Progressive 3D Modeling All the Way

Alex Locher[1] Michal Havlena[1] Luc Van Gool[1,2]

[1] Computer Vision Laboratory, ETH Zurich, Switzerland [2] VISICS, KU Leuven, Belgium

Abstract

This work proposes a method bridging the existing gap between progressive sparse 3D reconstruction (incremental Structure from Motion) and progressive point based dense 3D reconstruction (Multi-View Stereo). The presented algorithm is capable of adapting an existing dense 3D model to changes such as the addition or removal of new images, the merge of scene parts, or changes in the underlying camera calibration. The existing 3D model is transformed as consistently as possible and the structure is reused as much as possible without sacrificing the accuracy and/or completeness of the final result. A significant decrease in runtime is achieved compared to the re-computation of a new dense point cloud from scratch. We demonstrate the performance of the algorithm in various experiments on publicly available datasets of different sizes and compare it to the baseline. The work interacts seamlessly with publicly available software enabling an integrated progressive 3D modeling pipeline.

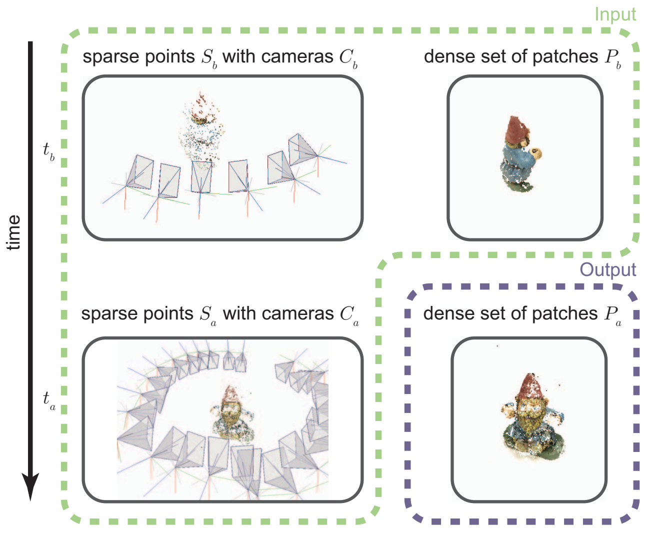
Input
sparse points S_b with cameras C_b
dense set of patches P_b
t_b
time
Output
sparse points S_a with cameras C_a
dense set of patches P_a
t_a

Figure 1. The proposed algorithm converts a dense point cloud P_b from time t_b into a dense point cloud P_a corresponding to the updated sparse model from time t_a.

1. Introduction

3D reconstruction from single photographs taken at different viewpoints is a long studied topic in computer vision. Many algorithms have been proposed and are able to reconstruct scenes up to very large scale of 100 Mio images [4]. While Structure-from-Motion (SfM) is usually used to calibrate individual cameras and create a sparse point cloud, Multi-View Stereo (MVS) algorithms use the output of SfM and estimate a much denser pointcloud of the same scene.

With the availability of mobile devices, suddenly everybody carrying a smart phone is a potential user of a 3D scanning technology. In this newly created scenario, users take pictures of a scene and interactively reconstruct a 3D model on-the-fly. While small objects can be handled by the limited processing power of the phone itself [14], larger scenes have to be offloaded to a remote server [9]. A key aspect in such a user-centric scenario is to provide the user with an immediate feedback on the status and quality of the 3D reconstruction process. If the user detects that a certain part of the scene is not yet well covered, she can take additional pictures in order to get an as complete model as possible.

In order to satisfy the demands of such or similar scenarios, reconstruction algorithms need to be able to incorporate user information on-the-fly and deliver progressive and multi-scale 3D output. While incremental SfM can incorporate new images into an existing 3D model and provide an updated output soon after [16], current MVS algorithms cannot handle changes in the input calibration. Even the recently published Progressive Prioritized Multi-view Stereo (HPMVS) algorithm [10], which can deliver a progressive multiscale output, cannot handle changes in input data. As a result, a modification of the sparse SfM model requires the execution of the MVS algorithm from scratch and an already computed dense model is completely invalidated. This contradicts the progressive scenario, in which a user should be able to add or remove images on-the-fly and the resulting changes in the sparse model should be propagated to the dense model. Additionally, the overall processing time for a 3D reconstruction, starting from the acquisition of the images up to a dense point cloud, can be reduced with a progressive 3D modeling pipeline as the densification stage

978-1-5090-5407-7/16 $31.00 © 2016 IEEE
DOI 10.1109/3DV.2016.11

can be started in parallel to the sparse reconstruction.

We therefore propose a progressive MVS algorithm capable of incorporating the changes in the sparse reconstruction into an already existing dense 3D point cloud.

