2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Automatic Reconstruction of Textured 3D Models

Benjamin Pitzer, Sören Kammel, Charles DuHadway, and Jan Becker
Research and Technology Center North America
Robert Bosch LLC
Palo Alto, California

Abstract— **This paper describes a system for automatic mapping and generation of textured 3D models of indoor environments without user interaction.**

Our data acquisition system is based on a Segway RMP platform which allows us to automatically acquire large amounts of textured 3D scans in a short amount of time. The first data processing step is registration and mapping. We propose a probabilistic, non-rigid registration method that incorporates statistical sensor models and surface prior distributions to optimize alignment and the reconstructed surface at the same time. Second, in order to fuse multiple scans and to reconstruct a consistent 3D surface representation, we incorporate a volumetric surface reconstruction method based on a oriented point. For the final step of texture reconstruction, we present a novel method to automatically generate blended textures from multiple images and multiple scans which are mapped onto the 3D model for photo-realistic visualization. We conclude our report with results from a large-scale, real-world experiment.

The most significant contribution of this research is a functional system that covers all steps required to automatically reconstruct textured 3D models of large indoor environments.

I. Introduction

3D representations of environments are important for a wide variety of current and future applications: autonomous navigation of robots, architecture, cultural heritage, crash and crime site reconstruction, and many more. Emergency planning, facility management, surveillance and real estate applications significantly benefit from 3D maps of building interiors. Creating such models from blueprints is a tedious task and hard to automate since many buildings do not comply with the blueprints created by their architects. And even accurate blueprints do not contain furniture or appliances added after the building construction.

Today, the digital modeling process of such sites is still primarily done manually. Because working time is expensive, these models typically lack details that might be vital for applications such as autonomous robot navigation.

In contrast, a fully automated 3D data acquisition and model generation involves the following complex components:

- automated acquisition of range and image data
- fusion of data from different viewpoints
- integration of range and image data into a a single consistent model
- simplification and smoothing of the model for visualization and storage

In this paper, we address the first three components and present a working system for the automatic reconstruction

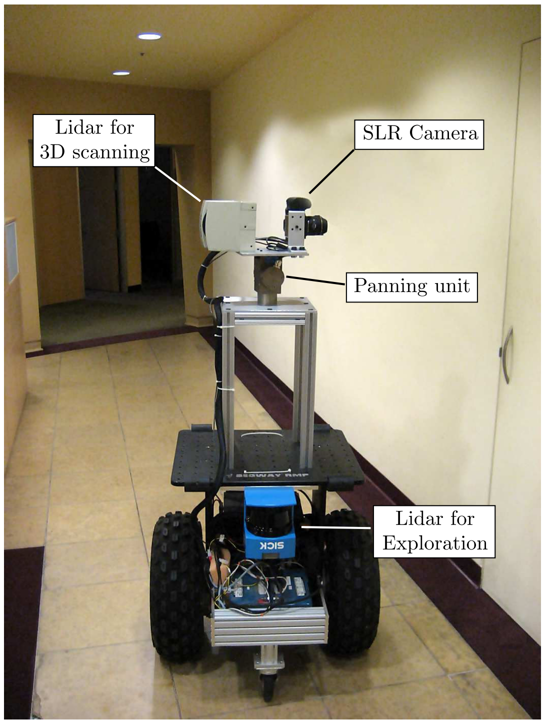

Fig. 1. The scanning platform is based on a Segway RMP, equipped with two laser range finders for navigation and 3D scanning respectively, a digital SLR camera for texture acquisition, and two on-board computers for data processing.

of textured 3D models.

A. State of the Art and Related Work

The reconstruction of 3D models for robot navigation gained significant interest in robotics research over the past years. The progress in this field is mainly based on recent innovations on statistical techniques for robotic mapping and localization. Several successful algorithms emerged, among them CEKF [1], SEIF [2], FastSLAM [3], MLR [4], TJTF [5], and Stochastic Gradient Descent [6], which are all capable of generating maps of large scale environments. Nearly all state of the art methods assume robot operation in a two-dimensional environment and therefore three parameters (2D position and heading) are sufficient to describe the robot's state. Just recently researchers are extending solutions to full 6 DoF poses [7] and mapping of 3D environments [8].

Many research groups use 2D laser range finders to build 3D map representations. Often, a combination of horizontally and vertically mounted scanners are used and localization of the robot and registration of the data is performed in 2D [9], [10]. The mobile robot Kurt3D [11] is among the first robots capable of building 3D maps by registering the data in 3D.

While in the robotics community laser range finders are predominant for accurate mapping tasks, in the computer-vision domain, researchers have developed powerful algorithms to reconstruct 3D models from photographs. Multi-view stereo (MVS) [12] is one of the most successful approaches which produces dense models. Another notable example is Furukawa et. al [13], who presented a fully automated 3D reconstruction and visualization system for architectural scenes based on camera images. Although significant progress to improve the robustness of computer-vision reconstruction approaches was made in the past years, the approaches yet cannot compete with the accuracy of laser range finders. Specifically, textureless scenes which are often found in indoor environments remain very challenging.

Our reconstruction approach is similar to existing approaches [14], [15]. The RESOLV project [14] aimed at modeling interiors for virtual reality and tele-presence and used a RIEGL laser and the well known ICP algorithm [16] for scan matching. However, their approach was designed to reconstruct small (single-room sized) environments; operating on the scale of a full office floor poses a major challenge. The AVENUE project [15] targeted the automation of the urban environment modelling process and used a CYRAX laser scanner and a feature-based scan matching approach for registration of the 3D scans.

