}^{\top}\boldsymbol{D}^{-1}\boldsymbol{J})^{-1}, \tag{2}$$

where $\boldsymbol{J}$ is the Jacobian matrix of f evaluated at $\tilde{\boldsymbol{p}}$.

However, since in our case the depth camera is a Kinect device, which is calibrated using the camera model of [13] for which the evaluation of $\boldsymbol{J}$ is tedious, we simplify the computations by using a simpler diagonal model for the covariance matrix $\boldsymbol{C}$. That is, given the back-projected point $\tilde{\boldsymbol{p}}$ we model its covariance matrix $\boldsymbol{C}$ as a diagonal matrix whose elements depend only on the depth coordinate z.

As shown in [13], the measurement noise causes the variances of coordinates of $\tilde{\boldsymbol{p}}$ to be depth dependent in such a manner that they can be approximated with a quadratic function. We fit a quadratic curve to the reprojection errors to estimate the variance along the optical axis (i.e. in z-direction) [13] so that we get

$$\mathrm{Var}(z) = (\alpha_2 z^2 + \alpha_1 z + \alpha_0)^2 \ , \tag{3}$$

where $\alpha_0 = 0.0032225$, $\alpha_1 = -0.0020925$, and $\alpha_2 = 0.0022078$. Along the image plane, the variance results from pixels back-projecting into rectangles in the real world. The actual position of a point is assumed to have a uniform distribution inside the rectangle of the corresponding pixel. The pixels back-project into rectangles instead of squares due to the camera having different focal lengths along the x- and y-axes. At a distance of 1 m, the rectangles have a width of $\beta_\mathrm{x} = 0.0017228\,\mathrm{m}$ and a height of $\beta_\mathrm{y} = 0.0017092\,\mathrm{m}$. Thus, by utilizing the formula for the variance of a uniform distribution, we get

$$\mathrm{Var}(x) = (\beta_\mathrm{x} z/\sqrt{12})^2 \tag{4}$$

and

$$\mathrm{Var}(y) = (\beta_\mathrm{y} z/\sqrt{12})^2 \ . \tag{5}$$

Possible alignment errors are taken into account by scaling variances along the image plane by λ_1 and variances along the optical axis by λ_2. Each point then has a covariance matrix

$$\boldsymbol{C} = \begin{bmatrix} \lambda_1 \mathrm{Var}(x) & 0 & 0 \\ 0 & \lambda_1 \mathrm{Var}(y) & 0 \\ 0 & 0 & \lambda_2 \mathrm{Var}(z) \end{bmatrix} \tag{6}$$

in the camera reference frame. We used fixed constant values for parameters λ_1 and λ_2 in all our experiments.

Finally, since the points originating from different depth maps are merged into a single nonredundant point cloud in the world coordinate frame, we need to transform the covariance matrices to the world frame as well. Given the rotation matrix $\boldsymbol{R}$ between the world frame and the camera coordinate frame, we may transform the covariance matrix $\boldsymbol{C}$ to the world frame by

$$\tilde{\boldsymbol{C}} = \boldsymbol{R}\boldsymbol{C}\boldsymbol{R}^\top \tag{7}$$

3.3 Point Cloud Refinement

As shown in Algorithm 1, our algorithm proceeds in a sequential manner and, at each iteration, it incrementally adds points from the currently processed depth map to the existing point cloud. Before adding a new point from the current depth map to the point cloud, the point is compared to the existing points of the cloud which originate from views directly connected to the current view and which project to the same pixel in the current view as the new point under consideration. If it turns out that the new point is located very near some existing points, in terms of a Mahalanobis distance, the new point is not added to the cloud but it is used to update the position estimates of the nearby existing points and their covariance matrices.