1.1. Related Work

PMVS published by Furukawa et al. [2, 3] is one of the most prominent MVS algorithms. It is a patch-based algorithm capable of densely reconstructing a wide range of different scene types starting from a sparse 3D model. The algorithm works purely in batch mode and therefore is neither progressive nor adaptive to changing calibration. The recently published HPMVS [10] can be seen as a progressive version of PMVS. HPMVS can progressively deliver a dense 3D point cloud which gets more detailed the longer the algorithm runs. The input of HPMVS is considered to be static and therefore the algorithm cannot handle changes in the sparse configuration and can also not make use of new images added to the 3D model.

Hoppe et al. [5] presented a method directly reconstructing a surface mesh out of the sparse point cloud, omitting the dense reconstruction step completely. While this leads to less detailed results, the computation is fast. The method is designed to deal with incremental information, however only the addition of new information is allowed. The algorithm cannot propagate the changes in the calibrations of current cameras to the existing structure. The algorithm presented by Sugiura et al. [13] is quite close to Hoppe et al. and is capable of creating an incremental surface and incorporating new points as well as cameras. The method is based on tetrahedra carving and works directly on the input points. Its output is therefore not as detailed as the one of an MVS algorithm. In addition, the algorithm cannot integrate changes of the camera calibration into the existing 3D model.

Roters et al. [11] presented an incremental MVS algorithm which progressively refines an existing point cloud by interpolating points in the centers of the existing triangles. Similarly to HPMVS, the algorithm cannot handle additional information and/or changes in the underlying 3D model. Yu et al. [19] wholly avoid the densification step and create a dense mesh directly out of the sparse SfM points. The method is capable of integrating new information into the existing model, but the change and/or removal of existing cameras is not explicitly handled. Furthermore, the sparse representation leads to a less detailed model representation compared to MVS methods.

Whelan et al. [17] presented a volumetric approach for the 3D reconstruction of scenes from RGB-D images. While this work is not directly related (SfM does not rely on depth images), it has some interesting aspects. In loop closures, the method performs a fusion of the overlapping mesh by estimating an *as rigid as possible* transformation. Camera poses are transformed in the pose graph, where the structure is deformed based on a mesh transformation. While the proposed algorithm contains some similar ideas, the work of Whelan et al. is not designed for MVS and the applied deformation relies on the availability of a mesh. In addition, only changes of camera poses (extrinsics) are handled in the presented loop closing event. Therefore the method is not suitable for the target application of the proposed work.

As shown in this section, quite some work has already incorporated progressive aspects in 3D reconstruction and some of them are also able to partially incorporate additional information into an existing 3D model. But to the best of our knowledge, none of it fully addressed the integrated progressive pipeline within the scope of SfM and MVS.

1.2. Contribution

This work presents a method bridging the gap between existing progressive SfM and MVS algorithms. The proposed method is able to incorporate changes in the underlying calibration as well as to include additional images which enables to have an integrated progressive 3D modeling pipeline starting from the input images up to the generated dense 3D point cloud. Model regions with a need for an extension and/or annealing are identified and processed only locally which brings large savings in runtime compared to the global method building the dense 3D model from scratch. We demonstrate the effectiveness of the proposed algorithm on a sample application and verify its output in several experiments.

An open-source implementation of the proposed algorithm is available at: `https://github.com/alexlocher/patw`

2. Prerequisites

The presented algorithm makes use of an open source algorithm HPMVS [10] and interacts seamlessly with it. In HPMVS, individual 3D points are internally organized in a hierarchical octree structure and a processing loop repeatedly densifies the point cloud in an expansion procedure and increases the resolution in a branching step. For the proposed algorithm, HPMVS can be treated as a blackbox taking 3D points as an input and progressively outputting a densified 3D point cloud. We therefore refer the reader to the original publication [10] for further details.

3. Progressive 3D Modeling

Next, an overview of the proposed method is given and the individual stages of the method are detailed.

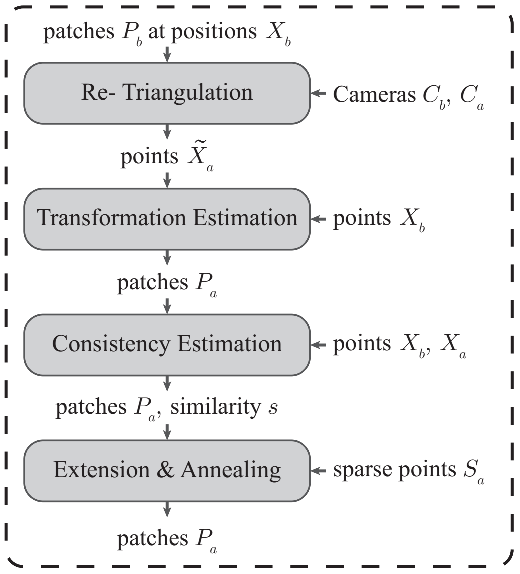

Figure 2. Single steps of the proposed algorithm and the most relevant inputs and outputs of the individual stages.