B. Overview

In the following, we present a system that enables the automatic creation of 3D models of large environments in a short amount of time. Fig. 2 shows the complete reconstruction process. The process is divided into the five steps *exploration*, *data acquisition*, *global registration*, *surface reconstruction* and *texture reconstruction*.

The *exploration* contains functions that enable the robot to autonomously navigate and explore the environment. This includes online 2D mapping for localization and collision avoidance, planning of view points for data acquisition, and navigating in between 3D scans. *Data acquisition* comprises the acquisition of the laser scans from a panning lidar sensor, which are then merged into 3D scans. The digital still camera simultaneously acquires images, which are undistorted using an offline camera calibration and subsequently merged into a circular panorama. The *global registration* step aligns the 3D points from the individual scans into a consistent map using a joint registration and reconstruction algorithm. A unique surface represented by a triangulated mesh is then generated using volumetric *surface reconstruction*. Finally, textures are generated from still images and blended over the mesh in the *texture reconstruction* step.

C. The Experimental Scanning Robot

For our experiments we use a Segway RMP robot as shown in Fig. 1. The RMP can carry loads up to 50 kg over a range of 15km. For the purpose of high-quality measurements, we equipped the RMP with an additional castor-wheel and disabled the dynamic stabilization. To collect data and perform online mapping and exploration, two on-board Mac Mini computers (2.0GHz Core 2 Duo processor, 1GB RAM) are mounted under the base plate. The computers use the open-source Robot Operating System (ROS) [17] which provides a structured communications layer above the host operating system (linux in our case).

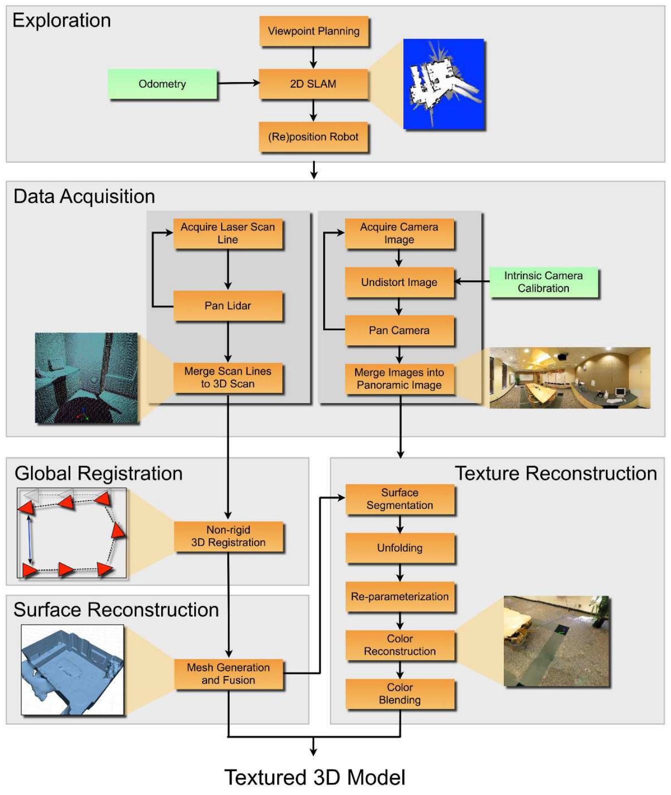

Fig. 2. Overview of the reconstruction and modelling process.

II. Exploration

A. 2D Mapping and Localization

For navigation, exploration, and localization purposes the robot builds and maintains a 2D map of its environment. The main sensor for this system is a horizontally mounted SICK LMS200 laser range finder (cf. Fig. 1). The laser readings and wheel odometry are sent to a SLAM module based on GMapping [18] which constructs a consistent high resolution 2D grid-based map of the environment suitable for path planning.

B. View Point Planning

Frontier based exploration [19] is used to provide active exploration of the robot's environment. An additional 2D grid map is maintained to record which parts of the environment have been observed. Boundaries between observed and unobserved regions (frontiers) are used as goal points for the robot's navigation system. Upon arriving at a goal point the robot will perform a full 3D scan of its environment and update its exploration map. New boundaries and goal points will then be calculated. This process repeats until no reachable frontiers remain.

III. Data Acquisition

A. Sensors

The range measurement component of this scanning system consists of a SICK LMS200 laser range finder which is mounted on a pan-rotation unit such that the plane of the laser's sweep is aligned with the vertical axis. The LMS unit provides accurate measurements up to a range of 30 meters over 180 degrees and with 1/2-degree resolution. Panning the laser 360 degrees about the vertical axis yields a spherical range image as shown in Fig. 3. The panning speed is adjusted to also yield a 1/2-degree scan resolution.

A digital still image camera equipped with a fish-eye lens is mounted on the same rotation unit opposite of the laser. This setup allows the system to capture high-resolution pictures of the scene while panning. Because of the camera's wide field-of-view it only needs to take six pictures to cover the scanned space.

B. System Calibration

In order to fuse data from different scan positions and texturize the point data obtained from the laser range finder, the camera's intrinsic and extrinsic parameters as well as the laser range finder's pose have to be determined. The intrinsic camera parameters were estimated using the method described in [20] assuming a pin-hole camera model with three radial and two tangential distortion parameters. The camera pose relative to the lidar is determined from correspondences of 3D points from the laser range finder with image pixels from the camera. The correspondences are selected manually from several camera views and a panoramic range image. The extrinsic parameters are calculated from the point correspondences by minimizing the reprojection error [20].

