

# 3D Reconstruction of Buildings with Automatic Facade Refinement

Larsen, C.; Moeslund, T.B.

*Published in:*
Lecture Notes in Computer Science

*DOI (link to publication from Publisher):*
10.1007/978-3-642-24028-7_42

*Publication date:*
2011

*Document Version*
Publisher's PDF, also known as Version of record

# 3D Reconstruction of Buildings with Automatic Facade Refinement

C. Larsen and T.B. Moeslund

Department for Architecture, Design and Media Technology
Aalborg University, Denmark

**Abstract.** 3D reconstruction and texturing of buildings have a large number of applications and have therefore been the focus of much attention in recent years. One aspect that is still lacking, however, is a way to reconstruct recessed features such as windows and doors. These may have little value when seen from a frontal viewpoint. But when the reconstructed model is rotated and zoomed the lack of details will leap out. In this work we therefore aim at reconstructing a 3D model with refined details. To this end we apply a structure from motion approach based on bottom up bundle adjustment to first estimate a 3D point cloud of a building. Next, a rectified texture of the facade is extracted and analyzed in order to detect recessed features and their depths, and enhance the 3D model accordingly. For evaluation we apply the method to a number of different buildings.

## 1 Introduction

3D reconstruction and texturing of buildings have a large number of applications ranging from urban planning and marketing, to tourism where services such as Google Earth and Microsoft's Virtual Earth have promoted the great potential to the public. Much research has been focused on different aspects of the basic problems of reconstruction and texturing [1,2,3] and the frontiers have in the later years been pushed forward. As the technology matures so does the desire for better and more precise models. One aspect that is currently lacking is an explicit modeling of recessed features such as windows and doors [4]. The consequence of not modeling such facade details is minor when viewing the reconstructed model from a frontal viewpoint or a distance. But when the reconstructed model is rotated and zoomed the lack of details will leap out. In this work we therefore aim at reconstructing a 3D model with refined details.

Recently methods have been reported that aim to recover detailed facade features. In [5] high resolution 3D data of a building facade is obtained using a terrestrial laser scanner. RANSAC is applied to segment the 3D point cloud into a number of planes corresponding to different facade features. In [6] a similar approach (LiDAR) is applied to obtain high resolution 3D data. Facade features (windows) are segmented based on the notion that glass does not reflect the LiDAR pulses, hence no data. From the detected windows a grammar is made that can generate synthetic facade features for occluding building parts or on new buildings (assuming a similar architecture). In [7] LiDAR data is also applied together with a grammar. A reversible jump MCMC approach is used to select hypotheses which are then compared with the input data. In [8] overlapping images are analyzed to find facade features in buildings with many repeated patterns. The

G. Bebis et al. (Eds.): ISVC 2011, Part I, LNCS 6938, pp. 451–460, 2011.

patterns are detected by looking for similar intensity profiles between interest points found by a Harris corner detector. In [9] facade features are estimated based on just one image. The inherent lack of 3D data is partly overcome by relying on a strong grammar and other architectural insides. In [4] a number of images of the same building are used to extract structure from motion. A sketch-based interface then allows a user to construct first the overall facades and next more and more detailed features. Detailed reconstruction, but the procedure takes minutes of user interaction for small buildings and tens of minutes for bigger buildings.

We seek a solution that does not involve complicated and expensive hardware, hence we aim at a solution based on imagery from a standard camera. The solution should not require learning a grammar or being limited to certain building types (like big buildings with a high number of repetitive facade features). In figure 1 our approach is illustrated. It starts out by estimating the intrinsic camera parameters without the need for a calibration. Next, a structure from motion approach is applied to estimate a 3D point-cloud from which a coarse model of a facade together with a rectified texture is extracted. Finally the detailed facade features are detected together with their depths and the final 3D detailed model is built. The paper follows the structure of the blocks in figure 1 together with a result section and a conclusive section.

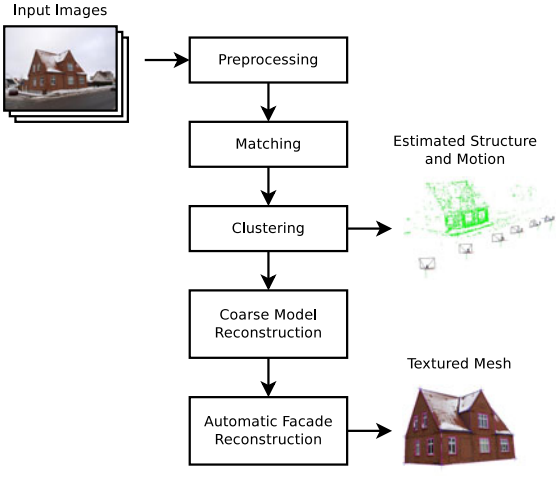


**Fig. 1.** Overview of the proposed system

## 2 Pre-processing

Before any 3D reconstruction can commence we need to find the intrinsic camera parameters: the focal length $\alpha$ in pixels and the principal point $(x_0, y_0)$ in pixels.