3.1. Overview

In brief, the proposed progressive reconstruction method takes a dense set of surface patches and a set of calibrated cameras and poses from time t_b (before) as well as the camera calibrations and poses and a sparse point cloud from time t_a (after) as an input and outputs the adjusted dense point cloud at time t_a. The existing 3D geometry is transformed, extended, and annealed in a way to come as close as possible to the result of re-running the MVS stage from scratch. Figure 1 illustrates the algorithm's input and output parameters.

The algorithm consists of four main steps, see Figure 2. In the first step, input patches P_b are re-triangulated using the old and the new camera calibrations C_b and C_a. In order to maximize the reuse of the existing 3D geometry, an "as consistent as possible" transformation is recursively estimated using an octree structure. To avoid ending up with an inconsistent 3D structure, a per point similarity measure is evaluated and problematic scene parts are identified. In the last stage, new sparse points as well as new images are integrated into the model and the original progressive multi-view stereo algorithm extends and anneals the model into the final dense set of patches P_a. Even though the algorithm is designed for dense point clouds, we also demonstrate the possibility of adapting an existing triangular mesh to a modified camera calibration.

3.2. Model Representation

A surface patch p consists of a 3D coordinate $\mathbf{X}$, normal $\mathbf{n}$, scale p_s, and visibility information $V(p)$, where $V(p)$ consists of a list of cameras the patch is visible in. One of the visible cameras in $V(p)$ is set as the reference camera $R(p)$. Each calibrated camera C_i defines a projective function π_i mapping a 3D point $\mathbf{X}$ to a corresponding pixel coordinate $\mathbf{x}_i$. Patches are spatially organized in a dynamic octree T with cells N, each having a width of w_N.

3.3. Re- Triangulation

First, all the points of the old model are re-triangulated using the new camera parameters. Every point $\mathbf{X}_b$ is projected into the image space of the visible images $V(p)$ of the corresponding patch. The resulting correspondences in the image space $\{\mathbf{x}_i | i \in V(p)\}$ can be assumed to be invariant to changes of the intrinsic and extrinsic camera parameters. $\tilde{\mathbf{X}}_a$ is obtained by triangulating the image correspondences $\{\mathbf{x}_i\}$ using the new calibration parameters of C_a. Due to the presence of noise in the old model, the old calibration, and the new calibration, the resulting 3D point $\tilde{\mathbf{X}}_a$ might be noisy as well. Therefore we re-project $\tilde{\mathbf{X}}_a$ into the visible images and remove points with high re-projection error δ, or more formally:

$$P_a = \left\{ p \,\middle|\, \frac{1}{|V(p)|} \sum_{i \in V(p)} \left\| \mathbf{x}_i - \pi_i(\tilde{\mathbf{X}}_a) \right\| \, < \, \delta_{max} \right\} \quad (1)$$

3.4. Transformation Estimation

Due to small local variations, the constellation of neighboring points can differ a lot between the original points X_b and the triangulated points $\tilde{X}_a$. These differences cause problems to a consistent transformation of the patches' normals and can even lead to overlapping triangles if the patches are vertices of a triangular mesh. In order to get a locally consistent point cloud and reduce the computational cost for the later annealing stage to a minimum, while maintaining the re-triangulated shape, we recursively estimate a 7 DoF similarity transformation field $\mathfrak{T}(x, y, z) \in \mathrm{Sim}(3)$ using an octree structure and point to point correspondences from X_b to $\tilde{X}_a$. Each transformation $\mathcal{T}$ has a rotation $\mathbf{R}$, translation $\mathbf{t}$, and scale s assigned (Equation 2). A 3D similarity transform can be efficiently estimated from at least three 3D to 3D correspondences by Singular Value Decomposition (SVD) of a self-adjoint 4×4 matrix [6].

$$\mathcal{T} = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & s^{-1} \end{array} \right) \qquad \mathbf{R} \in \mathrm{SO}(3)\,,\ \mathbf{t} \in \mathbb{R}^3\,,\ s \in \mathbb{R} \quad (2)$$

Instead of the minimum amount of three correspondences, we estimate a similarity transform from all correspondences within a certain cell N in the octree T_b of the patches P_b.[1] The Euclidean distance $e_{\mathcal{T}}$ between the points obtained by

[1]This is significantly faster than a robust RANSAC estimation and potential outliers are still handled in the next step.