C. Point Cloud Generation

A three-dimensional point cloud is generated by panning the laser and associating each vertical scan line with its pan angle. Since the raw scan data is not sampled equidistantly, it is resampled into an equidistant spherical grid. Each grid cell contains the distance measurement which is closest to the center of the cell. Measurements inside a cell are not averaged since this would cause artifacts at depth discontinuities. Some cells in the depth grid may not contain any valid depth measurements if an object is out of range or if the laser beam hits an absorbant surface and never returns. Those cells are marked invalid in subsequent processing steps.

IV. Global Registration

Multiple 3D scans are necessary to digitize large environments without occlusions. To create a correct and consistent model, the scans have to be merged into one common coordinate system. Since the robot does not have

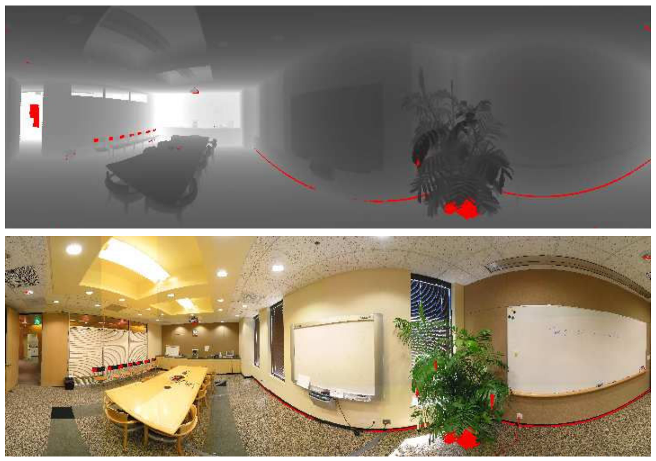

Fig. 3. Panoramic range image (top) and texture image (bottom) with 1/2 degree resolution. Invalid cells are marked red.

a precise, externally referenced position estimate, we have to address the problem of simultaneous localization and mapping (SLAM).

We use a novel probabilistic technique for solving the *offline* SLAM problem by jointly solving the data registration problem and the faithful reconstruction of the underlying geometry. The key insight of this approach is to incorporate a generic surface prior which guides the optimization towards maps that closely resemble the real environment. A more detailed description of this approach for the 3-DoF case can be found in [21].

The goal of SLAM is to simultaneously estimate both the robot's pose and a map of its environment. In probabilistic SLAM this is often expressed in a Bayesian filtering formulation [22]. Thrun et al. have shown [23] that a closed form expression of a posterior over the robot's pose and the map can be obtained by recursively applying the Bayes rule and a subsequent induction:

$$p\left(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}\right) = \eta\, p\left(\mathbf{x}_0\right) p\left(\mathbf{m}\right) \prod_t \left[p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) \prod_k p\left(\mathbf{z}_t^k | \mathbf{x}_t, \mathbf{m}\right) \right] \quad (1)$$

Here we adapted the common notation where at a time t the following quantities are defined: $\mathbf{x}_t$ is a vector describing the 3D position and attitude of the robot, $\mathbf{u}_t$ denotes a control vector that was applied at time $t-1$, $\mathbf{z}_t^k$ corresponds to the k^{th} observation, and $\mathbf{m}$ represents the map as a vector of feature $\mathbf{m} = \{m_i\}$.

In Eq. (1), $p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right)$ is known as the *motion model* which describes state transitions of the robot's pose in terms of a probability distribution. The state transitions are assumed to be a Markov process and independent of both the observations and the map. The term $p\left(\mathbf{z}_t^k | \mathbf{x}_t, \mathbf{m}\right)$ on the other hand denotes an *observation model* which models a observation z_t^k from a known pose and a known map as a probability distribution. Both models have been well studied for a variety of robots and sensors. We use a probabilistic motion model where the robot is assumed to perform a series of a rotation, a translation, and a second rotation [24] with an extension to the 6-DoF state space. Observations are modeled as a range measurement along a beam, which originates at the local coordinate system of the sensor [24].

The two prior terms $p(\mathbf{x}_0)$ and $p(\mathbf{m})$ characterize priors about the first robot pose and about the map respectively. Usually $p(\mathbf{x}_0)$ is used to anchor the initial pose to a fixed location. The map prior $p(\mathbf{m})$ is typically assumed to be unknown and subsumed into the normalizer η [25]. In our formulation, we want to explicitly use the map prior to achieve a better estimate of the robot's pose and the map.

A. Map Prior

The probability distribution $p(\mathbf{m})$ in Eq. (1) represents a *prior distribution* of all measured scenes. An exact probabilistic model of this distribution is infeasible and probably not even well defined. Hence we focus on partial models, which represent properties of the surface structure. We use a so called *manifold prior*. This prior is based on the idea

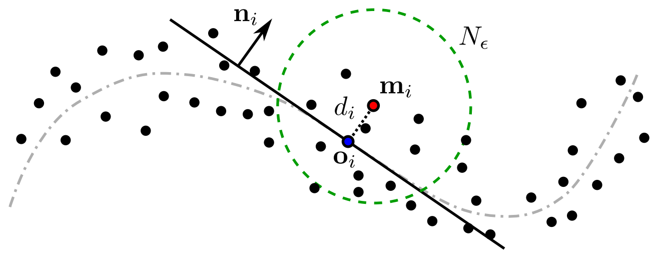

Fig. 4. The *manifold prior* uses a fixed neighborhood N_ε of a point to create a tangent plane defined by a point $\mathbf{o}_i$ and the normal $\mathbf{n}_i$. The distribution is then modelled as a Gaussian over the projected distance to the tangent plane.