A standard camera calibration can be applied to obtain these parameters. But if images are captured with different zoom factors multiple calibrations are required. Moreover, if the user is a non-expert calibration should be avoided altogether. To this end we suggest to approximate the intrinsic camera parameters based on the meta data, which

is stored together with a picture in most modern cameras. The meta data is stored in the Exchangeable Image File Format (EXIF) and includes, among other things, the camera type, resolution, and focal length (in mm).

A good approximation of the principal point is given by the resolution, i.e., the center of the picture. The focal length in pixels is given by $\alpha = f \cdot \frac{d_{\text{image}}}{d_{\text{sensor}}}$, where $f$ is the focal length in mm read from the EXIF data, $d_{\text{image}}$ is the diagonal of the image in pixels (directly given by the resolution), and $d_{\text{sensor}}$ is the diagonal of the sensor in mm. $d_{\text{sensor}}$ is unfortunately not present in the EXIF data, but the camera type is, and since $d_{\text{sensor}}$ is static for a camera a look-up table can be constructed and applied.

In this work we have used an Olympus E-520 camera. We have estimated the intrinsic camera parameters using the approximation scheme suggested above and compared them with those obtained using a real camera calibration. We found that all three parameters have errors less than $3\%$ and even more importantly that the results of the 3D reconstruction did not suffer from using the approximation.

## 3 Matching

The purpose of the matching step is to estimate the relative pose of the cameras in all pairs of input images if possible. For each pair of input images a robust set of corresponding keypoints is identified using SIFT and filtered using RANSAC. Then the fundamental matrix is estimated using the 8-point algorithm [10]. Combining this with the intrinsic camera parameters the essential matrix and finally the relative pose is recovered. For details see [11,12].

## 4 Clustering

The purpose of clustering is to recover structure and motion for all input images. In this context a cluster is the collection of cameras with estimated extrinsic parameters and a point cloud of estimated keypoint positions, see figure 6.

Traditionally methods for recovering structure and motion are based on finding a global initial guess and then performing a final optimization using bundle adjustment. For bundle adjustment to succeed, however, a good initial guess is required [13]. In this work a bottom up approach, where bundle adjustment is applied throughout clustering, has been developed in order to minimize the risk of a bad initial guess preventing recovery of structure and motion.

The overall approach is to initialize the cluster from the best matching image pair, and then add images to the cluster iteratively until all input images are included. When adding an image to the cluster, the relative pose recovered during matching is used for computing an initial estimate of the extrinsic camera parameters, and keypoint inliers are triangulated to obtain an estimate of keypoint positions. Each time an image has been added, the whole cluster is optimized by applying bundle adjustment. For details see [11,13,14].

## 5 Coarse Model

We now have a cloud of 3D points representing the building, see figure 6. Extracting structures directly from the point cloud is erroneous in general and especially for fine details such as recessed features. Inspired by [4] we therefore introduce a semi-automatic approach. Firstly the user is asked to click dominant features, denoted locators, in two images. From these a polygon is built and textured automatically.

A GUI is created wherein the user can flip through the different images. For each (overall) planar surface the user clicks its corners in two images and a polygon is built and visualized, see figure 2. If the user accepts the polygon its 3D corners are extracted using the relation between the two images found in the previous section, and a 3D surface is fitted using a least squares solution.

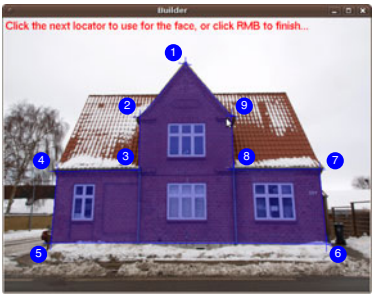


**Fig. 2.** Interactively defining a polygon by clicking at corresponding locators in two images. Here nine locators are used.

To create a rectified texture for a polygon, it is necessary to determine which of the input images that has the best view of the polygon. We select the input image corresponding to the camera with the smallest angle between the normal of the plane and the direction from the centroid to the camera, see figure 3.

As $\cos\theta$ goes towards 0 the quality of the obtained rectified texture decreases. Therefore for the selected image it is required that $\cos\theta > \tau$, where the threshold $\tau$ is set to $\cos 80°$ in this work. Furthermore it is required that the reprojections of all locators $\mathbf{L}_i$ used as vertices for the polygon lie within the boundary of the selected image, because otherwise a texture for the whole polygon can not be extracted.

