Seminar Work on
Reconstructing Building Interiors from Images

Eduard Feicho

August 13, 2010

Abstract

3D scene reconstruction of outdoor environments is a research topic that is popular as of recent. Impressive systems such as Google Street-View and Microsoft Photo Tourism show the maturity of these algorithms. However, with indoor scenes specific problems arise. In a recent paper, Manhattan-world assumptions constrain these problems such that it is possible to automatically estimate ground plans of buildings from interior images. To this end, axis-aligned planes are estimated first and then used to estimate depth maps, which are finally merged into a mesh surface. This seminar work explains the system and details relevant background information. A short discussion of the system concludes this work.

Contents

1 Introduction

Reconstruction of 3D scene points from a set of 2D images, which is known as Structure from Motion, has been a research topic for some time. The fundamental concept is the well-known epipolar geometry [24], which relates correspondences of image features between two views to the 3D scene point and the camera parameters. Therefore images are reconstructed in a pairwise fashion and then a solution of minimum error is found for all the cameras and reconstructed points.

Structure from Motion algorithms reconstruct a scene by means of sparse 3D scene points, which is not always a suitable output format and does not always provide the information we are actually interested in. For 3D scanning problems dense 3D scene points may be desired while for rendering purposes a simple mesh may be more suited because of memory and speed considerations. Depending on our objectives we therefore face different problems.

Recently, reconstruction of outdoor scenes has been a popular objective. Probably the most well-known approach is the Google Street-View service, where a camera-mounted vehicle drives through city streets acquiring raw video data from multiple cameras and laser scanners. These videos are used in a reconstruction pipeline to provide for Internet panoramas in 3 dimensions of urban places in the world.

A different approach uses images from online image hosting services to automatically reconstruct whole cities offline [40, 1]. The reconstructions can then be used to provide more intuitive path-based image viewing services [39].

This work is focused on the objective of reconstructing images of indoor scenes. Indoor scenes are closely related to urban street view scenes, because both face similar problems. Textured regions are suitable for finding image correspondences, but architectural environments often contain many flat, texture-less walls [16], which poses a problem for standard reconstruction algorithms. Indoor scenes are also related to unordered image collections, such as found on the Internet, because they are typically unordered and visibility reasoning is problematic due to walls that occlude the view.

In [16] a system is proposed for scene reconstruction of indoor environments, which is the main focus of this work. The authors make reconstruction feasible by invoking the Manhattan-world assumption [8, 15], which basically states that a scene only consists of planes aligned with three orthogonal main axes. This assumption is intuitively reasonable, since most walls are built in such a fashion and even much of the interior can be approximated by combinations of bounding boxes. An interesting output of their algorithm is a ground plan that explains how walls, windows and doors are placed.

This seminar paper explains the system and gives background information on the most important foundations. Focus is applied on the first stages of the system, which are 3D reconstruction using the Bundler software package [40] and applying Manhattan-world assumptions.

1.1 Overview

Before the system is explained in detail an outline of its pipeline is given here. It basically consists of four major stages:

In the first stage a sparse set of 3D scene points is estimated from a set of unordered input images using the Bundler software package. This stage is detailed in Section 3.1. The basic techniques which Bundler is based on are explained in the fundamentals Section 2. These include knowledge of how scene points are estimated from two views through epipolar geometry (cp. Section 2.1) and how cameras are calibrated afterwards using structure from motion, in short SfM (cp. Section 2.2). Figure 1 gives an intuition how the resulting sparse 3D scene points look like.

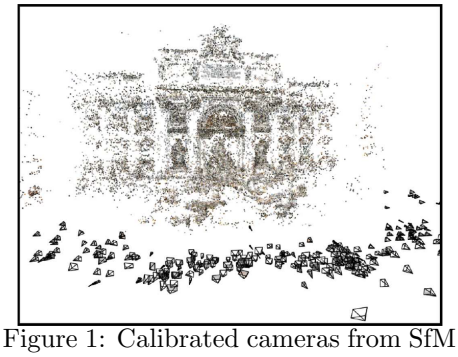


Figure 1: Calibrated cameras from SfM [40]

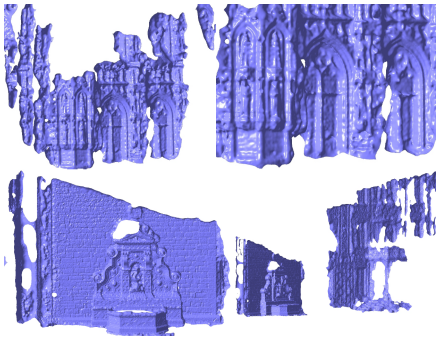


Figure 2: Surfaces from MVS [19]

The second stage aims at recovering a 3D surface or, in other words, a volume. The images and the calibrated cameras are given as input and the volume is reconstructed based on photo-consistency measures. This problem is called multi-view stereo and is explained in general in Section 2.4. The actual algorithm that is used for reconstructing building interiors is called patch-based multi-view stereo (cp. Section 3.2). Figure 2 shows how such a volume could look like. An important result of this stage are oriented points, i.e. each vertex is augmented with a surface normal, out of which main orientations are estimated in the next stage.

In the third stage (cp. Section 4) the specifics of building interiors are focused on. Here, the so-called Manhattan-world assumption is applied. It assumes that the scene consists of three orthogonal major axes (typically known as x-,y- and z-axes) and all surfaces are axis-aligned planar. The method to estimate these major axes is explained in Section 4.2. Then, the most evident axis-aligned (infinite) planes are estimated from clusters of plane offsets in Section 4.3. Finally, depth maps are constructed for each view, where each pixel is assigned to one of the planes. The problem is formulated as a Markov Random Field (MRF), which can be solved efficiently using graph cuts. MRFs and graph cuts are generally explained in Section 2.3 in order to establish a basic understanding. Figure 3 illustrates the kind of Manhattan-world depth maps that are recovered in this stage.

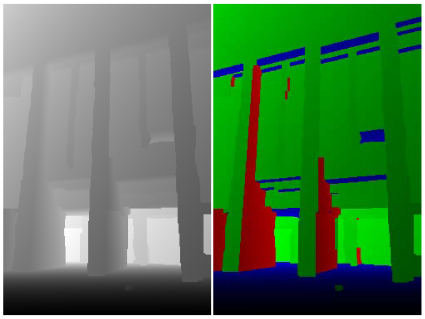


Figure 3: Manhattan-world depth maps [15]

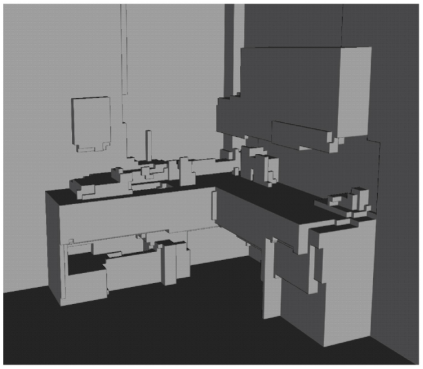


Figure 4: Manhattan-world mesh [16]

In the final stage (cp. Section 5) the depth maps are used to recover the surface of the planar structures. The problem of finding the surface boundaries is formulated as an MRF over a volumetric cost function (cp. Section 5.1). Given the minimal solution a mesh is then extracted (cp. Section 5.2), which is more suitable for image rendering purposes. Figure 4 shows what remarkably clean results are obtained with this technique. Further results are detailed in Section 6.

All of these stages are explained with more focus on the last two stages, which are specific to building interiors. Finally, this work concludes with a discussion in Section 7.

1.2 Related Work

The reconstruction algorithm for building interiors [16] builds on a fair amount of previous work in the fields of structure from motion (SfM), multi-view stereo (MVS), plane-fitting and stereo fusion. Interestingly, a lot of this work originates from the same group of authors from the University of Washington [40, 41, 19, 15].

The SfM techniques that are used have been proposed in [40]. There, the focus is on reconstructing large scale scenes from unordered image collections such as found on the Internet. Further research in scaling these techniques include [41, 1]. These types of images are related to indoor scenes as they are both typically obtained in an unordered fashion from a lot of different unrelated viewpoints.

Many MVS algorithms exist and are classified in [36]. Closely related to this work are the

methods that represent the scene with depth maps, where smoothness priors in the depth maps naturally direct towards 2D Markov random field formulations [42, 26, 45, 21, 31], which is used for Manhattan-world stereo [15]. Also related are methods that formulate a 3D volumetric MRF over voxels [34, 43, 26, 37, 18], which is similar to the final stage of reconstructing building interiors.

The Manhattan-world assumption is a strong prior that is used in this work. Related work that also incorporates these assumptions includes [2, 20, 29, 38]. Note, many of these approaches also belong to the class of outdoor urban scene stereo methods [20, 29, 7, 35, 3]. Images of indoor, outdoor and man-made scenes are related, because they are similarly texture-less, which is problematic for classical MVS algorithms. This work is also related to approaches for aerial imagery [44]. Due to their close relation to this work some of the approaches incorporating Manhattan-world priors are detailed further.

In [20] the scene points from sparse structure from motion are analyzed for lines, from which vanishing points are recovered. These in turn can be combined to give estimates of plane normals. The dominant Manhattan-world axes are recovered from them while perpendicularity constraints are applied. Plane-sweeping stereo based on [5] is then applied in the main directions in order to estimate depth maps of the planar scene. The results of these multiple sweeps are merged using MRFs and graph cuts.