The refinement process is formulated as a recursive least-squares estimation problem [14]. That is, we assume that each reconstructed surface point has a true location $\boldsymbol{p}$ and each measurement $\tilde{\boldsymbol{p}}$ of this surface point is a noise-corrupted version of the true point, i.e.

$$\tilde{\boldsymbol{p}} = \boldsymbol{p} + \boldsymbol{v}, \tag{8}$$

where $\boldsymbol{v}$ is a random noise vector having a zero-mean Gaussian distribution with a covariance matrix $\tilde{\boldsymbol{C}}$. Further, we assume that the existing, previously added point of the cloud, which is currently under consideration, has a position estimate $\hat{\boldsymbol{p}}$ with covariance matrix $\hat{\boldsymbol{C}}$. This setting is schematically illustrated in Fig. 1, where covariance matrices $\tilde{\boldsymbol{C}}$ and $\hat{\boldsymbol{C}}$ are illustrated by the ellipsoids, which visualize the uncertainty of $\tilde{\boldsymbol{p}}$ and $\hat{\boldsymbol{p}}$.

Before deciding whether the new point $\tilde{\boldsymbol{p}}$ and the existing point $\hat{\boldsymbol{p}}$ are considered as the same surface point, in which case they would be merged by updating $\hat{\boldsymbol{p}}$, we compute the result of the hypothesized merger. That is, we compute the estimate $\hat{\boldsymbol{p}}'$ of the point $\boldsymbol{p}$ by simply assuming that $\tilde{\boldsymbol{p}}$ would be a measurement of the same surface point as our current estimate $\hat{\boldsymbol{p}}$. According to a well-known recursive estimation formula, i.e. using the best linear unbiased estimator (BLUE) of [14, page 130], we get the updated estimate by

$$\hat{\boldsymbol{p}}' = \hat{\boldsymbol{p}} + \hat{\boldsymbol{C}}'\tilde{\boldsymbol{C}}^{-1}(\tilde{\boldsymbol{p}} - \hat{\boldsymbol{p}}), \tag{9}$$

where $\hat{\boldsymbol{C}}'$ is the covariance matrix of the updated estimate and is obtained by

$$\hat{\boldsymbol{C}}' = (\hat{\boldsymbol{C}}^{-1} + \tilde{\boldsymbol{C}}^{-1})^{-1}. \tag{10}$$

Now, given $\hat{\boldsymbol{p}}'$, the result of the merger of $\hat{\boldsymbol{p}}$ and $\tilde{\boldsymbol{p}}$, we compute its Mahalanobis distances d_1 and d_2 to $\hat{\boldsymbol{p}}$ and $\tilde{\boldsymbol{p}}$, respectively, using the corresponding covariance matrices $\hat{\boldsymbol{C}}$ and $\tilde{\boldsymbol{C}}$, i.e.,

$$d_1 = \sqrt{(\hat{\boldsymbol{p}}' - \hat{\boldsymbol{p}})^\top \hat{\boldsymbol{C}}^{-1}(\hat{\boldsymbol{p}}' - \hat{\boldsymbol{p}})} \tag{11}$$

$$d_2 = \sqrt{(\hat{\boldsymbol{p}}' - \tilde{\boldsymbol{p}})^\top \tilde{\boldsymbol{C}}^{-1}(\hat{\boldsymbol{p}}' - \tilde{\boldsymbol{p}})}. \tag{12}$$

If both distances d_1 and d_2 are smaller than a threshold τ we update the existing point in the cloud by setting

$$\hat{\boldsymbol{p}} \leftarrow \hat{\boldsymbol{p}}' \qquad \text{and} \qquad \hat{\boldsymbol{C}} \leftarrow \hat{\boldsymbol{C}}'. \tag{13}$$

This case is illustrated in Fig. 1 where the point $\hat{\boldsymbol{p}}'$ is located within the confidence ellipsoids of both $\hat{\boldsymbol{p}}$ and $\tilde{\boldsymbol{p}}$. In our experiments we used a fixed value $\tau = 3$, as shown in Algorithm 1.