that observations belong to continuous surfaces in the robot's environment. For a 3D map this means that the most probable surface must be a compact, connected, two-dimensional manifold, possibly with boundary, embedded in $\mathbb{R}^3$. The first step towards defining such a prior is to compute the tangent plane associated with each observed point $\mathbf{m}_i$. A tangent plane is defined by a 3D point $\mathbf{o}_i$ and normal $\mathbf{n}_i$. For all points we choose a local neighborhood N_ε of fixed diameter (typically $\varepsilon = 10 \ldots 20$ points). The center $\mathbf{o}_i$ is taken to be the centroid of N_ε, and the normal $\mathbf{n}_i$ is determined using principal component analysis [26]: the eigenvector with the smallest eigenvalue corresponds to the normal $\mathbf{n}_i$. The projected distance d_i of the point onto its tangent plane is defined by the dot product:

$$d_i = (\mathbf{m}_i - \mathbf{o}_i) \cdot \mathbf{n}_i \ . \tag{2}$$

Now we can define a Gaussian type manifold prior of the form:

$$p_m(\mathbf{m}) = \eta_m \prod_i \exp\left\{-\frac{d_i^2}{2\sigma_m}\right\}, \tag{3}$$

where σ_m is the variance of tangent plane distances and $\eta_m = \prod_i \left(\sigma_m \sqrt{2\pi}\right)^{-1}$ is a normalization factor.

Fig. 4 shows the properties of this prior. The observed points are drawn to their corresponding tangent planes.

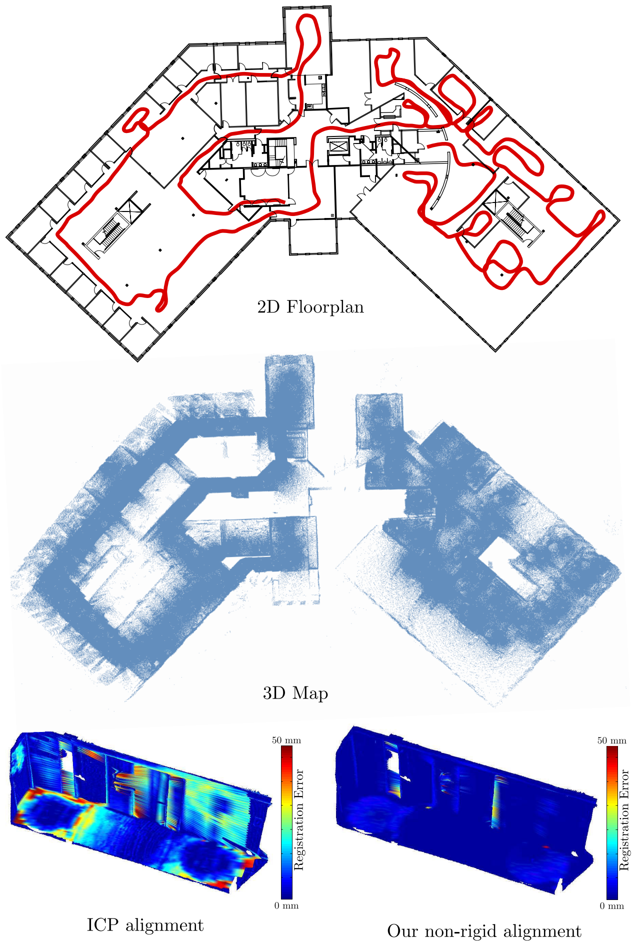

Fig. 5. The top figure shows a manually created floorplan and the trajectory taken by our scanning robot. The middle figure presents the 3D pointcloud registered with our registration algorithm. The enlarged detail of a hallway demonstrates that using our probabilistic non-rigid method results in a more accurate registration: The registration error visualization reveals a slight miss-alignment for the ICP registered dataset, while our non-rigid technique results in a good alignment over the whole surface.

Hence the most probable arrangement given only this prior is when all points are located on the same one-dimensional manifold. The point motion will be constrained due to the dependence of measurement and pose. In fact, a movement of a point will create a counter potential for the point and for the corresponding pose to comply with the measurement model. In other words, maximizing the posterior probability Eq. (1) will lead to a set of poses and map features that best explain the measurements as well as the prior model.

B. Optimization

First we use the position estimates of the navigation system as an initial estimate for $\mathbf{x}_{1:t}$ and the measurement model to calculate and initial estimate for $\mathbf{m}_{1:i}$. Next, we use a non-linear conjugate gradient variant to find the parameters which maximize the log-likelihood of $p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$. The result of this optimization is presented in Fig. 5.

V. Geometry Reconstruction

In our system, we use an algorithm which does not make any prior assumptions about connectivity of points. This *volumetric approach* for surface reconstruction is more efficient in situations where multiple scans are taken of the same surface as the 3D points are accumulated into voxel grid structures first.

An important tool for surface reconstruction from unorganized points is the signed distance function $\zeta : \mathbb{R}^3 \rightarrow \mathbb{R}$ that measures for each point the signed distance to the surface. The *implicit surface* $\mathcal{S}$ is defined as a zero-set of this scalar function $\mathcal{S} : \zeta(\mathbf{x}) = 0$ with $\mathbf{x} \in \mathbb{R}^3$. The aim is to construct a smooth volumetric field function $\zeta(\mathbf{x})$, such that the zero-set approximates the real surface as closely as possible.