[7] uses a simpler model that is similar to Manhattan-world. The world is assumed to consist of only three planes: The ground and two facades. Vertical lines are used as strong priors for facade reconstruction. Additionally, object detection is used to suppress obstacles that frequently appear in urban scenery.

[29] uses superpixels to overcome the effect of texture-less regions. Vanishing points are used to estimate the Manhattan-world axes. The dominant scene orientations as well as the superpixels are adopted in an MRF formulation in order to recover depth maps. The depth maps are not fused with MRFs, but instead a filter is iteratively applied on the reconstructed 3D points in order to filter out outliers. The algorithm uses kd-trees for efficiency.

In [38] piecewise planar surfaces are estimated that are not required to be aligned with three dominant axes. However, the approach is closely related to [16]. First, feature matching and structure from motion produces a sparse scene reconstruction. Planes are directly fitted to 3D points and lines utilizing vanishing point cues. As mentioned before, these are not limited to three orthogonal directions. An MRF is formulated to generate piecewise planar depth maps by incorporating photo-consistency, ray visibility constraints and geometric constraints using 3D lines and vanishing points. The authors achieve good results on both indoor and outdoor scenes.

Note, another interesting related prior is proposed in [35]. It elevates the idea of Manhattan-world to so-called Atlanta-world scenes. There, multiple dominant axes are estimated for subregions of the image instead of a single one for the whole image.

2 Fundamentals

This section gives an overview of fundamental techniques that are referred throughout this work. These include techniques to obtain the scene geometry with epipolar geometry (cp. Section 2.1) and structure from motion (cp. Section 2.2). Markov random fields (cp. Section 2.3) are detailed more than the other techniques, because of the key role MRF formulations play in later sections. An introduction to multi-view stereo is given in Section 2.4. Finally, Delaunay Triangulation is explained in Section 2.5

2.1 Epipolar Geometry

This section gives a short introduction to epipolar geometry. It describes the geometry between two images that see a common point in 3-space. The image projections of such a point are called correspondences. If such correspondences of a finite number of different scene points is given, the relative camera pose between the two cameras can be calculated. It is also possible to use more than two cameras, but stereo setups are used more often. A two-view setting is a natural choice with human eyes as an ideal.

Let (x, x') be a pair of corresponding image points in two views with cameras P, P'. The image points are projections of the same world point X. There exists a geometric relation between these points, which is called epipolar geometry and leads to the term fundamental matrix F.

Figure 5 indicates that the points x, x' and X lie on a plane π. The line joining the cameras is called baseline. The points where the images intersect the baseline are called epipoles. The line, which intersects the epipole and the ray backprojected by X is called the epipolar line. x is basically related to x' by a projective mapping from points-to-lines, x to l'. This projective relation in 2-space is manifested in the so-called homography H_π.

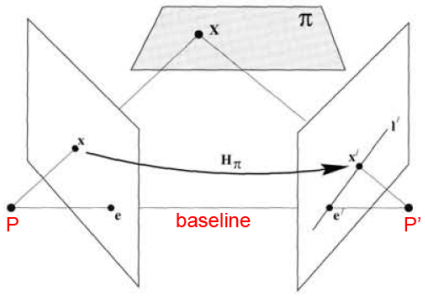


Figure 5: P, P' are cameras. x, x' are image points. The point X in 3-space projects on two views to x and x'. The correspondence (x, x') is geometrically described by a projective mapping in 2-space, a homography H_π. (*figure taken from [24]*)

Initially, suppose the cameras are calibrated. Then, they are related to each other only by their extrinsic parameters, which are rotation R and translation T. The first camera is fixed at the origin. Let $\vec{x}, \vec{x}'$ be the vectors in 3-space from the camera's pinhole to the image plane at coordinates x and x', respectively. Since $\vec{x}$ and T lie on the same plane π, the normal vector n of π is given by the cross-product of the spanning vectors $T \times X$. Furthermore the dot-product of perpendicular vectors is 0. Using this, the following equation is derived:

$$\vec{n} = T \times \vec{x}' = T \times (R\vec{x} + T) = T \times R\vec{x} \tag{1}$$

$$0 = \vec{x}' \cdot \vec{n} \overset{(1)}{=} \vec{x}' \cdot (T \times R\vec{x}) = \vec{x}'^T \underbrace{(T_\times R)}_{=:E} \vec{x}$$

where $\cdot$ denotes the dot-product and $T_\times$ is a matrix formulation of the cross-product. E is called essential matrix and can be calculated from only correspondences. The epipolar constraint has been derived.

Before the uncalibrated case is discussed a few key properties of the essential matrix are given:

- E is a 3×3 rank 2 homogeneous matrix with five degrees of freedom.
- The epipolar lines are given by $l = E\vec{x}', l' = E^T\vec{x}$.
- For the epipoles e, e' it holds $Ee = e'^T E = 0$.
- **Epipolar constraint:** For correspondences $\vec{x}, \vec{x}'$ it holds $\vec{x}'^T E\vec{x} = 0$.

Next, suppose the cameras are not calibrated. Then, the epipolar constraint can be rewritten by substituting in terms of unknown normalized coordinates:

$$\hat{x}'^T E\hat{x} = 0, \quad \text{with } \vec{x} = K\hat{x}, \ \vec{x}' = K'\hat{x}',$$

where K and K' are the camera calibration parameters. This yields the fundamental matrix with

- $F = K^{-T}EK'^{-1}$.
- F is a 3×3 rank 2 homogeneous matrix with seven degrees of freedom.

Now the fundamental matrix covers the relation of image correspondences. If at least eight of these are given, F can be solved for in linear time, because each point correspondence generates one linear equation in the entries of F and F has 7 degrees of freedom. This way the relative pose of two cameras can be obtained from two views and a set of eight corresponding image points. The eight-point algorithm has been widely used, because it is linear and gives a unique solution [23]. Assuming the camera's intrinsic parameters are known even five image points would suffice for the five-point relative pose algorithm of Nistér [32]. However, the five-point algorithm is not linear or unique anymore. In the worst case it results in ten solutions.

Afterwards it is trivial to solve for the epipoles, since they are the left and right null-space of the essential matrix. Once the epipoles are known, the relative camera pose is directly given by fixing the first camera at origin:

$$P = [I|0] \quad P' = [e'_{\times}F|e']$$

2.2 Structure from Motion

Epipolar geometry obtains relative camera pose from pairs of images. However, with many views the problem is to estimate the camera calibration parameters for all views. Changing a single camera can worsen the total reprojection error. Therefore matrix equations are arranged that solve for the camera parameters of all views while the reprojection error is minimized. Incremental structure from motion is a special type of SfM technique that adds in one camera to the full reconstruction at a time [40].

2.3 Markov Random Fields and Graph Cuts

This section establishes a foundation of key methods that occur throughout this work and in many computer vision problems. These are Markov random fields (MRFs) and graph cuts. A classical example of MRFs in computer vision are image segmentation problems, where segments are considered as smooth areas that may have non-smooth boundaries. MRFs formulate the problem of finding ideal segments in terms of an energy minimization problem. There are methods to solve this problem approximately by rephrasing it as a graph-theoretic problem of finding the min-cut/max-flow of a so-called source-sink graph. Using such graph cuts the solution can then be computed efficiently in polynomial time.

The aim of this section is to give a general, but thorough understanding of these techniques. This not only establishes a foundation to understand depth map reconstruction in Section 4.4, but also MRFs occur again in Section 5, where depth maps are used to recover a 3D mesh model of the scene. The use of MRFs for image segmentation, depth map reconstruction and depth map fusion already gives a clue on how general and important a role MRFs and graph cuts play in many computer vision tasks.

Energy Minimization with Markov Random Fields

In [3] the term energy minimization is explained in the context of early vision. These kind of early vision problems try to identify "quantities", such as pixel intensities, that are smooth on object surfaces and change abruptly on object boundaries. Since data is typically noisy, it is difficult to distinguish between noise and object boundaries. In terms of energy minimization the problem can be generally formulated as follows:

$$E(f) = E_{smooth}(f) + E_{data}(f) = \sum_{\{p,q\}\in N} V_{p,q}(f_p, f_q) + \sum_{p\in P} D_p(f_p),$$

where f is a labeling function that assigns a label to each pixel (such as a segment identification number). The so-called *smoothness term* $E_{smooth}(f)$ is a function of choice that measures the non-smoothness of the given labeling. Therefore minimization yields the better result. The *data term* $E_{data}(f)$ is another function of choice that is used to measure the incompatibility of the labeling with the evidence of the observed data. Minimization again yields a labeling that is best explained by this evidence.

The choices for the smoothness and data terms are not fully arbitrary. The computational complexity on the one hand and the implications of a given computer vision task on the other hand limit the possible number of functions, that are useful in practice. Therefore, in computer vision a general energy formulation such as the one on the right side of the equation is often used. The smoothness term consists of a penalty V comparing pairs of pixels in some neighborhood relation N. The data term is often a per-pixel or per-voxel score. For image segmentation an intuitive choice for N would be the well-known 4 or 8 pixel neighborhoods. More general, useful

neighborhood relations can be modeled as well. An example would be comparing all pairs of pixels in an image. Even higher-order neighboring relations in 3-space (voxels) can be modeled by changing N accordingly. In image segmentation problems the cost function V is often chosen to measure difference of pixel intensities.