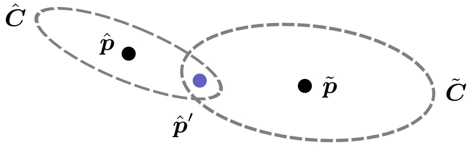


Fig. 1. Two points, $\hat{\boldsymbol{p}}$ and $\tilde{\boldsymbol{p}}$, in the three-dimensional space illustrated with their confidence ellipsoids represented by the covariance matrices $\hat{\boldsymbol{C}}$ and $\tilde{\boldsymbol{C}}$. Point $\hat{\boldsymbol{p}}'$ is the result of the merger of $\hat{\boldsymbol{p}}$ and $\tilde{\boldsymbol{p}}$ when their covariances are properly taken into account in the merging. In this case $\hat{\boldsymbol{p}}'$ is located within the confidence ellipsoids of both $\hat{\boldsymbol{p}}$ and $\tilde{\boldsymbol{p}}$, which means that they can be merged.

Otherwise, if either $d_1 \geq \tau$ or $d_2 \geq \tau$, the existing point $\hat{\boldsymbol{p}}$ is not updated in the cloud. Further, if there does not exist any previously added point in the cloud that would be updated by the point $\tilde{\boldsymbol{p}}$ of the current depth map (according to the aforementioned rules), we add a new point to the cloud and initialize its position and covariance estimates by setting $\hat{\boldsymbol{p}} \leftarrow \tilde{\boldsymbol{p}}$ and $\hat{\boldsymbol{C}} \leftarrow \tilde{\boldsymbol{C}}$. In fact, this kind of initialization directly follows from (9), (10) and (13) if we first set $\hat{\boldsymbol{C}}^{-1} = 0$, which corresponds to the case of a completely unknown point (which is maximally uncertain).

Finally, in addition to position and its covariance, we have a color value for each back-projected measurement point (obtained from the color image associated with the depth map). During the merging process we also refine the colors of the points at each update. For this, a running total of color values is kept for each point. When a refined point is created, its total is set to the sum of the totals of the existing point and new measurement. After all views have been processed, the final color of a point is the average, i.e., the running total divided by the number of color values it includes.