The first step of our surface reconstruction approach is to calculate a 3D indicator function χ (defined as 1 for points inside the model, and 0 for points outside). Kazhdan et al. [27] show that there exists an integral relationship between points sampled from the real surface and this indicator function. Specifically, they found that the problem of finding the indicator function reduces to finding the scalar function χ whose gradient best approximates a vector field $\mathbf{V}$ defined by the scan points, i. e. $\min \|\nabla\chi - \mathbf{V}\|$.

Since the gradient vectors of the binary indicator function would be unbounded at the surface, we convolve χ with a smoothing filter F and consider the gradient of the smoothed function. One can show [27] that the gradient of the smoothed indicator function is equal to the smoothed surface normal field:

$$\nabla(\chi \star F)(\mathbf{q}) = \int_{\mathcal{S}} F(\mathbf{q})\, N_{\mathcal{S}}(\mathbf{p})\, d\mathbf{p} \approx \mathbf{V}(\mathbf{q}) \quad (4)$$

where $\mathbf{q} \in \mathbb{R}^3$ and $N_{\mathcal{S}}(\mathbf{p})$ is the surface normal at $\mathbf{p} \in \mathcal{S}$. The surface normal field can be best approximated by the oriented scan points. In other words, the oriented point samples can be viewed as samples of the gradient of the model's smoothed indicator function. If we apply the divergence operator on both sides of Eq. (4), the variational problem transforms into a standard Poisson problem:

$$\triangle(\chi \star F) = \nabla \cdot \mathbf{V} \quad (5)$$

which can be solved efficiently by discretizing the 3D space into a regular grid $\mathcal{G}$ and using this grid as a space of functions. For each grid cell c, we set $F_c : \mathbb{R}^3 \rightarrow \mathbb{R}$ to be the smoothing function for a local patch. We choose F_c to be a *bilateral filter* [28] centered about the cell's centroid o_c of the following form:

$$F_c(\mathbf{q}) = \frac{1}{w_q} \sum_{i \in \mathcal{G}} G_{\sigma_s}(\|\mathbf{o}_c - \mathbf{q}\|)\, G_{\sigma_r}(|\mathbf{n}_c \cdot \mathbf{n}_i|)\, \mathbf{n}_i \quad (6)$$

where $G_\sigma(x)$ denotes a Gaussian kernel, n_c is the cell's normal vector and w_q is a normalization factor:

$$w_q = \sum_{i \in \mathcal{G}} G_{\sigma_s}(\|\mathbf{o}_c - \mathbf{q}\|)\, G_{\sigma_n}(|\mathbf{n}_c \cdot \mathbf{n}_i|) \ . \quad (7)$$

The parameters σ_s and σ_n will measure the amount of filtering for the normal field. Similar to a Gaussian convolution as proposed by Kazhdan et al. [27], the bilateral filter of Eq. (6) is a normalized weighted average where G_{σ_s} is a *spatial* Gaussian that decreases the influence of distant cells, G_{σ_n} a Gaussian that decreases the influence of cells i with a normal vector different from $\mathbf{n}_c$. Unlike the Gaussian filter our bilateral filter takes the variation of normals into account in order to preserve sharp features.

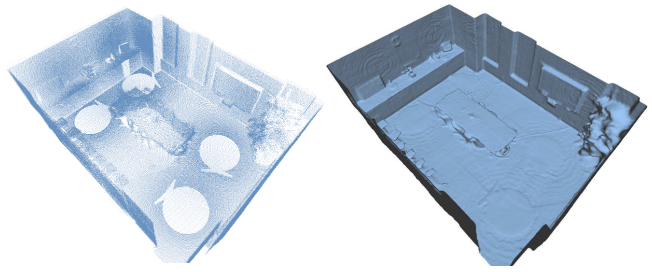
Fig. 6. Volumetric surface reconstruction based on oriented points.

Once the vector field is defined for each grid node, the gradient field of our indicator function defined in Eq. (4) can be efficiently represented as linear sum of all node functions. Now, we can solve the indicator function χ such that the gradient of χ is closest to $\mathbf{V}$.

Finally, to extract the iso-surface $\mathcal{S}$ from the indicator function, a method similar to the *Marching Cubes* algorithm [29] is used. This method creates vertices at zero-crossings of χ along edges of the grid nodes. The vertices are connected to triangles such that a continuous manifold along $\mathcal{S}$ is formed.

VI. Texture Reconstruction

In our system, we capture color images from a digital camera together with the geometry. We use these images to reconstruct texture maps which are mapped onto the 3D model to generate a greater realism. Our texture reconstruction approach consists of the following steps: *surface segmentation*, *surface unfolding*, *mesh re-parameterization*, *color reconstruction*, and *color blending*.

A. Surface Segmentation

The first subproblem for texturing a complex 3D surface is finding a surface partitioning. We seek to break the surface into several regions such that the distortion when flattening each region onto a plane is sufficiently small while the number of regions remains small at the same time. Since planes are developable surfaces (with zero Gaussian curvature) by definition, one possible approach is to segment the surface into nearly planar regions. We employ an incremental clustering approach with a subsequent merging strategy. Regions are grown from randomly chosen seeds and adjacent faces with similar surface normals are iteratively added. A major problem of this segmentation procedure is the resulting over-segmentation. In order to reduce the over-segmentation, we append an optimization procedure to merge segments by incorporating information about their similarity.