The algorithmic complexity of finding optimal solutions to this energy minimization problem depends on several factors. Finding the globally optimal solution is NP-hard [3]. In practical scenarios efficient algorithms are therefore approximate and prone to settle in local minima that may be far from the global optimum. One of the more successful approaches is graph cuts. A major advantage of graph cuts is their polynomial complexity [14]. Furthermore it can be shown that the result is within a known factor 2 of the global minimum, which is also called 2-approx [3].

In order to apply the algorithm a few constraints limit the choice of labeling functions. The class of functions that can be minimized with graph cuts is called regular or sub-modular [27]. Furthermore the interaction penalty $V_{p,q}$ either needs to be a metric or semi-metric [3]:

$$\textit{semimetric} \quad \left\{ \begin{array}{lcl} V(\alpha,\beta) = 0 & \Leftrightarrow & \alpha = \beta \\ V(\alpha,\beta) & = & V(\beta,\alpha) \geq 0 \\ V(\alpha,\beta) & \leq & V(\alpha,\gamma) + V(\gamma,\beta) \end{array} \right\} \textit{metric}$$

Two algorithms have been proposed for either the semimetric or metric case. Before the semimetric case is discussed, graph cuts are explained in general.

Graph Cuts

Consider a weighted graph similar to the one depicted in Figure 6(a). It consists of two distinct vertices, called source (α) and sink (β), and vertices for each entity that is to be labeled. The idea is to find paths in the graph from source to sink. These paths are referred to as flow. The sum of weights of all traversed edges along the flow is called the cost of that flow. A goal is to find a path with maximal flow, i.e. no other flow has a greater cost.

Let P be a partition $P = (A, B)$ such that the source is in A and the sink is in B. Furthermore all other vertices are either in A or B. Then, the source-sink graph is basically cut in two pieces. The edges that connect the partitions are the edges that define the cut. The cost of a cut is the sum of the weights of all edges of that cut. A goal is to find the cut with minimal cost.

These two problems are dual. It means the cost of the maximal flow is equal to the cost of the minimal cut. The intuition is as follows: Consider the flow as water and the edges as pipes with different diameter. For all paths from source to sink the maximal flow is bounded by some pipe with minimal diameter, which can be thought of as a bottleneck. Therefore, the maximal flow is bounded by the sum of all bottlenecks.

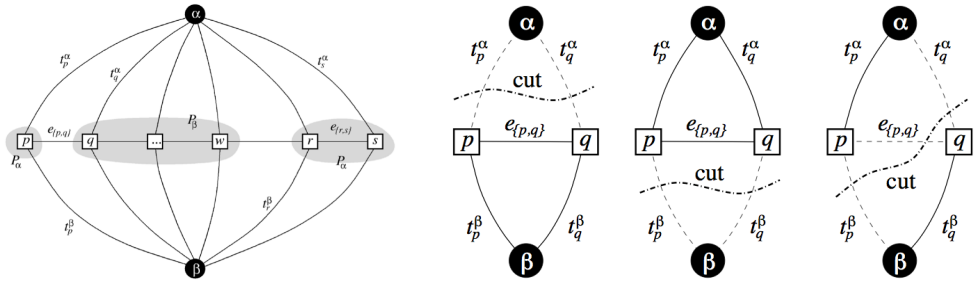
The problem of finding the MIN-CUT/MAX-FLOW can be solved efficiently in polynomial time. This makes it so interesting to be used in other fields. One of the classic algorithms in this field is from Ford Fulkerson [14].

Energy Minimization using Graph Cuts

The problem of minimizing the energy is now related to the problem of finding the minimal cut. In order to achieve this the energy term is to be transformed into a source-sink graph such that a cut indicates the assigned labels and the corresponding energy. The min-cut then naturally minimizes the energy term. The graph construction of [3] is outlined in Figure 6.

Graph Construction for Binary Labels

For simplicity assume the labels are binary, i.e. there are only two labels α and β. The undirected, weighted graph is constructed as follows: The set of vertices consists of the source α, the sink β and vertices for each entity (for example pixel) that is labeled. Each entity vertex is connected with both the source and the sink. These edges are called t-links. It is required that the solution indicates an assignment between the α- or β-link exclusively, because an entity can only be labeled once. Intuitively, the weights of these edges encode the data term. Similarly, neighborhood relations are



(a) Graph constructed for alpha-beta-swaps

(b) List of allowed cuts

edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V_{\{p,q\}}(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V_{\{p,q\}}(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{\{p,q\}}$	$V_{\{p,q\}}(\alpha, \beta)$	$\substack{\{p,q\} \in \mathcal{N} \\ p,q \in \mathcal{P}_{\alpha\beta}}$

(c) Edge weights

Figure 6: (a) Source-sink graph construction for use with alpha-beta-swaps. The terminals labeled alpha and beta correspond to the source and the sink. Edges model entity labels as well as neighboring relations. Energy penalties are encoded as edge weights. (b) Allowed cuts in the graph: Each entity vertex must be connected to alpha or beta exclusively. Only if neighboring entities are labeled differently a smoothness penalty occurs in the cut. (c) Edge weights of the graph instance for non-binary labels. *(figures from [3])*

modeled by connecting neighboring entity vertices. These edges are called n-links. Figure 6(a) portraits the 1D case: neighboring pixels at most have two neighbors. However, by adding edges between more pixels the same graph can be extended to higher-order neighboring relations. The weights of these edges naively could be assumed to model the smoothness term by assigning to them the smoothness cost of neighboring entities. This holds in the binary case, but the non-binary case requires more elaborate edge weights. An important observation is that the first condition of a *semimetric* states that the smoothness cost of equally labeled neighbors is 0. Therefore it is only required to model the case of differently labeled entities. In the binary case this cost is exactly $V(\alpha, \beta)$. The cost of a cut in the graph corresponds to the energy that arises when the entities are labeled the way the cut proposes. Observe in Figure 6(b) that there are only a few cases of possible cuts for a pair of entities. These cases arise from the facts that every entity vertex must be connected to exactly either the source or the sink and that the source and sink terminals must be separated in a cut. The first two cuts correspond to equally labeled neighbors and thus have exactly the cost of assigning either α or β to both entities. The third cut displays the case when neighbors are labeled differently and thus has the cost of the data term plus the smoothness penalty $V(\alpha, \beta)$. By induction over all pairs of entity vertices the cost of the whole cut corresponds to the energy of the induced labeling.

Non-Binary Labels

Next, a few modifications are applied in order to support non-binary labels. The entities that are equally labeled are called partitions. Consider the case when only labels are exchanged between two partitions α, β and all other labels remain the same. This is called an α-β-swap. The idea is to lift the graph construction for binary labels to support α-β-swaps and then iterate over all pairs of labels to find the minimum energy. This is outlined in Algorithm 1.

The motivation for the algorithm is the fact that α-β-swaps perform changes between two labels only and hence the graph construction is closely related to the binary label case. The data penalties of labels that are assigned to partitions other than α and β thereby do not affect the energy within

Algorithm 1 Graph-cut algorithm for energy minimization [3]

1: Start with an arbitrary labeling f
2: Set $success := 0$
3: **for each** pair of labels $\{\alpha, \beta\} \subset L$ **do**
4: Find $\hat{f} = \arg\min E(f')$ among f' within one $\square$ of f
5: **if** $E(f') < E(f)$ **then** set $f := \hat{f}$ and $success := 1$ **end if**
6: **end for**
7: **if** $success = 1$ **then goto** 2 **end if**
8: **return** f

with $\square \in \{\alpha\text{-}\beta\text{-swap}, \alpha\text{-expansion}\}$

one swap away. Similarly, the smoothness penalties of (γ, δ)-neighbors with $\gamma, \delta \notin \{\alpha, \beta\}$ do not affect the energy as well. Thus, the key observation of [3] is that the cost of a graph, that does not model these penalties, is a constant K away from the total energy. Since the constant is same for all cuts of a given graph instance the min-cut also minimizes the total energy.

However, the smoothness penalties play a role for (α, γ)- and (β, γ)-neighbors, if their costs differ. More formally, it is possible that $V(\alpha, \gamma) \neq V(\beta, \gamma)$ for $\gamma \notin \{\alpha, \beta\}$ holds. These costs are caused from the assignment of a specific label, either α or β, to some entity and are not related to (α, β)-neighbors at all. Therefore it is intuitive not to encode these neighboring costs into the n-links that in the binary case are used for costs of (α, β)-neighbors, but instead into the t-links. The graph construction for the non-binary case is now complete with the final edge weights given in Figure 6(c).

Note, another algorithm has been proposed for *metric* energy functions in [3]. It differs only in the graph construction such that in each iteration the energy is compared to the energy within one so-called α-expansion away from the current state. An α-expansion is a more general move than the α-β-swap in that it allows arbitrary labels to become α labels. Then, the algorithm converges to the optimal approximate solution faster. However, for details the reader is referred to [3].

Also note the reason why energies have to be regular/sub-modular is because the algorithm iterates over all pairs of labels in an arbitrary order. Therefore it is required that the order of pairs does not affect the algorithm in converging to the same solution. For more details on regularity conditions refer to [27].