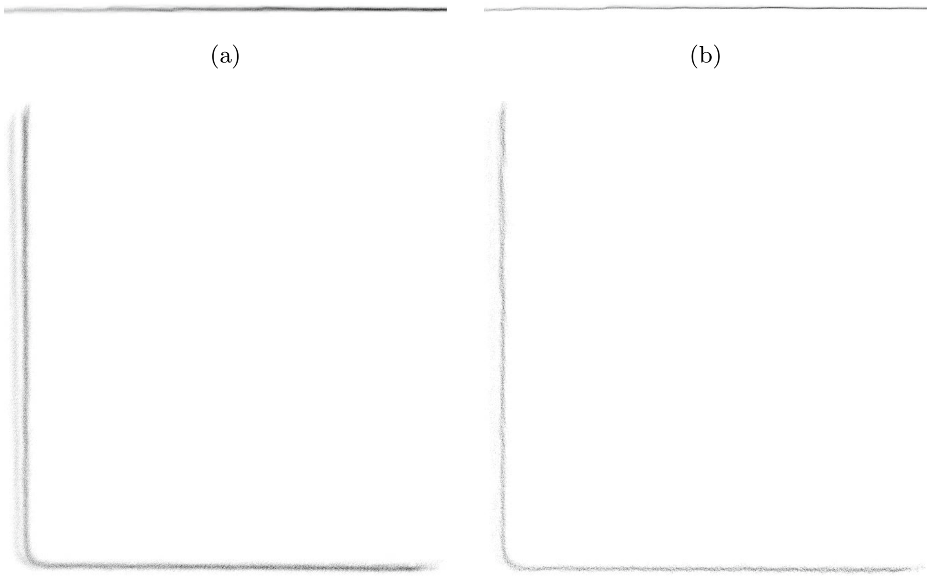
4 Results

The results were evaluated by creating point clouds of a whiteboard and a three-sided cube and by fitting a plane and a cube to the clouds, respectively. In each case the models were fitted to a redundant cloud including all the original points and to a simplified cloud output by the algorithm. The measurement variances were scaled with $\lambda_1 = 40$ and $\lambda_2 = 20$. The algorithm improved the residuals considerably as can be seen in Table 1. Slice images of the whiteboard and cube data sets before and after processing are shown in Fig. 2.

The performance of our implementation was evaluated on a 2 GHz Intel Core 2 processor with the whiteboard and cube data sets and an office data set. The results are listed in Table 2. The point clouds of the data sets are visualized in Figs. 3–5. These visualizations show that the proposed approach is able to significantly reduce the redundancy of point clouds without reducing the coverage. Moreover, Figs. 2 and 5 confirm that our approach also improves the accuracy of modeled surfaces by successfully fusing multiple measurements.

Table 1. Fitting results

Data set	Residual standard deviation (m)	Maximum residual (m)
whiteboard (redundant)	0.0033	0.0192
whiteboard (simplified)	0.0023	0.0184
cube (redundant)	0.0018	0.0074
cube (simplified)	0.0011	0.0068



(a)

(b)

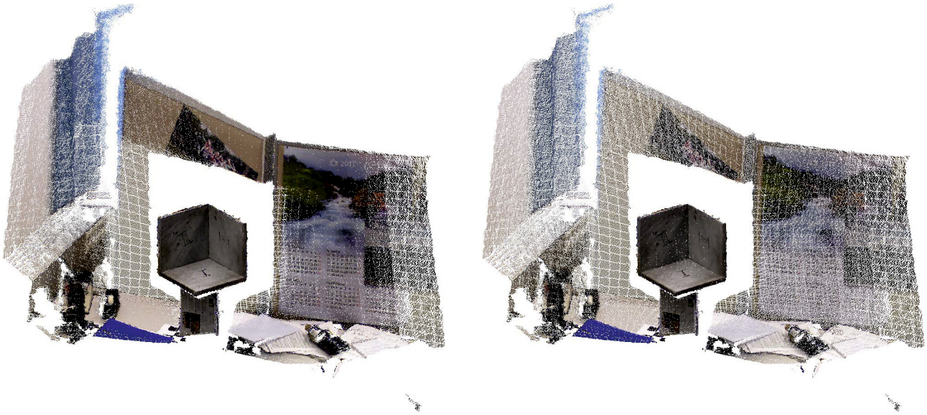
(c)

(d)

Fig. 2. Top-down slices of (a) the whiteboard point cloud before running the algorithm and (b) after running the algorithm and of (c) the three-sided cube point cloud before and (d) after. The whiteboard slice has a width of 1 meter and the cube has a side length of 20 centimeters. The input images for the cube had some misalignment, which created two parallel walls. Still, the algorithm managed to merge the points into a single wall and reduce the noise making the walls thinner. Also, the resulting corner is no smoother than in the input.

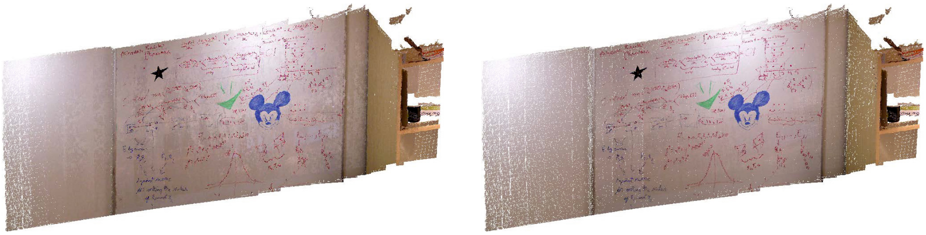
Table 2. Performance results

Data set	cube	whiteboard	office
View count	4	15	22
Point count before simplification	977 701	3 883 839	5 315 546
Point count after simplification	437 483	990 561	823 659
Ratio of reduction	55 %	74 %	85 %
Mean merging time per view (s)	0.16	0.19	0.21
Peak memory usage (MiB)	181	622	809



