Reconstructing Indoor Scenes with Omni-Cameras

Frank Bauer Martin Meister Marc Stamminger

University of Erlangen-Nuremberg
Am Wolfsmantel 33
91058 Erlangen, Germany

{frank.bauer, martin.meister, marc.stamminger}@informatik.uni-erlangen.de

ABSTRACT

We present a system similar to Debevec's *Facade* [DTM96] that improves the reconstruction of indoor scenes from photographs. With confined spaces it is often impractical to use regular photos as the base of the reconstruction. Combining pinhole cameras with fisheye shoots or photographs of any kind of reflective, parametrisable body such as light probes eases this problem. We call the later camera setup an omni-camera, because it enables us to acquire as much information as possible from a given viewpoint. Omni-cameras make it possible to reconstruct the geometry of an entire room from just one view. Removing the pinhole camera constraint invalidates some key assumptions made in *Facade*. This paper shows how to work around the problems arising from this approach by adding scene specific knowledge as well as a genetic component to the solver. When using omni-cameras we can no longer take advantage of a simple texture projection to obtain the materials for the scene. Instead we propose a method for texture generation that is transparent to the camera setup used.

Keywords
Image-Based, Modeling, Reconstruction, Architecture, Omni-Camera

1. INTRODUCTION

The reconstruction of shape from photographs is one of the fundamental problems of computer vision and computer graphics. It is used either to model important present-day landmark architectural scenes and famous buildings as well as in cultural heritage projects with scientific background, e.g. aiming at digitally preserving a present state of conservation. In recent development, textures and models produced by systems as the one detailed in this paper can be used as the base to retrieving grammars for procedural modelling approaches [MZWG07].

Image-based scene reconstruction under general circumstances with no a priori knowledge of the position of cameras or any constraints on the geometry of the real scene is an ill-posed problem. Several approaches have been proposed, dealing with a subset of unknown and known factors. For very densely sampled scenes traditional light field renderers can give a very good approximation of the model and the reflectance. Global proxy geometries are extractable by shape-from-silhouette methods from the visual hull [GGSC96]. In most cases however, these methods are only applicable to turntable setups of objects rather small in size.

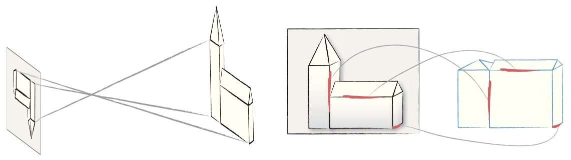
Figure 1: ***left*** **Pinhole camera model;** ***right*** **Setting up correspondences**

Another approach to the same problem is illustrated in [KS00]. The positions of the cameras are known and the underlying unknown scene is reconstructed by a volumetric approach that discards all voxels that are not mapped photo-consistently in all images. The algorithm works well for lambertian scenes with an extension to more complex colour models possible. Global effects such as shadowing, transparency and inter-reflections must be ignored and cannot be modelled.

Modelling from a sparse set of photographs requires additional constraints on the reconstruction algorithm. One feasible way was presented in 1996 [DTM96] where a user has to define a crude box-based geometry (the **base model**) and manually find correspondences between features in the images and features in that base model. The reconstruction features used are edges in the source camera images. When dealing with situations where some parts of the model cannot be seen, symmetries of the model are exploited to retrieve information about the hidden surfaces or blocks. Symmetries are also useful to reduce the dimension of the reconstruction problem.

2. FACADE

Facade was restricted to a pinhole-camera model (see Figure 1, left). Constraints imposed by this choice were closely interwoven into the algorithms presented. For example, setting up the feature correspondences (see Figure 1, right) usually works like this:

1. Mark two points on an edge in a source image (**projected edge**)
2. Construct a ray through the cameras focal point and each of the previously marked points (point rays)
3. The focal point and the two point rays construct a plane (**reconstruction plane** E)
4. The user chooses an edge (**source edge**) in the base model and links it to the projected edge

The rays spanning the reconstruction plane are called **reconstruction rays** $\tilde{r}$ in this paper. We need to find the translation T_K and rotation R_K for each camera in order to determine the position of the reconstruction plane in world-coordinates. Using the fact that this plane should contain the source edge we can derive some simple formulas to find the camera rotation matrix R_K (see Figure 2).