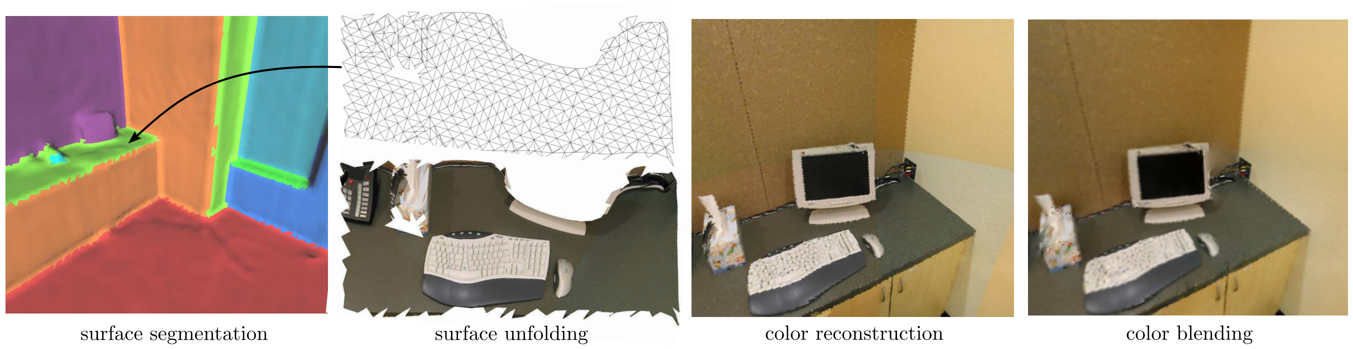

Fig. 7. For the texture reconstruction the mesh is *segmented* into nearly planar regions and each region is *unfolded* onto a 2D plane. The texture of each region is *reconstructed* from all camera images observing this part of the surface and the resulting color composite is *blended* afterwards to avoid discontinuity artifacts.

B. Unfolding

Given a set of disjoint surface regions, we compute a mapping from each surface point of a region to the texture domain. A rather simple way for constructing such a parameterization of a triangle mesh is based on the fact that the previously described segmentation procedure results in almost planar surface segments. We can find the best fitting plane in a least squares sense by using *principal component analysis* (PCA). The result is an orthogonal linear transformation W that transforms the data point $\mathbf{p}_i = (x_i, y_i, z_i)^{\mathrm{T}}$ to a new coordinate system $\tilde{\mathbf{p}}_i = (\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)^{\mathrm{T}}$ such that the greatest variance of the data is along the first coordinate and the smallest variance along the third coordinate. Then a mapping can be defined by projecting the transformed coordinates onto the plane spanned by the first and the second coordinate axis:

$$\mathbf{u}_i = \begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tilde{\mathbf{p}}_i \ . \tag{8}$$

Even though this method does not guarantee to result in a bijective mapping, we can easily check this criterion for each mapping. In a bijective map, the order of the triangle vertices (anticlockwise) will be preserved. In practice, the mapping is always bijective since the segmentation algorithm results in almost developable regions.

C. Mesh Re-parameterization

After having determined a mapping for each segment, we re-parameterize the mesh to obtain a densely sampled surface. For the re-parameterization we use an equidistant point grid in texture space where each grid point corresponds to a texture pixel. The space spanned by this grid will become our texture space $\mathcal{T}$.

In order to determine if any $\tilde{\mathbf{p}}_i = (u_i, v_i)^{\mathrm{T}}$ is inside or outside of a mapped triangle, we calculate the barycentric coordinate. The barycentric coordinate of the point $\tilde{\mathbf{p}}_i$ with respect to the vertices $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ of a triangle is a triplet of values, $\{b_1, b_2, b_3\}$, such that $\tilde{\mathbf{p}}_i = b_1\mathbf{v}_1 + b_2\mathbf{v}_2 + b_3\mathbf{v}_3$, with $b_1 + b_2 + b_3 = 1$. $\tilde{\mathbf{p}}_i$ lies inside the triangle if b_1, b_2, and b_3 are positive. In this way, we find the corresponding triangles for all points of the grid and use the barycentric coordinate to interpolate the mapping f at the point's coordinates. Grid points which are not part of the mapped surface are discarded.

D. Color Reconstruction

Knowing the pose and the intrinsic calibration of our scanner allows us to project any 3D surface point into any of the original camera images to retrieve the color from this particular view. However, every view carries only information on a part of the reconstructed surface. To find out if a given 3D point is visible in a certain view, we first transform the point into the camera coordinate system using the known view pose. Next, we use the intrinsic camera calibration to project the point to pixel coordinates. If the resulting coordinates are valid (i. e. in the range of the image dimensions) we can conclude that the 3D point is in the camera's view frustrum. However, the environment geometry possibly creates complex occlusions and makes it difficult to recognize if a 3D point was truly observed by the camera. To test if the 3D point is occluded in this view we trace the ray originating at the point to the center of the camera and determine if it intersects with any surface. This test can be efficiently performed using the GPU's occlusion culling. In many cases, a 3D point is visible from more than one view and in this case we reconstruct the color from the view closest to the 3D point.

E. Color Blending

Since the reconstructed texture maps are composites from multiple camera images, discontinuity artifacts usually become visible. The reason for those artifacts is that the surface reflectance varies by distance and angle of incident. For a consistent texturing we want to minimize the visibility of these discontinuity artifacts. We approach this problem by using a blending technique, which globally adjusts the color of all pixels.

Our algorithm extends the ideas of [30] to use a Poisson formulation for our multi-view blending problem. The procedure is as follows: for a texture with regions reconstructed from N camera images, we can treat the regions as separate functions: $f_{1:N}$. Now, let $\Omega_{1:N}$ be the definition space of

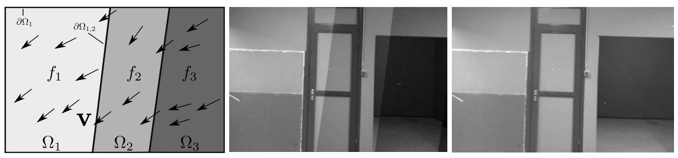

Fig. 8. The texture blending globally optimizes the texture color and removes discontinuities at boundaries between texture regions reconstructed from different camera images. The image in the middle shows the reconstructed texture and the right image the texture after blending.