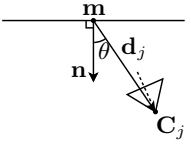


**Fig. 3.** The vectors and angles used for selecting the best image

### 5.1 Creating the Rectified Texture

For each 3D vertex of the polygon, two 2D projections are computed: a projection into the plane of the polygon, and a projection into the input image. The vertices of the polygon are defined by the locator positions $\mathbf{L}_i$, and the projection of each vertex into the plane is computed using two orthogonal axes $\mathbf{u}_x$ and $\mathbf{u}_y$ of unit length lying in the polygon plane. First the centroid, $\mathbf{m}$, is subtracted to obtain the local vertex coordinates $\mathbf{V}'_i = \mathbf{L}_i - \mathbf{m}$, which are then projected onto the plane to obtain the 3D coordinates

$$\mathbf{V}_i = \mathbf{V}'_i - (\mathbf{V}'_i \cdot \mathbf{n})\mathbf{n}. \tag{1}$$

Now to compute the 2D projections of the vertices, each $\mathbf{V}_i$ is projected onto the plane axes to obtain the vector

$$\hat{\mathbf{v}}_i = \begin{bmatrix} \mathbf{V}_i \cdot \mathbf{u}_x \\ \mathbf{V}_i \cdot \mathbf{u}_y \\ 1 \end{bmatrix}, \tag{2}$$

which is the homogeneous 3D-vector representing the 2D projection of locator $i$ into the plane of the polygon.

The projection of each vertex into the input image is simply achieved by projecting the locator position $\mathbf{L}_i$ using the camera calibration information available for the best image, $a$. That is the projection into image $a$ of the locator with position $\mathbf{L}_i$ is computed as the homogeneous 3D-vector

$$\mathbf{v}'_i = \mathrm{K}_a \mathbf{L}_i, \tag{3}$$

where the camera calibration matrix $\mathrm{K}_a$ is applied such that $\mathbf{v}'_i$ is expressed in pixel coordinates. The mapping between the polygon and the input image is now represented by the two corresponding sets of 2D projections: the set in the plane of the polygon defined by $\hat{\mathbf{v}}_i$, and the set in image $a$ defined by $\mathbf{v}'_i$.

The final step is to actually create the rectified texture using a homography computed from corresponding points $\mathbf{v}_i$ and $\mathbf{v}'_i$. From the previous step, the projections $\mathbf{v}'_i$ of the locators in image $a$ are available, and they are expressed in pixel coordinates. The corresponding pixel coordinates $\mathbf{v}_i$ in the rectified texture to be extracted are needed, and these can be derived from the projections $\hat{\mathbf{v}}_i$ in the plane obtained in the previous step. But as the length of the computed plane axes do not correspond to the size of a pixel in the rectified texture, we perform isotropic scaling and translation. Let $\mathrm{T}$ be a $3\times3$ matrix representing a transformation consisting of isotropic scaling and translation, then

$$\mathbf{v}_i = \mathrm{T}\hat{\mathbf{v}}_i, \tag{4}$$

where the scale factor is chosen such that the size of a pixel in the rectified texture becomes approximately the same size as in the input image, and the translation is chosen such that the rectified texture is cropped to the polygon but leaving a small border.

## 6 Facade Reconstruction

The final step is to refine the coarse model by adding local facade features, such as recessed windows, in order to increase the realism of the reconstructed model. To this end we first find the regions in question and second the depth of each region.

### 6.1 Facade Segmentation

A facade polygon typically consists of wall regions with large areas of similar color and texture, and smaller regions inside the wall regions, which represent facade features, that deviate in appearance from the wall. Thus for identifying regions in the facade we analyze the rectified texture. Often a wall appears primarily in one color with no significant texture or it is a brick wall consisting of bricks of similar color with joints between them. In the case of a plain colored wall, identifying the whole wall as one region is simple. For brick walls, however, it is more difficult due to the often high contrast between bricks and joints. We therefore blur the image such that the joints between bricks and other high frequency elements of the image are reduced.

The segmentation is based on color classification in the HSV space. First we apply the single-linkage algorithm, which is based on agglomerative hierarchical clustering. It starts with randomly selecting a subset of samples (HSV pixels). In this work 1.000 samples are used. Each sample starts as a single cluster and then iteratively merges with similar clusters until the cost (distance) of merging yet another cluster pair becomes to big. Specifically, let $D_i$ and $D_j$ be the subsets of observations in two candidate clusters for merging, then the distance between the clusters is given by

$$d_{\min}(D_i, D_j) = \min_{\substack{\mathbf{x} \in D_i \\ \mathbf{x}' \in D_j}} ||\mathbf{x} - \mathbf{x}'||. \tag{5}$$

We terminate the algorithm when $d_{\min}$ for the nearest cluster exceeds $0.025$. In figure 4 a representative subset of samples is shown. The colors illustrate the two non-singular clusters when the algorithm is terminated. For all available test sequences the biggest cluster represents the wall. We model this by a 3D Gaussian distribution and apply Mahalanobis distance to classify all HSV pixels in the rectified and blurred texture image, see figure 4. For illustrative purposes we also designed a classifier for the second biggest cluster (green pixels). The resulting binary image is filtered using morphology and connected-component analysis. Finally, each recessed feature is detected as a region of connected (black) pixels inside the filtered image, see figure 4.