Most edges in an architectural scene are axis aligned, so we know that the reconstruction plane of those edges should be parallel to a given axis d_{B_K}. In other words the plane normal vector $\vec{n_E}$ has to be perpendicular to that axis. This gives the equation

$$\vec{n_E} * R_K * d_{B_K} = 0$$

Having multiple edges that are parallel to different axes it is possible to build an objective function we can use to obtain an initial estimate for the camera rotation (for details see Section 5.1).

When the camera rotation is known the translation T_K is simultaneously reconstructed with all other parameters of the scene (size, location and rotation of the blocks in the base model) in [DTM96]. When no rotated block is present, the resulting functions are linear and a result can be computed.

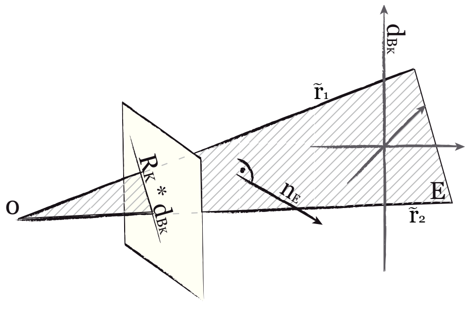

Figure 2: Camera rotation estimation

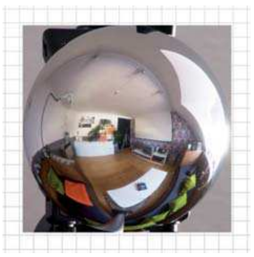

Figure 3: Photo of a Light Probe

With rotations the task is not as easy. In *Facade* the user needs to give an initial estimate for them. Since the resulting functions are no longer linear, an objective function is generated once again and solved with a Newton-Raphson algorithm (for details see Section 5.2).

At this stage we know the location and rotation of each camera in world-space as well as every parameter that defines our reconstructed model. The cameras in world-space are used to project their corresponding source images onto the blocks in the scene, allowing the software to impose view dependent texturing onto the scene in a very simple fashion, all possible because of the pinhole camera constraint.

3. OUR CONTRIBUTION

When shooting scenes inside we have to deal with confined spaces, where it is not feasible to take an overview picture capturing more than a part of a wall without using wide-angle lenses. To resolve this restriction we tried to use omni-camera setups like the photograph of a light probe (Figure 3).

We introduced new problems when building the reconstruction planes by removing the pinhole constraint *Facade* relies on (see Section 2). Non-skew rays reflected on the sphere for example are skew in general. Approximating a plane with those rays has the effect, that the resulting reconstruction plane does not contain the source edge. This renders the Newton-Raphson optimizers used by *Facade* less stable. We propose some additional enhancements to circumvent the loss in robustness in Section 5.3.

Marking a projected edge in the camera images is also no longer straight forward, as the source edge projects onto a curve in some setups.

Since the projective texturing relied on the pinhole camera model we can no longer use it for our omni-camera setups. Instead we propose a simple ray casting approach, as detailed in Section 6. This enables a texturing process independent of the camera setup used. We employ the textures to export a complete scene for use in other modelling or rendering applications or as a block replacement (Section 7) in a more complex scene.

When reconstructing an indoor scene we need rotated Blocks more frequently than with regular architec-

tural outdoor settings. That circumstance forces us to use non-linear optimization for almost every scene we reconstruct, slowing down the process. We propose some enhancements to the classic *Facade* approach (Section 5.1) resulting in lower reconstruction time and improved robustness that compensates for the error introduced by approximating the reconstruction plane.
The advantage of an omni-camera is obvious. With a light probe setup we can gather almost a 360-degree view of our room with only one shoot, often allowing us to reconstruct the geometry of the scene from one photograph.