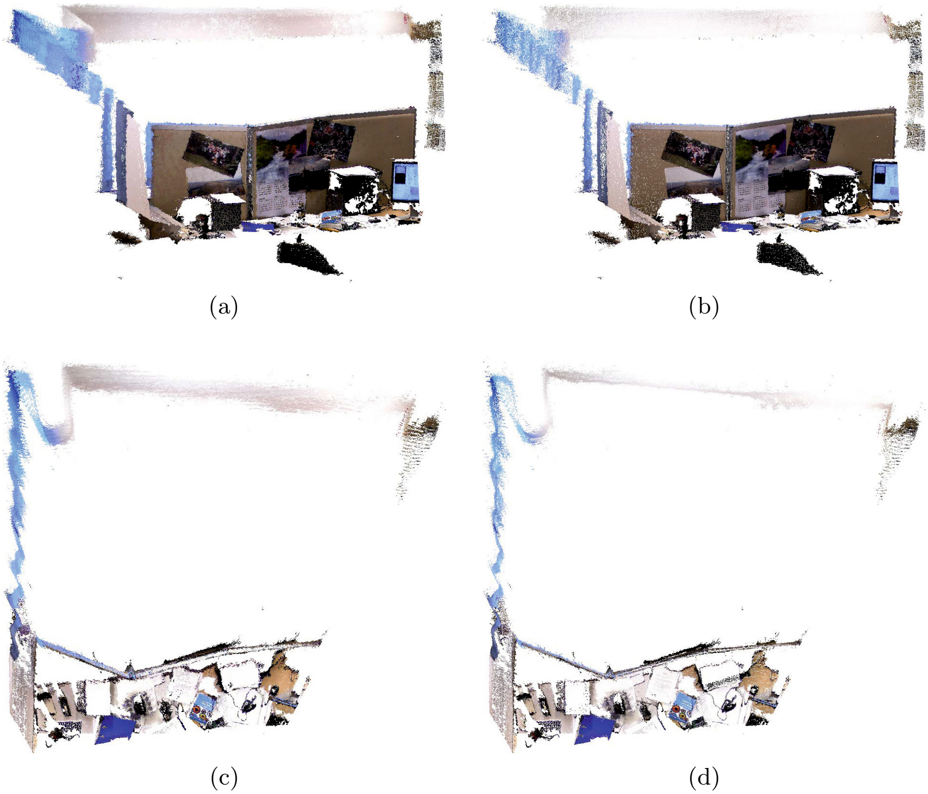
(a) (b)

Fig. 3. The cube data set as (a) an unprocessed redundant point cloud and (b) a simplified point cloud. The point count was reduced from 977 701 to 437 483 (reduced by 55 %).



(a) (b)

Fig. 4. The whiteboard data set as (a) an unprocessed redundant point cloud and (b) a simplified point cloud. The point count was reduced from 3 883 839 to 990 561 (reduced by 74 %).



(a)

(b)

(c)

(d)

Fig. 5. The office data set as (a) an unprocessed redundant point cloud and (b) a simplified point cloud. The (c) top-down view of the redundant cloud and (d) top-down view of the simplified cloud demonstrate how the the walls become thinner after running the algorithm. The point count was reduced from 5 315 546 to 823 659 (reduced by 85 %).

5 Discussion

As shown in Algorithm 1 and explained below, our approach is scalable to large amounts of data in terms of both memory and computational cost. Hence, it can be applied to long image sequences of large environments.