2.4 Multi-View Stereo

Given images with known camera parameters, multi-view stereo (MVS) is the problem of reconstructing a 3D scene without changing the camera parameters. Typically, MVS algorithms produce a dense scene reconstruction and result in well-defined volumes. Different algorithms have been created to achieve this task. Seitz et al. [36] propose a taxonomy to compare and evaluate them against their sets of ground-truth images. The taxonomy categorizes MVS algorithms depending on six key properties:

Initialization requirements are often encountered properties of algorithms. Seitz et al. [36] state many MVS algorithms require a bounding box and some require foreground/background segmentation. Those that work with depth maps sometimes constrain the range of depths. In other words the scene is initialized with near and far planes for each view.

Scene representations include *voxels, level-sets, polygon meshes and depth maps.* Recently, the scene has been represented as small patches, called *surfels* [19]. These are especially of interest because PMVS (cp. Section 3.2), which is the MVS algorithm used in [16], is based on surfels.

Photo-consistency measures are measures for the visual compatibility of the reconstruction with the set of input images. Variants are classified as measures in *scene space* or *image space.* Scene space measures project scene structures into the views and then measure the agreement in some way while image space measures solely rely on the image information. A notable example is normalized cross correlation (NCC). NCC is accentuated here, because it is used in [16]. Other

methods warp images to other views and compare them, which is called prediction error.

Visibility model: A major advantage of using multiple views over using only few views is that the probability increases that occluded objects are visible in another view. To make use of these cues visibility models have been developed that act as a filter on which views are to be considered during photo-consistency evaluation. These models are categorized as *geometric, quasi-geometric* and *outlier rejection* methods. Geometric models explicitly model the image formation process and shape of the scene. Quasi-geometric models rely on approximate geometric reasoning, such as analyzing only clusters of nearby views. Outlier rejection methods assume that occlusions appear infrequently. Simple outlier rejection techniques can be used under this assumption. Another simple heuristic is to avoid comparing views that are far apart.

Shape priors influence how and what type of scene structure is reconstructed. Methods that have a prior on the overall size of a surface are categorized as *minimal* or *maximal surfaces* , which indicates the preference of those methods. Other methods that typically use depth maps for scene representation often optimize an image-based smoothness term, because surfaces should be locally smooth. These methods can be formulated naturally with 2D MRFs. This is especially memorable, because Manhattan-World stereo [15] works this way (cp. Section 4). It uses a strong prior that assumes planar surfaces (especially walls) and poses the problem of generating smooth depth maps with non-smooth edges along plane boundaries as a 2D MRF.

The **reconstruction algorithm** is a fundamental property of MVS algorithms. Reconstruction algorithms can be divided into four classes. One class operates *iteratively* and in each iteration a cost function is tried to be decreased. An example are carving methods that initialize a bounding box of voxels and carve out pieces iteratively until a cost function converges. Another class computes *depth maps* instead of volumes and possibly merges them into a 3D scene as a post process. Third, *feature extraction and matching* techniques are applied and the reconstructed features are used as indicators for a *surface fit*. Fourth, a *cost function over a 3D volume* is defined and a surface is extracted from the volume of minimal cost. Some formulate the cost as a 3D volumetric MRF and use graph cuts. [16] also poses such an MRF in a final stage (cp. Section 5.1). The optimal surface of the graph cut is then extracted into a mesh model (cp. Section 5.2).

2.5 Delaunay Triangulation

With triangulation in computer graphics the goal is to connect a given set of points of a plane to a convex hull of triangles, where each point is a joint of a triangle (called triangle meshes). Delaunay triangulation [11] aims at fulfilling the following condition: For every (non-degenerate) triangle the circle circumventing its three vertices does not contain any other vertex other than those three that define it. This avoids sharp triangles, which yields nicer and numerically stable results in practical applications.

There are several techniques that calculate meshes that fulfill the Delaunay condition. Basic methods use flipping techniques. Figure 7 gives an example of how two adjacent triangles, which do not fulfill the Delaunay condition are fixed with a flip operation. Flip algorithms have a runtime of $O(n^2)$, but more efficient algorithms exist that run in $O(n \log n)$.

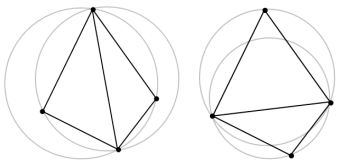


Figure 7: Flip operation for Delaunay triangulation *(figures from [10, 9])*

Delaunay triangulation is relevant in Section 5.2, where a triangle mesh is extracted by triangulating slices of a voxel grid with surface annotations. There, a constrained Delaunay triangulation is applied: Vertices and edges are given as input to the triangulation that are required to be part of the resulting mesh. Here, the Delaunay condition is weakened, because the result may not always be convex. However, important structures such as boundaries and holes can be preserved.

3 Dense Scene Reconstruction from Images

This section explains the initial stages of the reconstruction algorithm for building interiors. In the first stage the set of input images is processed in order to obtain the intrinsic and extrinsic camera

parameters of each taken photograph. Feature extraction and matching techniques produce feature correspondences between the views. This allows to solve for the epipolar geometry. Structure from motion is applied such that all of the views agree with the reconstruction with minimum reprojection error. The result is a set of calibrated cameras and a sparse scene reconstruction. These tasks are performed using a software package called BUNDLER [40, 4] (cp. Section 3.1).

In the second stage a dense reconstruction is obtained using multi-view stereo. The algorithm used in this stage is called patch-based multi-view stereo (PMVS) [19, 33] (cp. Section 3.2).

3.1 Bundler

BUNDLER [40, 4] is a structure from motion software that is specifically designed to run on unordered and unstructured sets of images. These kind of photographies are typically found in online image databases such as flickr or Google images[1]. The ultimate intent is to use the power of all the images on the web and reconstruct large-scale scenes such as whole cities [1]. Arranging the scene images in three-space, such as described in [40, 39], promises novel ways of browsing photo collections. The algorithm basically consists of three stages.

At first image features are extracted from every input image using a technique called scale-invariant feature transform (SIFT) [28]. SIFT is a popular technique that extracts local features in a scale-invariant manner (more specifically blob-like structures) and then produces a feature vector. Its major benefits are robustness and invariance to numerous types of image distortions such as scale-change, rotation, translation, small pixel shifts and changes in lighting conditions. This makes it well-suitable to the different kinds of images that are found on the Internet. On a side note, due to the nature of local features the final reconstruction typically becomes sparse. Only to some extent this is balanced by the amount of input views.

Secondly, feature vectors are matched using nearest-neighbor search, because the feature vectors form a vector space. Corresponding features of image pairs are used to estimate the epipolar geometry (cp. Section 2.1) using the well-known 8-point algorithm [23]. The fundamental matrices are estimated in a RANSAC [13] framework, which makes the estimation more robust to outliers. Furthermore, matching features are organized into tracks, which the authors define as connected sets of matching keypoints across multiple images [40].

Thirdly, incremental structure from motion is applied for each track. This means a well-conditioned initial pair of cameras is estimated and then another camera is added to the optimization at a time. The cameras are chosen in an intelligent fashion. The next camera is selected that observes the largest number of tracks with reconstructed points. Once all cameras of a track are estimated the next track is added. Finally, bundle adjustment [12] minimizes the reprojection error.

3.2 Patch-based Multi-View Stereo

Patch-based multi-view stereo (PMVS) [19] is a MVS algorithm that is used in the second major stage for reconstructing building interiors. Its principle of operation is briefly outlined.

Recall that in the last stage camera parameters and sparse scene structure have been estimated. In this stage the goal is a dense reconstruction. PMVS takes as input the set of calibrated camera parameters and the image views. The output is a dense set of extracted surfels, i.e. small rectangular patches with a surface normal. PMVS also reports the photo-consistency score of a surfel and which views a surfel is visible from. The algorithm is basically divided into three major stages, called *match, expand and filter*.

At first corresponding local features between images are acquired. PMVS extracts features using the multi-scale Harris detector [30] and matches them in a special way: Features are matched that lie within 2 pixels from the corresponding epipolar lines. From these features the corresponding point in 3-space is estimated. Surfel candidates are then generated at each point one by one starting with the nearest to the farthest. Here, p denotes the surfel patch, $c(p)$ denotes the reconstructed point in 3-space, $n(p)$ the patch normal and $R(p)$ a reference image, that the patch belongs to. The

[1] http://www.flickr.com, http://images.google.com

surfel normal $n(p)$ is important for Manhattan-world stereo (cp. Section 4). Initially, it is defined as the normalized vector from surfel position to the reference camera's optical center. The set of images, which a patch is visible from and denoted $V(p)$, is derived with the help of comparing the surfel normal to the vector from surfel position to other camera's optical centers. In a final stage the normal vector is optimized over all views in $V(p)$ and chosen as the one with the best photometric consistency score. The photo-consistency score is obtained using normalized cross correlation of the unit surfel that is projected into the reference camera's view and another given view. Note, the projection differs depending on the surfel normal that is varied.

In the second stage a given set of patches is expanded in order to arrive at a more dense reconstruction. This works in two stages. First, cells are identified that need to be expanded. Note that the input images are divided into a regular grid of cells, i.e. cells of size 2×2 [19]. Expansion candidates are the neighboring cells of the cell that the patch center projects to. There is no need for expansion if the cell already contains another neighboring patch p', where the neighboring relation is defined in 3-space as

$$|(c(p) - c(p')) \cdot n(p)| + |(c(p) - c(p')) \cdot n(p')| < 2\rho_1 \tag{2}$$

for some constant ρ_1. Also, there is no need for expansion if the cell contains a patch with a photo-consistency score distant from the one of p. The intuition is that there is a depth discontinuity, for which expansion would not be appropriate. Next, the actual expansion takes place, which is not detailed here. Suffice to say new surfels are generated, then optimized using some dedicated procedure and the set of visible images $V(p')$ is updated according to some rules.