In order to give the scenes some more depth and allow easy incremental modelling, we also introduce the use of block replacements to our system as discussed in Section 7.

4. SCENE PREPARATION

Preparing a scene for reconstruction is predominantly independent of the camera system used. We will highlight everything that is different for sundries setups or whenever the classic *Facade* setting is not applicable to our omni-cameras.

4.1. Photographs

With a standard camera setup we have to consider some constraints that arise from the pinhole assumption. We need to set long focal lengths and a big aperture value to gather results that match a photo taken by an ideal pinhole camera as much as possible.
When shooting a light probe setup we direct the camera towards the mirror ball in such a way that the centre of the mirror ball lies on the optical axis of that camera. The diameter of the light probe should be as small as feasible compared to the focal length. We generally work with focal lengths of 450mm and sphere diameters of 80 to 150mm. With this setup the camera and its supporting tripod obscure as little space on the image as possible.
In a pre-process we mask unwanted geometry in the obtained photographs. Omitting this step would result in the camera and the tripod to get projected on the textures in the final step.

4.2. Base Model

Every object in the scene has to be represented by a crude approximating block (like a cube or a ramp) defined by a type and a set of numeric parameters (like width, height...).
By using constraints on the blocks (like symmetries) we can reduce the number of parameters that have to be determined during the reconstruction.
We would like to point out, that in contrast to *Facade* our system does not rely on a crude approximation for

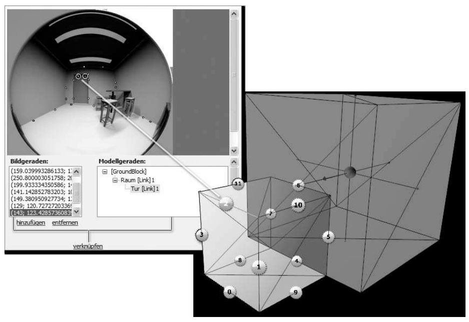

Figure 4: Link a curve in the source image to an edge in the model. The crude model shows the cube for the room itself and a door.

the block parameters given by the user. Our optimization to the reconstruction process makes it more robust than the original approach.

4.3. Adding Cameras

We have to create a camera for every taken image. Our software allows us to mix cameras of different types. We found that it simplifies the reconstruction process if we use the omni-setups to reconstruct the geometry of the room and its objects as well as a first and very crude texture. Regular photographs are then used to refine the visual quality of the result by adding additional images in a later iteration.
For each camera our system needs to know the following intrinsic parameters: camera type (regular pinhole, light probe setup...), film size and focal length. In case of a light probe setup we additionally need the diameter of the mirror ball and the distance of the focal point to the balls centre.
If the images were taken digitally we can use the EXIF information stored to automatically determine the film size (by camera model) and the focal length. If the user specifies the radius of the sphere we can also automatically compute the distance between sphere and camera (assuming the sphere completely fills the photograph).

5. SCENE RECONSTRUCTION

We need to set up correspondences before we can reconstruct the camera transformation matrices R_K and T_K or any other parameter. Section 2 already explained how this is done for the pinhole camera. It also detailed that we need to construct a reconstruction plane for the *Facade*-algorithms to work properly.
When using a light probe setup, the user has to perform the same basic steps. In this case however a line in world space projects to a curve in image space. Marking two points on that curve is still enough to identify a straight edge in world-space (see Figure 4).
For every point marked in the image we construct point rays $\vec{d}_c = \overline{B_K F}$ (see Figure 5). When using light probes the point rays are not equal to the reconstruction rays $\tilde{r}$, as they do not intersect with the source

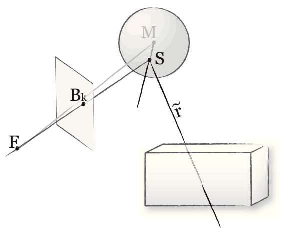

Figure 5: Constructing the reflected ray from a selected point B_K in the image plane.