$f_{1:N}$, $\partial\Omega_{i,j}$ be the boundary between Ω_i and Ω_j, and $\partial\Omega_i$ the texture boundary of the i^{th} texture. Finally, we define $\mathbf{V}$ to be a guidance vector field defined over $\Omega_{1:N}$. See Fig. 8 (left) for an illustration of this notation.

Our goal is to find $f'_{1:N}$ which have the same definition space as $f_{1:N}$ and no visible discontinuities at their boundaries. We cast this problem as membrane interpolant that satisfies:

$$\min_{f'_{1:N}} \sum_i \iint_{\Omega_i} |\nabla f'_i - \mathbf{V}|^2 \tag{9}$$

with the Dirichlet boundary conditions $f'_i\mid_{\partial\Omega_{i,j}} = f'_j\mid_{\partial\Omega_{i,j}}$ and $f'_i\mid_{\partial\Omega_i} = f_i\mid_{\partial\Omega_i}$. We set the guidance vector field $\mathbf{V}$ to equal the derivatives of $f_{1:N}$, which means we constrain the derivatives of $f'_{1:N}$ to be the same as the derivatives of $f_{1:N}$. The first boundary constraint guarantees a smooth boundary between texture regions while the second constraint is necessary since the gradient operator is invariant through multiplicative factors. The solution of Eq. (9) is the unique solution of the following Poisson equation:

$$\nabla \cdot \nabla f_{1:N} = \triangle f_{1:N} = \nabla \mathbf{V} \;\text{ over }\; \Omega_{1:N} \tag{10}$$

under the same boundary conditions as Eq. (10). In the discrete texture domain, this can be efficiently solved as a sparse linear system. See [30] for further details.

VII. Results

A number of experiments have been conducted using the previously described approaches. In particular, we have created a 3D model of Bosch's office in Palo Alto. Snapshots of this model are depicted in Fig. 9.

The largest fraction of the time required for the complete 3D reconstruction process was spent on the data acquisition; 6 hours were necessary to scan one office floor by taking 127 scans (approx. 3 min per scan). Registration, surface and texture reconstruction took on the order of 100 minutes for the Bosch dataset on a standard desktop computer (3.4 GHz, 4GB RAM). About 70% of this time was spend on IO operations on the 8GB of compressed raw data. The registration was performed on a sub-sampled dataset and took 20 minutes to converge. Projecting the registration results onto the high-resolution data yielded good results. Our volumetric surface reconstruction approach found a highly detailed approximation of the real surface in approximately 65 minutes. Again, we employed a multi-grid scheme to speed up the reconstruction. Structures such as legs of chairs, as well as plants, due to fine leaf and branch structures, turned out to be problematic. The reconstruction typically fused multiples of such structures into a single blob or merged them with a nearby wall. Improvements are certainly possible by scanning higher resolution, with the obvious drawback of increased memory requirements and extended acquisition and processing times. For the final step of model reconstruction, we found that the automatic texture reconstruction procedure results in high-quality texture maps in only 15 minutes for the Bosch dataset. Some failures led to a distorted, unrealistic looking texture and were caused by inaccurately reconstructed geometry.

VIII. Conclusion

Our research has led to a number of interesting results. We presented an automated system for the creation of large scale 3D models in a short amount of time with moderate hardware requirements. The presented paper describes the required components, specifically *data acquisition*, *registration*, *geometry reconstruction*, and *texture reconstruction*. The capabilities of our system were demonstrated in several experiments by capturing large models with up to more than 54 million triangles covering an area of 50 m by 140 m meter. The resulting quality in terms of geometric and texture details is remarkable and to the best of our knowledge, our system is the first to fully automatically reconstruct large indoor environment models of this quality.

A common restriction made in mapping systems is the assumption of a static environment. Our system obliges the same restriction. Although the registration is fairly robust to artifacts created by a limited amount of dynamic objects during the scan process, the methods presented for geometry and texture reconstruction will fail. More research is required to distinguish dynamic and static parts in a scene and to then consider only the latter for 3D modeling. The ultimate goal remains a system being able to operate autonomous in dynamic or even crowded environments for an indefinite amount of time.

References

[1] J. E. Guivant and E. M. Nebot, "Improving computational and memory requirements of simultaneous localization and map building algorithms." in *IEEE International Conference on Robotics and Automation*, 2002, pp. 2731–2736.

[2] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *International Journal of Robotics Research*, 2004.

[3] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: a factored solution to the simultaneous localization and mapping problem," in *18th National Conference on Artificial Intelligence*, 2002, pp. 593–598.

[4] U. Frese and T. Duckett, "A multigrid approach for accelerating relaxation-based SLAM," in *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, Acapulco, Mexico, 2003, pp. 39–46.

[5] M. A. Paskin, "Thin junction tree filters for simultaneous localization and mapping." in *18th International Joint Conference on Artificial Intelligence*, 2003, pp. 1157–1166.

[6] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *IEEE International Conference on Robotics and Automation*, May 15-19 2006, pp. 2262–2269.

[7] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Robotics: Science and Systems*, 2007.