Third, erroneous patches are filtered out. Three filters decide:

1. If two patches p, p' are not considered neighbors for a given image (equation (2)), although they project to the same cell in another view v, with $v \in V(p)$, then p is possibly in a visibility conflict. p is filtered out, if additionally an inequality holds over photo-consistency.
2. The second filter is also a visibility filter, where a minimum number of images, in which the currently considered patch is visible, have to fulfill some depth map test.
3. The third filter compares the magnitude of $V(p)$ between neighboring cells.

By repeating expansion and filtering more dense structures can be computed. In the standard setting PMVS repeats these stages three times. For more detailed insight into PMVS refer to [19].

4 Manhattan-World Depth Maps

In this section the stage of reconstructing per-view depth maps from the output of the multi-view-stereo algorithm is detailed. To this end, the Manhattan-world assumption is exploited. It states that all surfaces in a scene are aligned with three dominant orthogonal directions. As a consequence the scene is piecewise axis-aligned-planar [15]. These dominant directions are first estimated (cp. Section 4.2). Then, each given 3D point implicitly casts a vote for the three possible axis-aligned planes that the point may lie on. These votes are clustered and peaks indicate hypotheses for evident planes (cp. Section 4.3). Finally, the aim is to provide depth maps for each view. These are calculated using Markov random fields and graph cuts (cp. Section 4.4).

4.1 Multi-View Stereo Preprocessing

Initially, a few modifications are done for working with the PMVS algorithm. "The initial photo-consistency threshold is set to 0.98, which PMVS iteratively relaxes to 0.65" [15]. From the results of PMVS only points are kept that are seen by at least three views. Points in nearly texture-less regions are rejected by projecting them into their visible views and comparing the local image region. A point is rejected if the average standard deviation over all visible images in a 7×7 window around the projected point is below a threshold $\tau = 3$. Furthermore a 3D sampling rate R is calculated as the average foreshortened diameter of spheres around all points that project to unit circles in the views. That is, the projected diameter of a sphere equals the pixel spacing of

that image. In order to arrive at a foreshortened diameter it is weighted with the dot product of the surfel orientation and the viewing direction.

4.2 Estimation of Manhattan-World Axes

Since the Manhattan-World assumption states that three main, orthogonal axes exist in the scene, these axes are recovered at first. To this end, a histogram is built of normal directions of all the oriented points. The histogram covers the range over a unit hemisphere, i.e. half a sphere, because normals that are opposite to each other indicate the same axis. Furthermore, the histogram is divided into 1000 bins [15].

Having created the histogram of normal directions, recovery of the Manhattan-World axes is performed by histogram analysis. Maxima in the histogram indicate dominant axes. The first main axis is defined as the average normal of the largest histogram bin. In order to enforce perpendicularity, the second axis is found in bins that are 90 degrees away from the first axis. The third axis, respectively, is found in bins that are 90 degrees away from the first and second axes. Due to numerical errors of real-world data the bins do not have to be perfectly perpendicular. Instead, the bins are chosen that are between 80 and 100 degrees away. As an interesting side note the authors state in their experiments axes vary only within 2 degrees from the ideal 90 degrees [15].

4.3 Estimation of Candidate Planes

In the previous stage the three Manhattan-World axes have been estimated. In this stage the second statement of the Manhattan-World assumption is focused on, which is that the scene basically consists of planes that are aligned with the main axis. These planes are estimated in the following way. Given the position of a 3D point P and a surface normal $\vec{d}$, the plane equation is:

$$\vec{d} \cdot (P - X) = 0 \Leftrightarrow \vec{d} \cdot P = \vec{d} \cdot X, \tag{3}$$

where X denotes an arbitrary point on the plane and $\cdot$ denotes the dot product. The left equation is explained by the fact that the surface normal and any vector pointing within the plane are perpendicular and the dot product of perpendicular vectors is zero. From the right equation one can see that the dot product of surface normal and points that lie on the same plane has the same value. This value is a unique offset for planes with fixed surface normal. To illustrate this, consider a point D on the axis that is parallel to the normal n of some plane with offset off. D is determined by stretching the axis vector with some factor $\lambda \cdot n = D$. Inserting this into (3) gives:

$$off = \vec{n} \cdot D = \vec{n} \cdot (\lambda \cdot \vec{n}) = \lambda \cdot (\vec{n} \cdot \vec{n}) = \lambda \cdot ||n||_2^2 \overset{||n||=1}{=} \lambda$$

This points out that the offset is actually the distance of the plane to the origin iff. normalized normals are used. With unnormalized normals the term index is more appropriate than distance.

Now a way is established of assigning points to planes that are aligned with one of the three Manhattan-World axes. The next step is to find planes with sufficient evidence. Again histograms are used. For every axis direction $\vec{d_k}$ the histogram contains the plane offsets of all 3D points. Mean shift clustering [6] is used to find clusters and peaks in the histogram. The offsets of the peaks indicate planes with sufficient evidence. As a lower bound on the evidence clusters need to have at least 50 samples. Furthermore, the bandwidth σ of the mean shift algorithm influences the number of windows that are shifted to find clusters and thus influences how many clusters are found, at all. σ is chosen to be R or $2R$ (cp. Section 4.1). The evident planes are forwarded together with their normals to the next stage. Note, two equal planes with opposite normals are forwarded, because oriented planes are desired for visibility reasoning.

4.4 Reconstruction of Depth Maps

The next stage of the algorithm computes depth maps of the previously generated Manhattan-World scene, i.e. given the set of hypothesis planes each pixel of each view is assigned to exactly

one plane. This problem is formulated as a 2D MRF [15]:

$$E = \lambda \sum_{\{p,q\} \in N_4(p)} E_s(h_p, h_q) + \sum_{p \in P} E_d(h_p),$$

where h is a labeling function that assigns a plane hypothesis to each pixel and $N_4(p)$ is the 4 pixel neighborhood in 2-space. A scaling factor λ is chosen for experimental reasons. The values are described in more detail in [15]. It remains to define the data and the smoothness terms.

The data term $E_d(h_p)$

Recall that the data term measures the disagreement between the labeling function and the observed data. Since we seek to recover depth maps, where each pixel indicates the depth of the surface that is visible from the pixel, a natural choice of a disagreement function is a measure of visibility conflicts. To this end, the label assignments are related to the 3D MVS points.

Let $\{P_i\}$ denote the set of all MVS points, $P \in \{P_i\}$ be a specific MVS point and V_P the set of images P is visible from. Given a target depth map M and a labeling $h_p = l$ that assigns plane l to pixel p, let $Plane$ be the 3D point on the plane that projects to p. Visibility conflicts naturally occur when a point is occluded by another. In order to recognize occluding points they are projected onto the depth map. Let $\hat{\pi}_M(P)$ be that projection rounded to the nearest pixel coordinate in M. If the projections of two points equal, they lie on the same viewing ray. The points occlude each other except when two points are close enough to each other. In this case they are assumed to be the same. The proximity of two points P, Q is measured with respect to the depth of P as

$$Dist_M(P, Q) = (Q - P) \cdot \frac{O_M - P}{||O_M - P||},$$

where O_M is the optical center of the depth map M. Note that $Dist_M(P, Q)$ is a signed function and the sign indicates which of the points would occlude the other. Due to different sources of error, such as noisy measurements, erroneous points or camera calibration errors, a threshold $\gamma = 10R$ (cp. Section 4.1) is defined. The threshold defines no-conflict regions around a point. Visibility conflicts are categorized into three cases:

Case 1: Visibility conflicts basically arise from the following implications. A point P is visible in the depth map M of interest, iff. $M \in V_P$ and therefore must not be occluded. Let $h_p = l$ be a labeling for some pixel p within M. Then the corresponding point $Plane$ must also be visible in M. Let $|Dist_M(P, Plane)| > \gamma$ such that the points are outside the no-conflict region. Then, the conflict is given, iff. both points lie on the same viewing ray, i.e. they project to the same pixel coordinate $\hat{\pi}_M(P) = \hat{\pi}_M(Plane) = p$.

Case 2: Here, let $M \notin V_P$. Now P may be occluded in M. However, $Plane$ still must not be occluded by P or else the labeling is contradictory. If $\hat{\pi}_M(P) = p$ and $Dist_M(P, Plane) > \gamma$ the labeling h_p is declared as conflicting. Note, the only difference compared to the first case is that the sign of the depth distance function $Dist_M(P, Plane)$ is used to differentiate whether P occludes $Plane$ (conflict) or vice versa (no conflict).