edge in global space. To obtain the reconstruction rays we have to reflect them on the surface of the mirror ball using:

$$\begin{aligned}\vec{n_s} &= \frac{M-S}{\|M-S\|}\\ \tilde{r} &= \vec{d_c} - (2*(\vec{d_c}\circ\vec{n_s})*\vec{n_s})\end{aligned}$$

As the rays $\tilde{r}$ are not guaranteed to be non-skew we cannot create the plane E from them as easy as it is done at this point in the process for the pinhole camera. With the pinhole model we were able to use the focal point (where the reconstruction rays intersect) and the direction of the two reconstruction rays to construct the reconstruction plane E.

We still use the directions of our two reconstruction rays, but since they are skew, they do not intersect in one point. We decided to use the average of the two starting points of our reconstruction rays as an approximation of an intersection point.

In contrast to the reconstruction planes obtained by the pinhole model the orientation of this plane varies depending on the points we select on the projection of the edge in the source image. This obviously will be a problem for a robust reconstruction. By adapting the optimization strategies introduced in *Facade* we can still obtain very good results, as we will describe in detail in 5.1 and 5.2.

5.1. Camera Rotation

We proceed similar to the way suggested by Debevec in [DTM96] by finding an appropriate objective function $O = \sum(\mathrm{Err}_i)^2$ using the pseudo reconstruction plane E instead of the ones described in the original work. We use the square of this sum to better fit the Newton-Raphson method that is used throughout the paper.

An error or disparity function Err_i can be set up to calculate the rotation of each camera separately. The camera rotation R_K is processed in an upstream task, to reduce the number of parameters that have to be estimated, and to separate the error prone optimization of the Euler rotations from the rest of the reconstruction.

For each correspondence we have one pseudo reconstruction plane E. Together with the Euler camera rotation matrix $R_K{}^{-1}$ we can use that plane to formulate the disparity function Err_i. We choose $R_K{}^{-1}$ such that the normal $\vec{n_E}$ of the reconstruction plane is perpendicular to the desired direction $\vec{d_{B_K}}$ of the source edge in the model. This corresponds to a rotation of the camera around its pivot using $R_K{}^{-1}$ (see Figure 2).

$$\mathrm{Err}_i = (\vec{n_E}\circ(R_K * \vec{d_{B_K}}))^2 \qquad (1)$$

As we explained in the previous chapter, the pseudo reconstruction plane E is only an approximation of a plane that really contains the source edge. This renders the Newton-Rhapson optimization less robust due to the additional error. In order to compensate, we propose the use of a genetic algorithm to automatically find crude initial values for the rotation matrix. The one we used is a genetic algorithm based on an elite selection strategy without crossovers [Mit98].

Every generation contains 1000 individuals. Each is composed of the three Euler angles that determine the rotation of one camera. The initial population samples the cameras rotational space around its coordinate axes equidistantly (10 degrees). Each individual is assigned a quality, which corresponds to the square of the evaluation of the error function Err_i using the angles specified by that individual.

With each iteration a new generation is created containing the best 60% of parent individuals, and 40% newly created ones. The new ones are built based on the values of a chosen parent (an individual with high quality is more likely to be chosen). Those values are changed using a Gaussian distributed mutation. The distribution is adapted by decreasing the variance of the Gaussian in approximately every 20th generation to achieve a very dense sampling around the individuals of later generations. The iteration is stopped if the best individual of a generation meets a predefined criterion, or the 500th generation was spawned.

We find that the final result (the camera rotation) of this genetic procedure is only marginally improved by the following Newton-Raphson optimization.

5.2. Translation and Model Parameters

To obtain the global model parameters and the camera translations, we have to build another objective function, that represents the distance of each edge from the model to their corresponding reconstruction plane E (see Figure 6) in world-space.