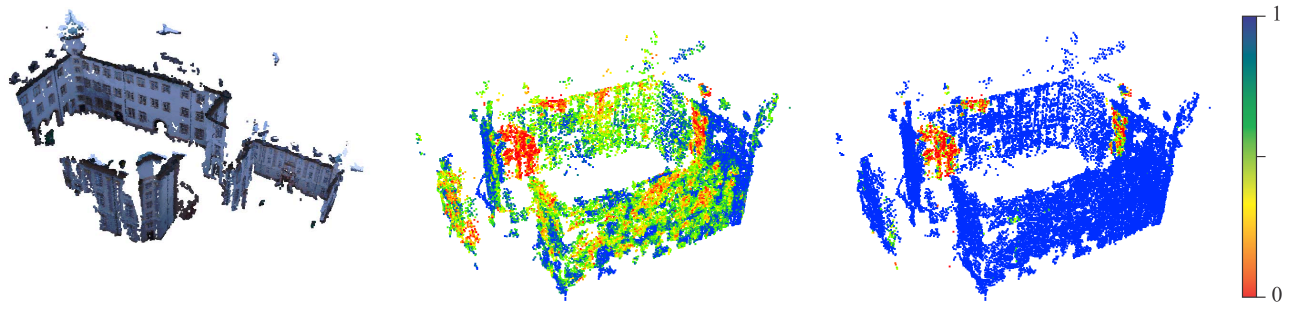

Figure 3. The color coding shows the similarity measure $\mathfrak{s}$ of individual points (blue color indicates that the neighborhood is consistent, whereas the constellation of neighboring points changed a lot in the red regions). Naive re-triangulation (center) of the patches P_b (left) leads to inconsistent neighborhoods, which is problematic especially for meshed pointclouds. Instead, we recursively estimate a consistent transformation among the points (right). The remaining inconsistencies are resolved in the subsequent extension and annealing step.

applying transformation $\mathcal{T}^N$ to X_b^N and the triangulated points $\tilde{X}_a^N$ serves as an error criterion. To find the right cell resolution for estimating the transformation in different parts of the model, the eight subcells of a given root cell of the octree are recursively traversed until $e_{\mathcal{T}}$ is below the threshold $\lambda_{\mathcal{T}}$ or the amount of points within the cell N is smaller than n_{min}.

$$e_{\mathcal{T}} = \frac{1}{\left|X_b^N\right|} \sum_{j \in X_b^N} \left\| \mathcal{T}^N(\mathbf{X}_b^j) - \tilde{\mathbf{X}}_a^j \right\| \tag{3}$$

Algorithm 1 summarizes the transformation estimation. After the recursive transformation estimation, every patch has a corresponding $\mathcal{T}^N$ and the final positions X_a and normals n_a are set by applying the individual transformation. Figure 3 shows a color coded similarity (Equation 4) of neighboring points of a transformed model with directly triangulated points (middle) and with the proposed transformation method (right). Note that points obtained by the transformation represent the same scene, but have a much more consistent neighborhood.

```
Data: List of points and correspondences from cell $N$
Result: transformation $\mathcal{T}$ for all patches in $N$.
Function estimateTN ($N$)
    if $\left|X_b^N\right| > n_{min}$ then
        $\mathcal{T}^N \leftarrow estT(X_b^N, \tilde{X}_a^N)$
        calculate $e_{\mathcal{T}}$ (Equation 3)
        if $e_{\mathcal{T}} > \lambda_{\mathcal{T}}$ then
            forall the $N_c \in$ children($N$) do
                estimateTN ($N_c$)
            end
        end
    end
    return
end
```

Algorithm 1: A similarity transformation $\mathcal{T}$ between patches is recursively estimated within individual cells N of an octree T.

3.5. Detection of Problematic Regions

Even if we transform patches by a locally consistent similarity transformation, changes of the structure in the neighborhood of some of the patches are unavoidable. For example in the case of a loop closure, two completely separate parts of the model will be merged into a single area. In order to handle such regions properly, the algorithm needs to detect and identify areas subject to changes. We therefore compare the local neighbors of every patch in P_a to the neighbors of the corresponding patch in P_b and introduce an effective similarity measure $\mathfrak{s}$ which counts the number of identical neighbors.

$$\mathfrak{s}_j = \frac{\left|\mathrm{kNN}(P_b^j) \cap \mathrm{kNN}(P_a^j)\right|}{k} \qquad \forall j \in 1 \dots |P_a| \tag{4}$$

To maintain large scale capabilities, nearest neighbor search is efficiently solved by an octree structure on patch's locations $\mathbf{X}_b$ and $\mathbf{X}_a$. Figure 3 (right) shows the color coded similarity measure on an artificially created loop closure for real data. Note that the detected inconsistencies are resolved in the following processing step.