Case 3: A third visibility conflict occurs if the 3D point $Plane$ occludes a point P that is visible from another view $N \neq M$. This case shares properties with the first two cases. Similar to the first case the visibility conflict results from $Plane$ occluding some point. However, here $Plane$ does not lie on a viewing ray relative to M. Similar to the second case the plane hypothesis is related to information not apparent in the target view. However, here the conflict arises from $Plane$ occluding P, which is reverse in the second case. More formally, the following must hold: $\forall N \in V_P, N \neq M : \hat{\pi}_N(P) \neq \hat{\pi}_N(Plane)$.
The no-conflict region is modified in this case such that $Dist_N(P, Plane) < -\hat{\gamma}_{P,N}$ holds with a new threshold $\hat{\gamma}_{P,N} = \gamma / |n_{h_p} \cdot r_N(P)|$. Here, n_{h_p} denotes the normal associated with the hypothesized plane and $r_N(P)$ denotes the normalized viewing ray vector from the optical center of the view N towards the point P. Note that the normal has been forwarded in the previous stage when candidate planes were estimated. The authors argue this modification

is necessary, "because the visibility information becomes unreliable when the corresponding visible ray is nearly parallel to both the image plane and the plane hypothesis" [15].

Finally, each individual conflict is summed over all pixels of all MVS points and depth maps [15]:

$$E_d^i = \begin{cases} \max(0, C(P_i) - 0.7) & \text{if } h_p \text{ conflicts with } P_i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad E_d(h_p) = \min(0.5, \sum_i E_d^i(h_p))$$

Here, $C(P_i)$ is the photo-consistency score from PMVS. 0.7 is a lower-bound, which is used to ignore small conflict penalties, and 0.5 is an upper-bound, which is imposed for robustness in the sense that single individual conflicts should not affect the overall cost arbitrarily much.

The smoothness term $E_s(h_p, h_q)$

Recall, the smoothness term measures smoothness of pairs of pixels. The neighborhood relation that is used in this algorithm is a 4 pixel neighborhood. The intuition for smooth pixels in Manhattan-World depth maps is that plane hypotheses of neighboring pixels should be nearby planes and ideally the same plane. Though, a reasonable exception of smoothness occurs when two planes meet. Since Manhattan-World is assumed, the intersection line of these planes has a direction that is parallel to one of the major axes. Such a line is called a dominant line. Therefore the individual contribution of a pair of pixels to the energy minimization term consists of a measure of distance of the hypothesized planes and a weighting factor that relaxes the distance at dominant lines.

The plane distance $\Delta_s(h_p, h_q)$ is defined on the viewing ray that is shot through the midpoint between p and q. Although this measure is not as natural as back-projecting p and q on their respective hypothesis planes and then taking the distance of the 3D points, the used measure is sub-modular, which is required for efficient graph cuts to work (cp. Section 2.3).

In order to calculate the weighting factor a method to identify dominant lines has to be established first. On the one hand dominant lines are parallel to one of the major axes. On the other hand parallel lines in 3-space meet at the same vanishing point v in image space. Vanishing points v_k are estimated for each dominant axis $k \in \{1, 2, 3\}$. Then, the direction in image space of a dominant line passing through pixel p is given by $\vec{l_k} = v_k - p$. Furthermore, considering early vision techniques, the gradient is exploited to find lines. Recall that the contrast change is large for lines in the direction orthogonal to the actual direction of the line. Therefore the gradient response is also large with respect to the orthogonal direction denoted $\vec{l_k^\perp}$. The strength of an edge along $\vec{l_k}$ is measured as:

$$e_k(p) = \sum_{p' \in w(p)} || \bigtriangledown_{\vec{l_k^\perp}} I(p')|| \;\; / \sum_{p' \in w(p)} || \bigtriangledown_{\vec{l_k}} I(p')||.$$

The authors note that the gradient is calculated via bilinear interpolation and finite differences. That means pixel values are interpolated in a bilinear fashion from a set of four pixels, because the image space is discrete and in two dimensions. Finite differences are typically used as an approximation of image derivatives of the unknown continuous image signal. $w(p)$ furthermore denotes a local image window of size 7×21, elongated along the $\vec{l_k}$ direction [15].

The overall contribution to the smoothness term can now be defined as:

$$s(p) = \begin{cases} 0.01 & \text{if } \max(e_1(p), e_2(p), e_3(p)) > \beta \\ 1 & \text{otherwise} \end{cases}, \quad E_s(h_p, h_q) = \min\left(10, s(p)\frac{\Delta_s(h_p, h_q)}{R}\right).$$

Here, $s(p)$ decreases (relaxes) the energy contribution if a line response is encountered in a dominant direction, which is larger than a threshold $\beta = 2$. The authors note the value of 2 "corresponds to an edge being within 25 degrees of any dominant line direction" [15]. Furthermore, the authors state that a small value of 0.01 is chosen for relaxation instead of a value of 0 in order to constrain the energy minimization in cases where the data term is 0. Also the distance measure is scaled with a factor R (cp. Section 4.1), and an upper-bound of 10 is deployed for reasons of robustness.

5 Depth Map Integration

In the last stage the reconstruction algorithm [16] uses the estimated depth maps as input and defines a cost volume over voxels with binary labels. The problem of finding the minimal surface then can be formulated as an energy minimization term in a volumetric MRF framework, which is solved using graph cuts. In this section the volumetric cost function is first motivated.

Consider an object in 3-space whose surface is to be defined. An implicit representation of the surface can be obtained as follows: Given a voxel grid of resolution N_r and a binary labeling $\{interior, exterior\}$ for each voxel that denotes whether a voxel is inside or outside the volume of the object. Then, the surface is implicitly defined at neighbors of unequally labeled voxels, i.e. at the volume boundaries. The first objective is to recover the optimal labeling (cp. Section 5.1). Given the labeling the surface is transformed into a format that is more suitable for rendering purposes, such as a triangle mesh (cp. Section 5.2). Finally, efficiency enhancements are explained (cp. Section 5.3). The results are shown separately in Section 6.

5.1 Obtaining the Minimal Volume

Let $\{D^1, D^2, \ldots\}$ be the set of depth maps that are provided as input. Given a depth map D^i and a voxel v, let p_v denote the nearest pixel on D^i close to the projection of v (more precisely its voxel center). Figure 8(a) visualizes the notation that is used here. Note, the authors originally use p_v to both denote the pixel on the depth map as well as the corresponding point in 3-space. In order to avoid confusion let P_v explicitly denote this scene point. Let $\bar{v}$ denote the voxel behind P_v. Furthermore $\Omega(p_v)$ denotes the set of voxels along a ray from P_v to the optical center of the depth map D^i. $d^i(v)$ denotes the depth of v with respect to D^i. The idea is to let each depth map vote for the labeling as follows: $\Omega(p_v)$ are voxels in the free space along the eye-ray and are therefore labeled exterior. $\bar{v}$ is labeled interior, because it is behind the known point P_v of the surface. Implicitly the surface is represented by the voxel labeling in places where interior and exterior voxels are neighbors. This summarizes the basic idea and notation that is used for the volumetric reconstruction of the surface given a set of depth maps.

Next, a volumetric energy minimization term is defined:

$$E(f) = E_{smooth}(f) + E_{data}(f) = \sum_{\{u,v\} \in N_6} V_{u,v}(l_u, l_v) + \sum_{v \in VOXELS} D_v(v).$$

Since the objective here is a volume of voxels, the neighboring relation N_6 denotes the voxel neighbors along each of the coordinate axes.

The smoothness term

The *smoothness term* is defined as [16]: $V_{u,v}(l_u, l_v) = \delta(l_u = l_v) = \begin{cases} 0 & , l_u = l_v \\ 1 & , l_u \neq l_v \end{cases}$, where δ is the well-known Kronecker-Delta, given at the right side of the equation.

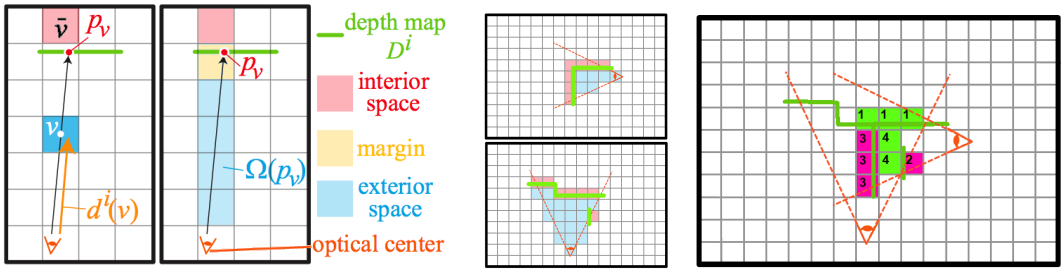
The data term

Recall, the *data term* describes disagreement of the labeling function with the observations. Thus, for a given labeling the disagreements of all depth maps over a voxel label are summed [16]:

$$I(v) = \sum_{i=1}^{N_c} I^i(v), \quad E(v) = \sum_{i=1}^{N_c} E^i(v), \qquad \text{with}$$

$$I^i(v) = \begin{cases} 1 & , \text{if } 0 < d^i(v) - d^i(p_v) \leq \mu \\ 0 & , \text{else} \end{cases}, \qquad E^i(v) = \begin{cases} 1 & , \text{if } 0 < d^i(v), d^i(v) - d^i(p_v) \leq -2\mu \\ 0 & , \text{else} \end{cases}.$$

Here, μ is set to the voxel resolution N_r. This definition basically formalizes the algorithm idea as outlined earlier: A depth map votes for a voxel being interior (or full), if the voxel is behind



(a) Notations for visibility reasoning

(b) Two views voting for same voxels

(c) Disagreement cases of bottom view

Figure 8: (a) Notations used for visibility reasoning. A depth map votes for voxel labels. (b) Two depth maps that see the same voxels (note, for simplicity the margin is omitted). (c) The two depth maps agree and disagree over some voxel labels. The numbers correspond to the disagreement cases (see text) with respect to the bottom depth map. *(figures adapted from [16])*

the surface, i.e. the depth of the voxel $d^i(v)$ is close to the surface depth $d^i(p_v)$ with respect to some tolerance μ. Note that only a single voxel is voted interior along an eye ray of a single pixel, because the evidence of a depth map ends at the surface. For instance, if there is a wall the depth map does not tell its thickness. It can only be recovered if the depth map can be related to another depth map that informs about the other side of the wall. In contrast, a depth map votes for exterior (or empty), if the voxel lies in the free space of the eye ray between optical center and the surface, i.e. the depth of the voxel is positive and is at least 2μ units distant from the surface, where the constant factor 2 is chosen as an error margin due to errors in depth maps. Note that $I(v)$ and $E(v)$ do not count disagreements, yet. Moreover they simply count the votes of all depth maps for a voxel being interior and exterior. Costs for disagreements could be defined as $D_v(l_v = interior) = E(v)$ and $D_v(l_v = exterior) = I(v)$.