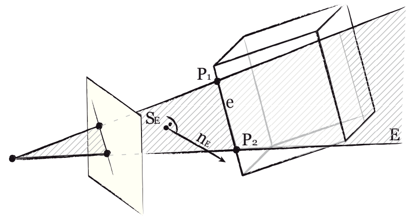

Figure 6: Camera Translation and block parameters

For two points P on a source edge e we can calculate that error by

$$d(e) = \vec{n_E} \circ (((R * (P - M)) - S_E) \quad (2)$$

where S_E is an arbitrary point on the reconstruction plane. This is a slight difference compared to the implementation in [DTM96]. We also have to take into account, that our reconstruction plane E is not as stable as the one used in a pure pinhole setup as we explained in Section 5.1.

Using Equation 2, we can calculate the objective function for all images I and all line correspondences e_I in that image by:

$$O = \sum_I \sum_{e_I} d(e_I)^2 \quad (3)$$

Minimizing this function yields the reconstruction of the scene.

5.3. Improving Reconstruction Results

If all rotation matrices R_B for all blocks in the scene are constant, we can compute the solution by solving a linear equation system, which is simple and often applicable when working with outdoor architecture. However, we found that indoor scenes tend to have a higher quantity of rotated blocks.

Since the evaluation of a Newton-Raphson algorithm can become slow, and might get caught in local minima, we decided to split this process into two separate tasks.

First we automatically select all base geometry blocks that are rotated around a known angle (this, of course, includes blocks not rotated) along with all camera translations. We can solve the resulting linear equation system comprised of the square of the distances $d(e)^2$ define in equation 2 for all correspondences relating to blocks not rotated.

As a result the camera translation and the fixedly rotated blocks are now consistently set up. Only the parameters of blocks with an unknown rotation remain unset. We build the same objective function as described in equation 3 for all unset edges. Since we do not include edges already computed in the linear step, the dimension and complexity of the objective function O is reduced.

At first we used a standard Newton-Raphson implementation that had to re-evaluate the hessian symbolically in each step. This proved to be a very slow solution, as the terms that had to be optimized were rather complex, and we had to tackle with big hessian matrices comprised of the second derivate of those terms.

We changed that implementation to a quasi Newton-Raphson algorithm, as described in [BNS94]. This method has the advantage that we never have to compute the hessian matrix, but can calculate an estimate for it through the gradients of the objective function.

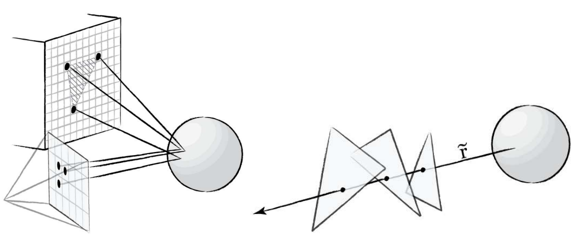

Figure 7: *left* Projecting source pixels into the reconstructed scene; *right* Only the first surface hit by a ray gets textured.

As a side note we would like to point to the fact that the optimization over the SO(3) group [Kue03] that is done for all camera and block rotation matrices is often not enough. Special care has to be taken since ambiguities can occur (for example positive or negative view directions) that can either be resolved by user-interaction, reparameterization of the rotational domain or specially adapted tests [ML03].

Our software automatically fixes the camera view direction by checking if the intersection of any given reconstruction ray with the model object is located in the expected octant of the cameras coordinate system.

In case of a light probe, the expected octant is determined by the location of the point that corresponds to the reconstruction ray on the light probe. For a regular pinhole setup we simply check if the intersection is located in front of or behind the camera.

In addition we added an extra phase into the Newton-Raphson optimization that is evaluating the objective function for different angles after each iteration.

After the result for one iteration is calculated, we change the variables that represent angles in 45-degree steps and calculate the error value achieved with the altered angles. If the result is smaller than the one obtained through the optimization step, we will use the new angles in the following Newton-iteration.