3.6. Extension & Annealing

In the next step, eventual new data of the sparse point cloud S_a are integrated with existing patches P_a organized in a dynamic octree T_a. The octree cell level is chosen according to the patch's scale such that $w_N \geq p_s > \frac{w_N}{2}$. New patches are initialized from S_a if the target cell in the octree is not yet occupied. Besides new points, newly added

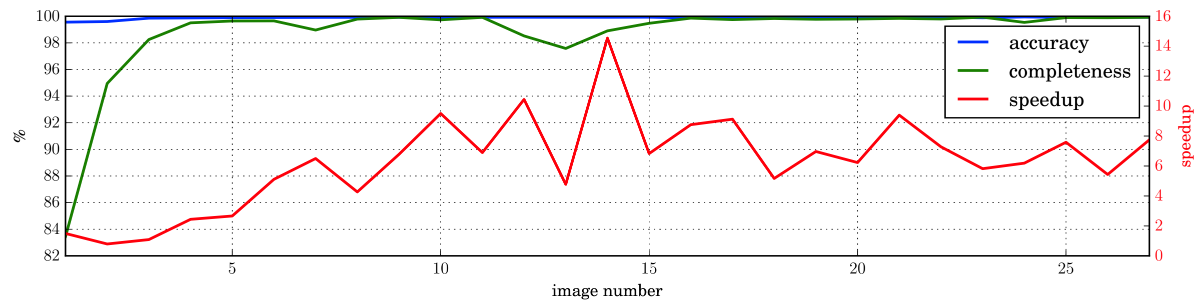

Figure 4. Comparison of the proposed algorithm against running the HPMVS algorithm from scratch on the Castle-P30 dataset. Note that a significant decrease in runtime is achieved without scarifying the quality and/or accuracy of the final model.

C_{added} and removed $C_{removed}$ cameras and their images have to be incorporated into the new model as well. We flag a patch as *dirty* if:

- the patch was visible from a removed camera and had less than λ_1 visible images
- the reference image $R(p)$ was removed
- the patch is within the field of view of a new camera and has less than λ_2 visible images

All cells of the octree T_a fulfilling one of the following conditions are added to a processing queue:

- the cell contains multiple patches
- the cell contains a patch freshly initialized from S_a
- the cell contains a patch flagged as *dirty*
- the cell contains a patch with $\mathfrak{s} < \lambda_{\mathfrak{s}}$

All cells in the processing queue are processed by the Progressive Multi-View Stereo algorithm presented by Locher et al. [10] using all cameras C_a. Positions and normals of the patches are optimized and the expansion procedure of HPMVS assures that the model is smoothly annealed and also grown into the extra regions covered by the newly added cameras C_{added}.

3.7. A Word on Meshes

Even though the presented method is designed for surface patches, certain meshes can be handled by a similar procedure as well. If the meshing algorithm is able to remesh individual sub parts, the generated output of the proposed algorithm can be used to transform an existing mesh. Typically, individual vertices of a mesh do not contain camera visibility information. Therefore we perform a nearest neighbor search on the corresponding dense patch set to obtain visibility information for each vertex. The proposed algorithm is then executed with vertices instead of patches. Afterwards, all triangles connected to vertices deleted in the triangulation step or marked as *dirty* are removed. Finally, the remaining mesh is regenerated and possibly extended to patches newly created in the extension and annealing step. We demonstrate this capability using the ball-pivoting [1] algorithm in Section 4.5.

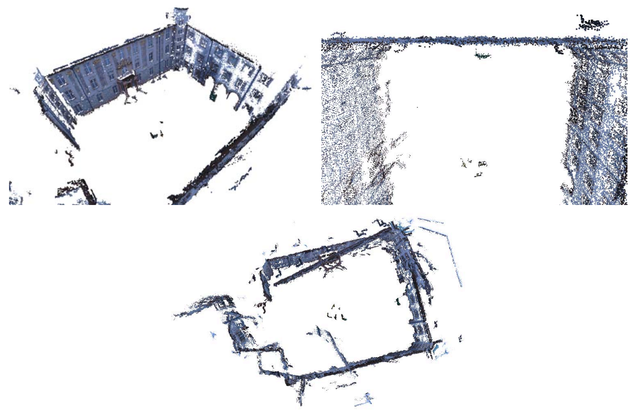

Figure 5. Final output for the successfully merged submodels from Figure 3 (top left) and a closeup of the seamlessly merged wall (top right). A failure case for a badly chosen $\lambda_{\mathcal{T}}$ $(= 0.1 \cdot \rho)$ where the individual parts of the model are merged wrongly (bottom).

4. Experiments and Results

We designed several experiments to show the power and accuracy of the proposed algorithm and tested it on several publicly available datasets. The following parameters were used for all the experiments if not mentioned otherwise: $\lambda_1 = 3, \lambda_2 = 5, \lambda_{\mathfrak{s}} = 0.5, \delta_{max} = 10.0, n_{min} = 10, k = 4$.