Due to inconsistencies between the depth maps these terms do not suffice, yet. Weighting factors are added in order to account for disagreements between depth maps themselves [16]:

$$D_v(l_v = interior) = \sum_{i=1}^{N_c} \omega^i(p_v)\psi^i(v)E^i(v) \qquad D_v(l_v = exterior) = \sum_{i=1}^{N_c} \omega^i(p_v)I^i(v)$$

where $\omega^i(p_v)$ measures the consistency of depth maps, while $\psi^i(p_v)$ is a correction term that reduces the effects of distant depth maps for exterior evidence. More precisely [16]:

$$\omega^i(p_v) = exp\left[\frac{I(\bar{v})}{8} - \lambda_2\left(E(\bar{v}) + \sum_{v' \in \Omega(p_v)} I(v')\right)\right], \quad \psi^i(v) = \min\left(1, exp\left[-\frac{d^i(p_v) - d^i(v) - 2\mu}{8\mu}\right]\right).$$

In order to explain these factors the cases of disagreement are figured first. The geometry of interest consists of interior and exterior votes along a ray from optical center through some point in 3-space (cp. Figure 8(a)). Let the term target interior vote and target exterior votes denote these votes. The according depth map is called target map. Take another depth map as a whole, then all of these interior and exterior votes yield the interior and exterior space, respectively, of that depth map. This depth map is called alien map. The cases of agreement or disagreement between the target map and the alien map are considered (cp. Figure 8(c)):

Case 1: The target voxel is in the interior space of the alien map: The maps agree.
Case 2: The target voxel is in the exterior space of the alien map: The maps disagree.
Case 3: A target exterior voxel is in the interior space of the alien map: The maps disagree.
Case 4: A target exterior voxel is in the exterior space of the alien map: The maps agree.

Note, the authors do not use the fourth case of agreement. More formally, the term $I(\bar{v})$ is a count of the number of all depth maps that also provide evidence of $\bar{v}$ being interior, i.e. the

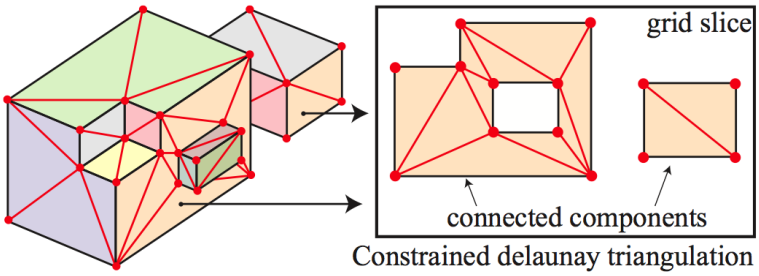


Figure 9: Mesh extraction from connected components on voxel slices *(figure from [16])*

number of times $\bar{v}$ is in the interior space of an alien map. Intuitively it is the agreement of a surface part. This yields the first case. Similarly $E(\bar{v})$ counts the number of depth maps, where $\bar{v}$ is in the exterior space. This yields the second case. Finally, the term $\sum_{v' \in \Omega(p_v)} I(v')$ counts the number of third case disagreements. The authors have furthermore experimented with some scaling factors for the agreements and disagreements. Some choices and results for λ_2 are given in [16]. The choice for the signs in ω^i is trivial: The exponential function is positive and ever-expanding, so agreements have to be added in order to get a high weighting and disagreements are subtracted in order to get a low weighting.

Since $\omega^i(v)$ captures already agreement and disagreement over both the interior and exterior space, it is used as a weighting factor for both cases in $D_v(l_v)$. However, a different weighting scheme, which the authors have not used, seems also reasonable: The cost for exterior disagreements in $D_v(l_v = interior)$ could be weighted with a weighting factor specific to exterior space and vice versa. So instead of defining one term, which can be outlined in a pseudo-fashion as $\underbrace{agree_{int}}_{\text{case 1}} - (\underbrace{disagree_{int}}_{\text{case 2}} + \underbrace{disagree_{ext}}_{\text{case 3}})$, two terms could be defined similar to $\underbrace{agree_{int}}_{\text{case 1}} - \underbrace{disagree_{int}}_{\text{case 2}}$ as a weighting factor for $I^i(v)$ and $\underbrace{agree_{ext}}_{\text{case 4}} - \underbrace{disagree_{ext}}_{\text{case 3}}$ as a factor for $E^i(v)$.

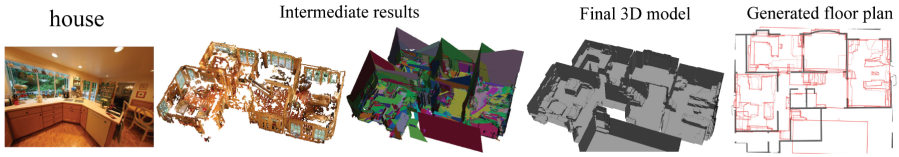
Finally, the term $\psi^i(v)$ is used to adjust the evidence for exterior votes $E^i(v)$ to the same units of interior votes $I^i(v)$. The authors argue that "interior evidence lies immediately behind depth maps, and hence, covers a 2D manifold", which is a plane embedded in 3-space, "while exterior evidence covers a 3D space in front of the depth map" [16].

The authors furthermore note, that the minimum of the energy function is often not unique. In these cases the min-volume and the max-volume have the same energies with N_6 neighboring relations. However, the max-volume tends to be jagged while the min-volume is smoother in general (cp. Figure 10(c)). Therefore, a small penalty of 10^{-6} is added to $D_v(l_v = interior)$ to find the min-volume solution. The authors point out this results in "small numbers of clean, sharp corners in areas where no observations drive the surface" [16].

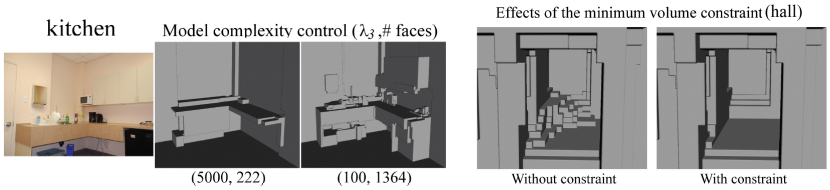
5.2 Mesh-Extraction

Now that the minimum volume solution is given, a mesh is extracted, which is more suitable for rendering purposes. Figure 9 illustrates the idea: The voxel grid is cut into slices along the voxel boundaries and for each of the dominant axes. The surface boundaries are determined on the slice based on the voxel labels. The surface boundary is given, where incident voxels have different labels. Given the boundaries of surfaces, connected components are further identified. Then a mesh is generated for each of these connected components using constrained Delaunay triangulation (cp. Section 2.5). The boundaries of the connected component form the constraints of the triangulation.

The mesh is then refined for every connected component with sub-voxel accuracy, which consists of two stages: At first, relevant MVS points are collected. These have to be within 1.5μ of the connected component and their normal needs to be within 45 degrees of the normal of the connected component. Secondly, the signed distances to the centroid of the MVS points are computed. Each

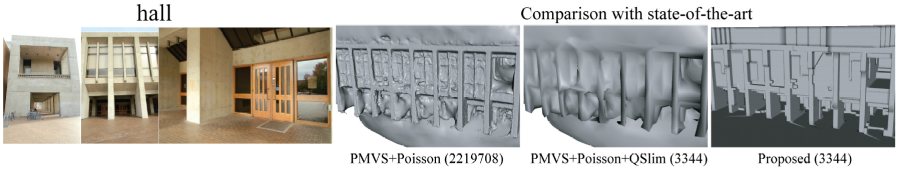


(a) *House* scene with intermediate results and ground plan



(b) *Kitchen* scene results with different level of detail

(c) Effects of preferring the min-volume



(d) *Hall* scene compared to state-of-the-art MVS algorithms

Figure 10: (a) *House* scene with intermediate results of PMVS and Manhattan-world depth maps (one color for each view). Also the final model and the automatically generated floor plan. (b) *Kitchen* scene with different level of detail. (c) Preferring the min-volume over the max-volume removes jagged surfaces in some areas. (d) *Hall* scene compared to state-of-the-art MVS algorithms. Note the clean results of the proposed reconstruction algorithm. *(figures adapted from [16])*

vertex is then shifted by this distance in the normal direction. After all connected components on all slices for all dominant axes have been processed the resulting axis-aligned mesh is final.