This alleviates the user from the need to set an approximate rotation for any block before starting the reconstruction, as it was necessary with *Facade*. All in all our changes made the reconstruction of rotated blocks and the use of pseudo reconstruction planes more reliable.

6. TEXTURE GENERATION

With our omni-cameras we can no longer use the simple projective texturing approach, as it is used in *Facade*. We propose to use a ray-casting algorithm to render the textures for a scene. This creates a layer of abstraction between the camera model used and the texturing process.

We build a reconstruction ray for each pixel in each source image, constructing it the exact same way as it is done for the points marked by the user during the scene reconstruction. Reutilizing the code generating

the reconstruction rays makes this process transparent to the underlying camera type.
By intersecting those rays with our scene geometry we obtain a list of points in world space that are visible through a given pixel. Only the closest intersection in front of the camera is coloured with the colour of the source pixel (see Figure 7, right). We choose to use forward ray tracing, because it is not always possible (depending on the camera type used) to find a unique ray from world space into the source images. Think about the contour of a sphere. At this singularity, one point in world-space is mapped to all points on that contour.

To write colour values to the textures we apply a linear interpolation of three projected neighbouring camera rays to fill larger texture regions by rasterizing the resulting triangle in object texture space (Figure 7 left). The alpha mask provided for each image in the pre process can be used to blend out regions that are defective. Furthermore, we calculate the scalar product of the ray with the surface normal and weight the incoming texels accordingly (similar to the blend field methods in [BBM$^+$01]). This ensures that rays with grazing angles contribute little to the resulting textures.
Another way for the user to interact with texture generation is to select a global weight for each texture. Especially in the presence of regular pinhole source images it may be advisable to discard any texture information from the reflective sources where more detailed source pictures are available. The pinhole source images have a far better local resolution and produce an increased local texture quality. In Figure 11 such detail images can be seen for the white radio on top of the shelf and the cupboard on the floor.
Of course there are problems with regions that are not seen from any of the source images. As they are never hit by any ray, we can fill the missing texture regions with a blank colour (see the greyish colour in Figure 8 on the floor projecting away from the chairs), fill it by interpolation techniques from neighbouring texels or by utilizing a texture synthesis approach [WL00].

Using this approach to generate textures allows our system to export the result to a file (for example to VRML), making the reconstructed scene independent from a specialized viewer.

7. MODEL EXCHANGE

To alleviate the hassle for the user to work out seemingly unnecessary details, we provide for an easy model block exchange.
Instead of reproducing an indoor scene with every detail, we use a bounding box as base geometry for an object we want to replace with a more detailed version. After scene reconstruction is finished any table model from a 3D model library can be fitted into the scene by applying the affine transformation that is available from the reconstruction process for each block. In Figure 8 the bounding boxes are shown over the replaced geometry.

Since we also allow the export of our scenes into arbitrary formats (like VRML or XSI), the model exchange can be used to build a complex scene iteratively. With this system it is convenient to first model some details of the scene using standard photographs for higher resolution. When finished the results are exported to a file and (if necessary) refined in an external editor.
In the next iteration we could start to model the room itself, representing the previous reconstructed detailed model with a simple block. When the reconstruction of the entire room is finished, we simply fit the model stored in the file into the bounding box of the block we created as a placeholder.

8. RESULTS

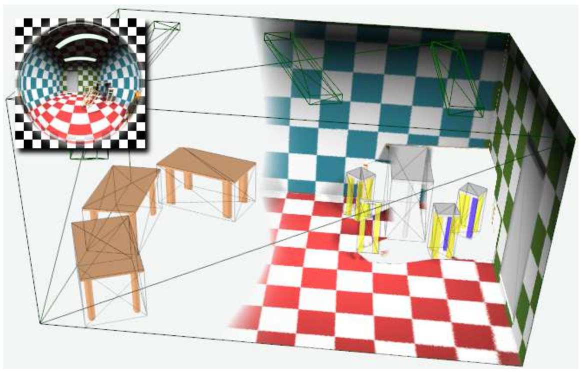

Figure 8: Reconstruction of a synthetic example from one light probe image.