First, the memory cost of the algorithm depends on the number of points added to the cloud for each new view. Thus, it depends on the amount of novel content in each view. If there is little movement between views, most of the new measurements are used to refine existing points instead of adding new points.

Second, the computational cost of adding a new view is bounded by the number of connected views it has. This is the result of only projecting points from connected views. Two views are considered connected if they might contain common points which should be merged. When processing long sequences of large environments, a view usually has only few connected views.

Further, considering possibilities for future improvements, we would like to note that capturing thin objects from both sides may be problematic using our algorithm as the two sides might get merged together when refining points. This issue could be fixed by introducing surface normal information to the points and not merging points whose normals differ too much. This can be accomplished by using a surfel cloud [12] instead of a point cloud as was done by Henry et. al [11] or by simply assigning a surface normal vector for each point as in e.g. [17].

Finally, the performance of the algorithm could be improved by offloading some processing to the GPU and further optimizing the implementation. For example, point projection is well suited for such a parallelizing optimization, and it was employed by Merrell et al. [5].

6 Conclusion

A new method has been presented for creating a nonredundant point cloud from overlapping depth maps. This point cloud simplification significantly reduces the number of points compared to an unprocessed point cloud. This in turn results in lower memory and computational cost in later stages in a processing pipeline. Additionally, the points more precisely represent the true surface since they are estimates from multiple nearby measurements. In particular, multiple measurements of the same surface location are combined in a statistically justified manner, unlike in many previous approaches.

References

1. Moenning, C., Dodgson, N.: Intrinsic point cloud simplification. In: Proc. 14th GraphiCon., vol. 14 (2004)
2. Song, H., Feng, H.: A global clustering approach to point cloud simplification with a specified data reduction ratio. Computer-Aided Design 40(3), 281–292 (2008)
3. Yu, Z., Wong, H., Peng, H., Ma, Q.: ASM: An adaptive simplification method for 3D point-based models. Computer-Aided Design 42(7), 598–612 (2010)
4. Shi, B., Liang, J., Liu, Q.: Adaptive simplification of point cloud using k-means clustering. Computer-Aided Design 43(8), 910–922 (2011)
5. Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J., Yang, R., Nistér, D., Pollefeys, M.: Real-time visibility-based fusion of depth maps. In: IEEE 11th International Conference on Computer Vision (ICCV), pp. 1–8 (2007)
6. Labatut, P., Pons, J.P., Keriven, R.: Robust and efficient surface reconstruction from range data. Comput. Graph. Forum 28(8), 2275–2290 (2009)
7. Newcombe, R., Davison, A., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 127–136 (2011)
8. Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., McDonald, J.: Kintinuous: Spatially Extended KinectFusion. Technical report, MIT CSAIL (2012)
9. Roth, H., Vona, M.: Moving Volume KinectFusion. In: British Machine Vision Conference (BMVC) (2012)

10. Heredia, F., Favier, R.: Kinect Fusion extensions to large scale environments (2012) (accessed November 19, 2012)
11. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In: The 12th International Symposium on Experimental Robotics (ISER), vol. 20, pp. 22–25 (2010)
12. Habbecke, M., Kobbelt, L.: A surface-growing approach to multi-view stereo reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
13. Herrera C.,D., Kannala, J., Heikkilä, J.: Joint depth and color camera calibration with distortion correction. IEEE Transactions on Pattern Analysis and Machine Intelligence (2012)
14. Mendel, J.: Lessons in Estimation Theory for Signal Processing, Communications, and Control. Prentice Hall (1995)
15. Jancosek, M., Shekhovtsov, A., Pajdla, T.: Scalable multi-view stereo. In: IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1526–1533 (2009)
16. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge University Press (2000)
17. Ylimäki, M., Kannala, J., Holappa, J., Heikkilä, J., Brandt, S.: Robust and accurate multi-view reconstruction by prioritized matching. In: International Conference on Pattern Recognition (2012)