5.3 Enhancements

The authors propose three enhancements [16] to the procedure that is outlined above.

First, in order to accelerate the reconstruction and reduce memory footprint, some slices are pruned. The intuition is that most important slices are defined by the number of nearby scene corners and edges (grid pixels) within $\mu/2$ distance of the slice away. If this number is less than a threshold λ_3, the slice is removed from consideration.

Secondly, the ground plane is estimated for cases when many horizontally taken images provide few evidence for the ground. Among all six directed dominant directions the one is chosen as up-vector that is most compatible with the camera up-vectors. The ground plane is then identified among horizontal slices as the bottom-most one with a number of grid pixels larger than average.

Thirdly, all voxels at the boundary of the voxel grid are marked interior, so that always a bounding box is obtained, which can be used for image-based rendering purposes.

6 Results

The reconstruction algorithm for building interiors has been tested on four datasets, namely *kitchen, hall, house, gallery*, on a "dual quad-core 2.66Ghz PC" [16]. These datasets typically have an image resolution of 2-3 Megapixels. The running time depends on the number of images. For the *kitchen* scene with 22 images the total run time is 91 minutes while the *hall* dataset with 97 images requires 515.7 minutes. The largest dataset *gallery* with 492 images requires a few days (6545.8 minutes). However, with pruning certain slices (cp. Section 5.3) the running times can be

drastically reduced to a few minutes even for such large scenes. The detailed running times indicate that the Manhattan-world stereo approach is the bottleneck of the system, which consumes most of the time. However, the authors anticipate that this step can be parallelized in order to compute each depth map individually.

Figure 10(a) shows results for the *house* scene. The reconstruction algorithm successfully recovers large-scale structures such as walls, staircases and furniture. Despite the sparse output of the MVS algorithm and the noisy depth maps the final 3D model is remarkably clean. A floor plan can be generated automatically by projecting the walls to the ground with the pixel intensity indicating the amount of projected surface.

Figure 10(b) shows results for the *kitchen* scene with varying choices for λ_3. Note, that the dominant features of the scene are recovered even for rather large thresholds. Also note how detailed features re-appear for smaller thresholds. Irregular structures that are not piecewise planar and aligned with the axes are approximated with axis-aligned bounding boxes.

Figure 10(c) gives an example of how the min-volume preference prevents jagged surfaces that are present in the max-volume solution.

In Figure 10(d) the authors compare their results with PMVS+Poisson surface reconstructions [25]. They also try to normalize the complexity of the mesh with QSlim [22] in order to be comparable with their simplified mesh. However, QSlim just worsened the results. These results show what remarkably clean models their algorithm produces.

7 Discussion

This section concludes this work with a discussion of the aforementioned topics.

In Section 3 the first stages of reconstructing building interiors were introduced. The Washington research group has created remarkable algorithms for efficient structure from motion suitable for large, unordered datasets. A well-performing multi-view stereo algorithm provides a dense scene reconstruction in the second stage. Thereafter, the scene is revised: Manhattan-world depth maps are extracted from the scene data and finally integrated again to provide a much simpler reconstruction of building interiors.

The last stages of estimating Manhattan-world stereo (cp. Section 4) and integrating depth maps (cp. Section 5) have much in common with multi-view stereo approaches. Both techniques aim at dense reconstructions of a scene. With focus on the taxonomy of MVS algorithms the reconstruction algorithm can be classified more precisely. A 2D MRF formulation over depth maps is part of the approach as well as a 3D volumetric MRF over voxels. The Manhattan-world assumption can be considered as a shape prior. With this in mind, arguably the whole pipeline of reconstructing building interiors contains some redundancy, because a custom MVS approach is built on top of another MVS approach (PMVS). In my opinion this yields both assets and drawbacks, which need to be discussed.

On the upside is the diversity of the used techniques. Possibly one technique may balance the drawbacks of another one. Enforcing the Manhattan-world assumption as a post-process enables to take advantage of the maturity of well-known algorithms. The modular structure allows to exchange each part individually when research progresses. Examples are the efforts to parallelize and scale the algorithms for structure from motion as well as for PMVS [41, 1, 17]. Furthermore, each individual stage is reasonable. Uncalibrated cameras and diverse input images argues for repeatable and distinguishable local image features. The sparse nature of these reconstructions argue for multi-view geometry for dense reconstructions.

On the downside a long and complicated pipeline yields a more time consuming algorithm. Not only is it possible that diversity balances assets and drawbacks, but also shortcomings of one stage can have a bad impact on successive stages. This kind of error propagation should be analyzed and avoided. Furthermore, redundancy is sometimes an indicator for bad system design. Another example of redundancy is the fact that different local image features are extracted both with Bundler (using SIFT features) as well as with PMVS (using Harris features). There, the choice of the extraction algorithm is inconsistent due to the modular structure of the algorithm.

[29] show that Manhattan-world assumptions can be applied directly after having obtained structure from motion and without the need of another MVS algorithm. However, their algorithm operates in an outdoor environment with sequential input data. In [38] an intermediate MVS algorithm is not used. Indoor scenes are successfully reconstructed using piecewise planar assumptions that are weaker than the Manhattan-world assumption. These algorithms have in common that they start with estimating main orientation parameters and then estimate candidate planes. MRFs are also often used as a core element of the reconstruction pipeline. These sometimes only differ in the way they formulate the MRF.

Due to their similar nature it would be very interesting to compare these approaches in more detail. In my opinion, ground truth datasets could be derived from existing real-world datasets, that are used to compare MVS algorithms, by simplifying the known scene geometry to bounding boxes. These are easy to create. Then, MVS algorithms, which rely on planarity assumptions, could be compared on how well their reconstruction compares to the ground-truth bounding box scenes when real-world data is provided as input.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Proc. of ICCV*, 2009.

[2] A. Bartoli. A random sampling strategy for piecewise planar scene segmentation. *Computer Vision and Image Understanding*, 105(1), 2007.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11), 2001.

[4] *BUNDLER*. Software available from: `http://phototour.cs.washington.edu/bundler/`.

[5] R. T. Collins. A space-sweep approach to true multi-image matching. In *Proc. of CVPR*, 1996.

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5), 2002.

[7] N. Cornelis, B. Leibe, K. Cornelis, and L. J. Van Gool. 3d urban scene modeling integrating recognition and reconstruction. *IJCV*, 78(2-3), 2008.

[8] J. M. Coughlan and A. L. Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In *Proc. of NIPS*, 2000.

[9] *Image of situation after Flip operation for Delaunay Triangulation.* Image source: `http://en.wikipedia.org/wiki/File:Delaunay_after_flip.png`, License: `http://creativecommons.org/licenses/by-sa/3.0/legalcode`.

[10] *Image of situation before Flip operation for Delaunay Triangulation.* Image source: `http://en.wikipedia.org/wiki/File:Delaunay_before_flip.png`, License: `http://creativecommons.org/licenses/by-sa/3.0/legalcode`.

[11] B. N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, (6), 1934.

[12] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Proc. of Photogrammetric Computer Vision (PCV)*, 2006.

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6), 1981.

[14] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[15] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhattan-world stereo. In *Proc. of CVPR*, 2009.

[16] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *Proc. of ICCV*, 2009.

[17] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Proc. of CVPR*, 2010.

[18] Y. Furukawa and J. Ponce. High-fidelity image-based modeling. Technical report, UIUC, 2006-02.

[19] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *PAMI*, 2009.

[20] D. Gallup, J. M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Proc. of CVPR*, 2007.

[21] P. Gargallo and P. Sturm. Bayesian 3d modeling from images using multiple depth maps. In *Proc. of CVPR*, volume 2, 2005.

[22] M. Garland. *Qslim: Quadric-based simplification algorithm.* `http://mgarland.org/software/qslim.html`.

[23] R. I. Hartley. In defense of the eight-point algorithm. *PAMI*, 19(6), 1997.

[24] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, second edition, 2004.

[25] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[26] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. of ECCV*, 2002.

[27] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *Proc. of ECCV*, volume 3, 2002.

[28] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[29] B. Micusík and J. Kosecka. Piecewise planar city 3d modeling from street view panoramic sequences. In *Proc. of CVPR*, 2009.

[30] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1), 2004.

[31] P J. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proc. of ICCV*, 1998.

[32] D. Nistér. An efficient solution to the five-point relative pose problem. *PAMI*, 26(6), 2004.

[33] *PMVS*. Software available from: `http://grail.cs.washington.edu/software/pmvs/`.

[34] S. Roy and I. J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Proc. of ICCV*, 1998.

[35] G. Schindler and F. Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *Proc. of ICCV*, 2004.

[36] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. of CVPR*, volume 1, 2006.

[37] S. N. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *Proc. of ICCV*, volume 1, 2005.

[38] S. N. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *Proc. of ICCV*, 2009.

[39] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. *ACM Trans. Graph.*, 27(3), 2008.

[40] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, 2006.

[41] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal sets for efficient structure from motion. In *Proc. of CVPR*, 2008.

[42] R. Szeliski. A multi-view approach to motion and stereo. In *Proc. of CVPR*, 1999.

[43] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proc. of CVPR*, volume 2, 2005.

[44] L. Zebedin, J. Bauer, K. Karner, and H. Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *Proc. of ECCV*, 2008.

[45] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. A. J. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3), 2004.