We tested our implementation with four different scenes. The first one was a synthetic scene (Figure 8), to show the general usability of the algorithm. The scene was generated with Blender, using a near optimal light probe. Only half of the reconstructed room is textured, as the image resolution of the light probe is not high enough for the room behind the probe.
The second scene contained two offices (Figures 9 and 10). The third one was a home office (Figure 11) and the fourth a living room (Figure 12).
While a calibrated camera was used in [DTM96] to acquire the photos for the reconstruction, all our results were obtained with an uncalibrated one.

We compare the dimensions of the found parameters to the ones in the real scene, which gives us a measure for the quality of the reconstructed geometry. Since the reconstructed width of the scene is always set to 1.0, we have to multiply all our values with the actual width. However, the camera position and rotation in

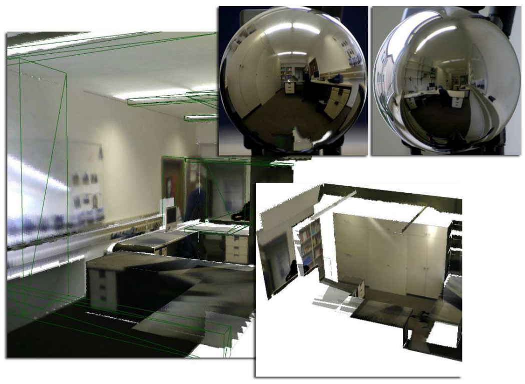

Figure 9: Reconstruction of a small office from two light probes with 800x800 pixels each. This example shows the projection of an unmodelled chair onto the desktop.

the real scene was not recorded, so we had to compare those results visually.
The most challenging scene was the home office scene (Figure 11), because the space in that room is quite confined. The floor has a footprint of 12.5 m^2. Just considering the area of the room, that is more than $1m$ high, we get a size of about 9 m^2. It was reconstructed using a simple cube and an additional ramp for the room model. In a room this small it would not be practical to use regular photographs for the reconstruction of the geometry.

Parameter	Measurement	Reconst.	Result
Width	2.70m	1.000	2.70m
Height	2.30m	0.857	2.31m
Length	4.65m	1.718	4.64m

Table 1: Comparing reconstructed parameters to real world measurements in the home office scene.

The results in Table 1 show, that the reconstructed dimensions are in good correspondence with the measured values. The size of the reconstructed scene is only 1 cm off the real values.

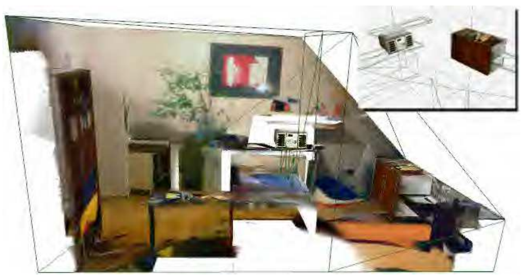

Figure 11: Confined space (9 m^2) home office scene: 3D view of extracted textures from light probe and detail perspective images.

We would like to point out, that we used the foot of a lamp to reconstruct, which was a slightly flattened sphere. This demonstrates that our method can generate robust results for the geometry with suboptimal light probes. We gain this robustness through our extensions to the calculation of the reconstruction plane (see Section 5) where we calculate the average of the starting points of the skew reconstruction rays.
Using the deformed light probe the texture correspondence was not always given. This became most obvious, if the surface projects to the outer regions of the mirror ball (see Figure 11).
The reconstruction of the camera position was very precise. We put the hemisphere we used as the light probe on the door and the walls of the room. This position was reconstructed correctly.

In order to determine the influence of a deformed sphere on the reconstruction results, we used a non deformed light probe for the living room scene in Figure 12. The results for the geometry was only marginally better then the one in the home office scene, but the textures were in better correspondence (except in the outer regions of the mirror ball).