4.1. Comparison to Baseline

To show the effectiveness of the proposed method, we directly compare runtime, accuracy, and completeness of the algorithm's output to the dense 3D model obtained by running HPMVS on the new sparse model from scratch. We used VisualSFM [18] to incrementally reconstruct the Castle-P30 dataset [12] starting from just three images and

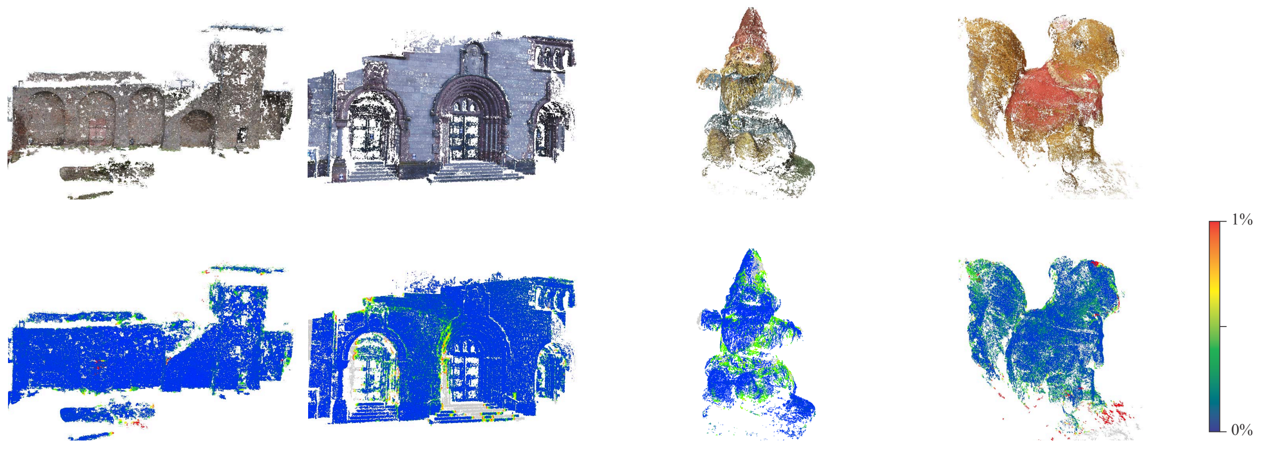

Figure 6. Visualization of the final model (top) and the distance to baseline (bottom) for the different datasets.

adding more images sequentially, resulting in 27 consecutive sparse models. As a baseline we run HPMVS on every intermediate sparse reconstruction from scratch. Our method was then applied between two consecutive models and its result is compared to the model obtained by the baseline method. The three evaluated measures accuracy, completeness, and speed-up are defined as follows:

Accuracy
We report the RMS error between the point cloud obtained by our algorithm X_a and the baseline result B as an accuracy measure. We normalize the obtained distance by the diagonal of the baseline model's bounding box ρ.

$$A = \frac{1}{|X_a| \cdot \rho} \sum_{j \in 1 \ldots |X_a|} \|\mathbf{X}_a^j - \text{NN}(\mathbf{X}_a^j, B)\| \tag{5}$$

Completeness
We count the number of points in the baseline model which are covered by a point in the output of the proposed algorithm. The threshold d is set to 0.1% of the model's bounding box ρ.

$$C = \frac{\left|\{\mathbf{X} | \mathbf{X} \in B \wedge d > \|\mathbf{X} - \text{NN}(\mathbf{X}, X_a)\|\}\right|}{|B|} \tag{6}$$

Speed-up
The execution time of the proposed algorithm with regard to the runtime of the baseline algorithm is of a large importance. Therefore we report the speed-up S as a ratio between the two timings as a third measure.

$$S = \frac{\text{runtime}_{\text{ HPMVS}}}{\text{runtime}_{\text{ proposed}}} \tag{7}$$

In Figure 4 we plot the three measures on all consecutive pairs of images in the Castle-P30 dataset. While the accuracy is constantly high, the completeness and speed-up

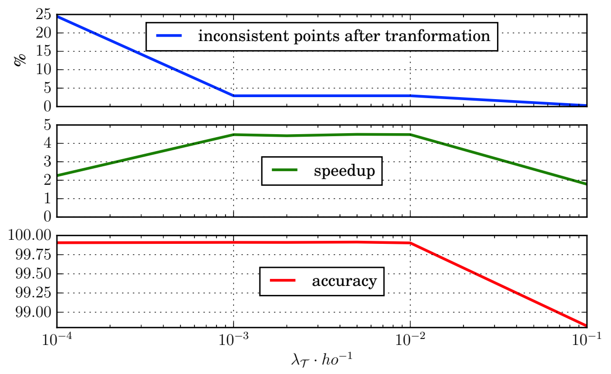

Figure 7. Variation of $\lambda_{\mathcal{T}}$ and the corresponding inconsistent points ($\mathfrak{s} < 0.5$), speed-up, and accuracy on the loop-closing for the Castle-P30 dataset.

measures clearly show that the proposed method is more effective if the model has a certain size already. This can also be intuitively justified, as small models often change a lot if additional images are added whereas in larger models only a small part is affected. Note that the speed-up curve is still subject to some variations. This can be explained by the fact that not every image contains the same amount of new information and therefore the time needed by the annealing stage varies as well.