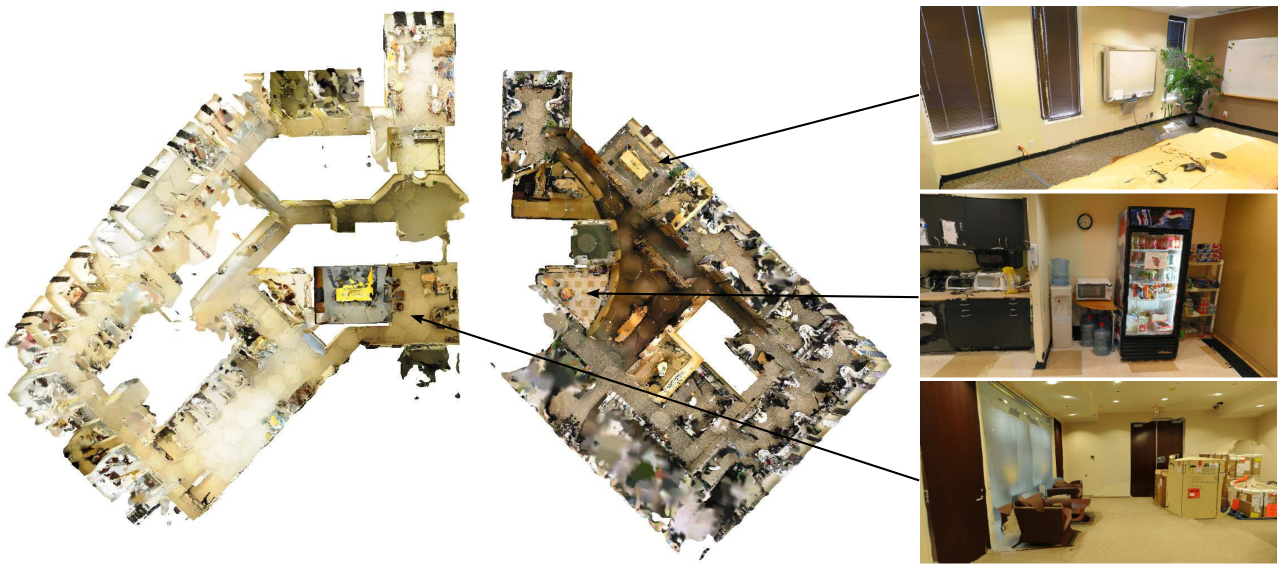

Fig. 9. The reconstructed office model consists of 28.167.234 vertices and 54.745.336 triangles covering an area of 50 m by 140 m meter.

[8] D. Borrmann, J. Elseberg, K. Lingemann, A. Nchter, and J. Hertzberg, “The efficient extension of globally consistent scan matching to 6 dof,” in *4th International Symposium on 3D Data Processing, Visualization and Transmission*, 2008, pp. 29–36.

[9] S. Thrun, W. Burgard, and D. Fox, “A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping,” in *IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2000.

[10] P. Biber, S. Fleck, and W. Strasser, “The wägele: A mobile platform for acquisition of 3d models of indoor outdoor environments,” in *9th Tübingen Perception Conference*, 2006.

[11] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, K. Pervölz, M. Hennig, K. R. Tiruchinapalli, R. Worst, and T. Christaller, “Mapping of rescue environments with kurt3d,” in *International Workshop on Safety, Security, and Rescue Robotics*, Kobe, Japan, June 2005, pp. 158–163.

[12] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, “A comparison and evaluation of multi-view stereo reconstruction algorithms,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2006, pp. 519–528.

[13] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, “Reconstructing building interiors from images,” in *International Conference on Computer Vision*, 2009.

[14] V. Sequeira, K. Ng, E. Wolfart, J. Gonalves, and D. Hogg, “Automated reconstruction of 3d models from real environments,” *Journal of Photogrammetry and Remote Sensing*, vol. 55, no. 1, pp. 1–22, 1999.

[15] A. Georgiev, “Design, implementation and localization of a mobile robot for urban site modeling,” Ph.D. dissertation, Computer Science Dept., Columbia University, New York, NY, 2003.

[16] P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *International Conference on Robotics and Automation*, ser. Open-Source Software workshop, 2009.

[18] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[19] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, California, 1997, p. 146.

[20] Z. Zhang, “Flexible camera calibration by viewing a plane from unknown orientations,” *IEEE International Conference on Computer Vision*, vol. 1, p. 666, 1999.

[21] B. Pitzer and C. Stiller, “Probabilistic mapping for mobile robots using spatial correlation models,” in *IEEE International Conference on Robotics and Automation*, 2010.

[22] S. Thrun, “Robotic mapping: A survey,” in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002.

[23] S. Thrun and M. Montemerlo, “The GraphSLAM algorithm with applications to large-scale mapping of urban structures,” *International Journal on Robotics Research*, vol. 25, no. 5/6, pp. 403–430, 2005.

[24] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[25] T. Bailey and H. Durrant-Whyte, “Simultaneous localisation and mapping (slam): Part i - state of the art,” *Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, June 2006.

[26] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuetzle, “Surface reconstruction from unorganized points,” *Computer Graphics*, vol. 26, no. 2, pp. 71–78, 1992.

[27] M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson surface reconstruction,” in *4th Eurographics symposium on Geometry processing*, Aire-la-Ville, Switzerland, 2006, pp. 61–70.

[28] T. R. Jones, F. Durand, and M. Desbrun, “Non-iterative, feature-preserving mesh smoothing,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 943–949, 2003.

[29] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3d surface construction algorithm,” in *14th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1987, pp. 163–169.

[30] P. Pérez, M. Gangnet, and A. Blake, “Poisson image editing,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 313–318, 2003.