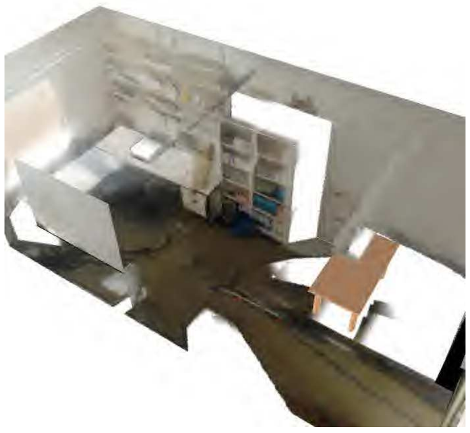

Figure 10: Another small office scene using a quite bumpy light probe.

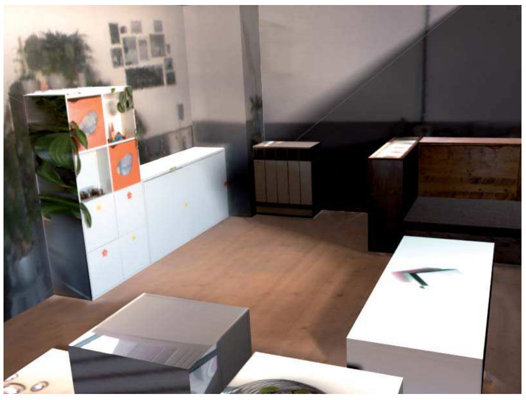

Figure 12: A living room reconstructed using 2 light probe images and several pinhole shoots

9. CONCLUSION

We showed how to reconstruct an indoor scene using only one or very few images by applying a well-known method to various omni-cameras. Texture generation can be largely automated and yields atlas maps for single objects or the whole scene exportable to any 3D graphics format. Crude base blocks can be substituted with complex 3D geometry reducing the amount of detail work for the user while enhancing quality.

Allowing the usage of non-pinhole camera setups revealed a series of difficulties with the existing approach we had to tackle. Most of them due to the additional error introduced by the pseudo reconstruction planes we have to use in omni-setups.

The use of a genetic algorithm to get a good approximation for the initial values of the camera rotation used in the quasi Newton-Raphson minimization proved to be very precise, and made the estimation more robust compared to the original approach using just a Newton-Raphson solver.

By splitting the following estimate of the camera translation and the model parameters in a step where all non-rotated blocks are computed and a step that minimizes the parameters for all rotated blocks, we gained a big advantage in terms of speed and precision. It also contributed to a more reliable reconstruction.

The most surprising result was, that our method could robustly reconstruct a scene using non-optimal spheres as light probes.

In the future, we would like to extend the presented system by automatic edge detection in the source images that would speed up the scene preparations. Reconstructing the illumination of the scene as detailed in [SHR+99] from the generated textured objects is another possible enhancement that would be very useful when importing external geometry, as it allows us to match the lighting of the loaded textures to the scene.

REFERENCES

[BBM+01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured lumigraph rendering. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 425–432. ACM Press / ACM SIGGRAPH, 2001.

[BNS94] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 11–20, New York, NY, USA, 1996. ACM Press.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, pages 43–54, New York, NY, USA, 1996. ACM Press.

[KS00] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[Kue03] Frank O. Kuehnel. On the minimization over so(3) manifolds. Technical report, RIACS NASA Ames Research Center, 2003.

[Mit98] Melanie Mitchell. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. The MIT Press, 1998.

[ML03] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. Technical report, Society for Industrial and Applied Mathematics, 2003.

[MZWG07] Pascal Mueller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. 26(3), 2007.

[SHR+99] H. Schirmacher, W. Heidrich, M. Rubick, D. Schiron, and H. Seidel. Image-based brdf reconstruction, 1999.

[WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. pages 479–488, 2000.