4.2. Dealing with Large Changes

While incremental scene updates usually only cause small changes in the model, loop closures with a global bundle adjustment in SfM can cause very large changes distributed over the whole model and all cameras. We therefore investigated the behavior of the proposed algorithm in a scenario where two parts of a common model are merged into a single one and refined through a global bundle adjustment step. Figure 3 (left) shows the created scene with

dataset	# images	# points	total time	accuracy	completeness	speed-up
Fountain-P11 [12]	11	760k	66s	99.9%	99.7%	6.93
HerzJesu-P8 [12]	8	700k	214s	99.8%	96.2%	2.51
Castle-P30 [12]	29	1.1M	86s	99.9%	99.9%	7.75
Citywall [15]	542	3.3M	860s	99.9%	99.7%	4.86
squirrel [7]	23	200k	44s	99.8%	96.4%	2.81
gnome [7]	25	150k	52s	99.4%	98.5%	1.42

Table 1. Timings and other measures for different publicly available datasets. Accuracy, completeness, and speed-up are measured relatively to the baseline running HPMVS from scratch.

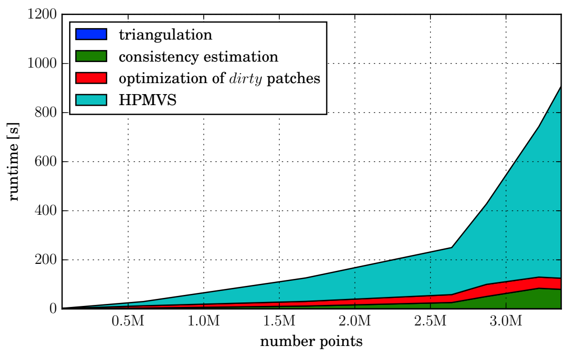

Figure 8. Runtimes for the individual stages of the algorithm when varying the number input points on the Citywall dataset.

the two submodels. In Figure 3 (right) the merged model before the extension and annealing with color coded consistency is shown and Figure 5 (top) shows the final output of the algorithm as well as a closeup of the merged wall. The algorithm successfully merged the two submodels and aligned them seamlessly.

4.3. Recursive Transformation Estimation

The recursive transformation estimation eliminates small variations in the local neighborhoods of points. An inconsistent neighborhood between $\mathbf{X}_b$ and $\mathbf{X}_a$ leads to more effort in the annealing stage and therefore would reduce the speed-up. The maximum allowed mean error per cell $\lambda_\mathcal{T}$ between the re-triangulated $\tilde{\mathbf{X}}_a$ and the transformed $\mathbf{X}_a$ patch positions acts as the main parameter to balance between runtime and accuracy. We therefore report the algorithm's runtime and accuracy A with respect to $\lambda_\mathcal{T}$. Figure 7 shows the corresponding plot on the loop closing scene in the Castle-P30 dataset. A very strict $\lambda_\mathcal{T}$ (e.g. 0.0001 of the bounding box ρ) leads to more than 25% of inconsistent neighbors and also the runtime of the algorithm is suboptimal. If $\lambda_\mathcal{T}$ is set too loosely, the algorithm's accuracy suffers. The region between $\lambda_\mathcal{T} = 0.01 \cdot \rho$ and $\lambda_\mathcal{T} = 0.001 \cdot \rho$ has shown to be optimal for different datasets.

4.4. Scalability

Dense models often consist of a very large amount of patches. It is therefore quite important to analyze the scalability of the proposed algorithm with regard to the number of input points. Figure 8 reports the timings of the individual stages of the algorithm on the Citywall dataset with 564 images [15]. In the experiment we used a sparse reconstruction with 90% of the images as the before model S_b and a complete reconstruction followed by a global bundle adjustment as S_a. The number of points is varied by lowering the final resolution of the HPMVS algorithm by a factor of 2^n. The plot shows that the major part of the runtime is spent on the original HPMVS method, including the time spent on adding new points from the sparse model. The actual conversion of the existing model, including the triangulation, consistency estimation, and the optimization of the *dirty* patches takes significantly less time and scales well even for large models.

Table 1 shows timings of the algorithm for the different datasets. All datasets were incrementally reconstructed using VisualSFM [18] and the numbers reported are for the update between the second-last and the last models. See Figure 6 for some of the reconstructed models and the visualization of their accuracies.

4.5. Meshes

We demonstrate the capability of the algorithm to process meshes on the Foutain-P11 dataset (see Figure 9). A dense model created out of 8 input images using HPMVS is meshed using the Poisson surface reconstruction [8]. Due to the meshing procedure, vertices of the resulting mesh are no longer identical to the patch positions in the original point cloud. As proposed in Section 3.7, we extract the visibility information for the vertices using a nearest neighbor search to the dense point cloud. The vertices are then treated as patches and transformed using the proposed algorithm, see Figure 9c. Triangles touching *dirty* vertices are colored red. The annealed point cloud still contains the same vertices and represents a partial mesh which can then be extended to the new vertices using e.g. the ball-pivoting algorithm [1].

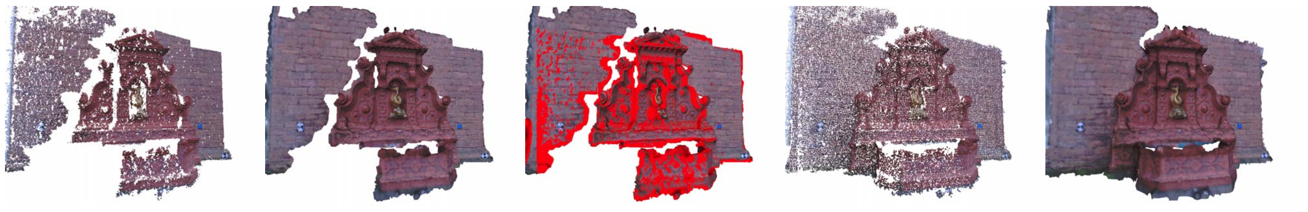
(a) points before (b) mesh before (c) triangles removed (d) points after (e) mesh after

Figure 9. The proposed algorithm can also be used to process vertices of a triangular mesh. Visibility information for individual vertices is extracted by a nearest neighbor search on the corresponding dense point cloud. Triangles touching *dirty* vertices are removed and the resulting partial mesh is extended to the newly added points.

5. Conclusions

We presented an algorithm bridging the gap between progressive SfM and progressive MVS, enabling the full potential of a progressive 3D reconstruction pipeline from the image acquisition up to the dense 3D pointcloud. The algorithm takes an existing dense 3D pointcloud and an updated sparse 3D reconstruction as an input and delivers the adapted dense 3D pointcloud as an output. For efficient and consistent transformation of the existing structure, a 7 DoF similarity transformation field is estimated recursively in an octree representation. Problematic regions are identified by an efficient nearest neighbor search and new cameras and scene parts are integrated in a combined extension and annealing step. We have shown on multiple datasets that the algorithm maintains the same accuracy and completeness as re-running the complete dense reconstruction step from scratch, while a significant speed-up is achieved. An analysis of the runtime shows that the algorithm scales equally well as the baseline method when increasing the size of the dataset. Since most of the points are re-used for the final 3D representation, the algorithm can also be used for transforming a triangular mesh. The algorithm is particularly useful in mobile scenarios with limited computational and time budget — due to the reduced runtime, a much lower latency between the image capture and the output of a dense 3D representation is achieved compared to the baseline.

Acknowledgements Research funded by the EUs Horizon 2020 programme under grant agreement No 687757 – REPLICATE and supported by NVIDIA Corporation through the Academic Hardware Grant.

References

[1] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *TVCG*, 5(4):349–359, 1999.

[2] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards Internet-scale multi-view stereo. *CVPR*, 2010.

[3] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 32(8):1362–1376, Aug 2010.

[4] J. Heinly, J. L. Schönenberger, E. Dunn, and J.-M. Frahm. Reconstructing the World* in Six Days. *CVPR*, 2015.

[5] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof. Incremental Surface Extraction from Sparse Structure-from-Motion Point Clouds. *BMVC*, 2013.

[6] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629, 1987.

[7] Z. Kang. Meshrecon dataset. available online: `http://zhuoliang.me/meshrecon.html`.

[8] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. *SGP*, 2006.

[9] A. Locher, M. Perdoch, H. Riemenschneider, and L. Van Gool. Mobile Phone and Cloud a Dream Team for 3D Reconstruction. *WACV*, 2016.

[10] A. Locher, M. Perdoch, and L. Van Gool. Progressive Prioritized Multi-view Stereo. *CVPR*, 2016.

[11] J. Roters and X. Jiang. Incremental dense reconstruction from sparse 3d points with an integrated level-of-detail concept. In *Advances in Depth Image Analysis and Applications*, pages 116–125. Springer, 2013.

[12] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. *CVPR*, 2008.

[13] T. Sugiura, A. Torii, and M. Okutomi. 3d surface extraction using incremental tetrahedra carving. *ICCV Workshop*, 2013.

[14] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live Metric 3D Reconstruction on Mobile Phones. *ICCV*, 2013.

[15] TU Darmstadt. Citywall dataset. available online: `http://www.gcc.tu-darmstadt.de/home/proj/mve`.

[16] O. Untzelmann, T. Sattler, S. Middelberg, and L. Kobbelt. A scalable collaborative online system for city reconstruction. *ICCVW*, 2013.

[17] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *IJRR*, 34(4-5):598–626, 2015.

[18] C. Wu. VisualSFM: A Visual Structure from Motion System, 2011.

[19] S. Yu and M. Lhuillier. Incremental reconstruction of manifold surface from sparse visual mapping. *3DIMPVT*, 2012.