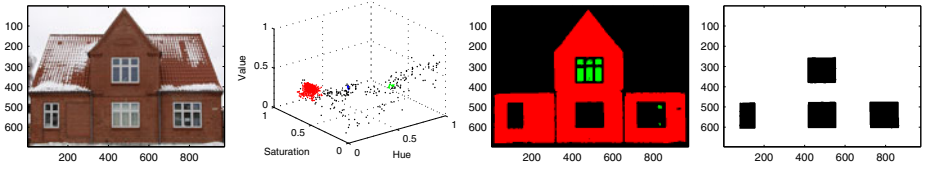


**Fig. 4.** 1: Rectified texture. 2: The single-linkage algorithm applied to the random sampled pixels. The two largest clusters are marked in red and green. 3: Classification of all pixels in the polygon texture. 4: The segmented regions are filtered.

### 6.2 Refining the Model

Each of the segmented regions is represented as a subset of pixels in the binary image. For refining the coarse model, however, it is necessary to transform the pixels into

polygons that represent the contour of the detected facade region. Both contour sampling and polygon approximation were investigated, but it turned out that finding the oriented rectangle with minimum area enclosing the pixels was a more stable solution[1].

For each rectangle a hole is carped out of the polygon and replaced by five new polygons. One representing the recessed feature and four representing the edges of the wall surrounding the recessed feature. The textures are found as explained in section 5. For textures not defined we copy the texture from the opposite edge.

The actual depth of a recessed feature is estimated using an analysis-by-synthesis approach. For each possible depth value we update the 3D model. We then synthesize the texture of the model for the recessed feature (including the edges) as seen from the viewpoint of a specific image and compare with the actual pixel values in that image using the correlation coefficient

$$r = \frac{\sum(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{y}_i - \overline{\mathbf{y}})}{\sqrt{\sum(\mathbf{x}_i - \overline{\mathbf{x}})^2}\sqrt{\sum(\mathbf{y}_i - \overline{\mathbf{y}})^2}} \tag{6}$$

where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the $i$th pixels in the image and synthesized texture, respectively, and $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ are the mean pixel values in the image and synthesized texture, respectively. This is done for different depth values and averaged over three different images (the most frontal and the most extreme viewpoints from each side). The maximum of the averaged curve defines the most likely depth value.

## 7 Results

Eleven different data sets are used to evaluate the system. In figure 5 sample images can be seen.

**Fig. 5.** Sample input images from the eleven data sets

[1] This is of course only true for rectangular regions.

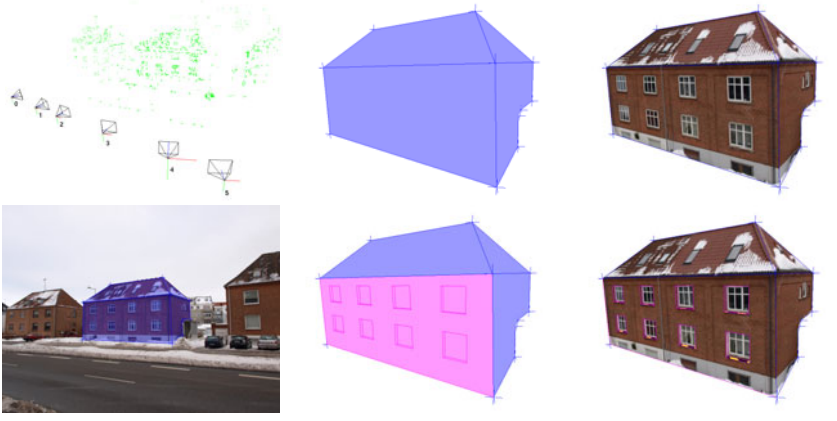

**Fig. 6.** The different steps in our system illustrated for data set *i*. Top left: Recovered structure from motion. Top center: The coarse model. Top right: The rectified texture of the course model. Bottom left: The coarse model superimposed on an input image. Bottom center: The refined model for one facade. Bottom right: The refined model including texture.

**Fig. 7.** Three different synthetic viewpoints of the building in data set *a* with recessed features inserted (top row) and without recessed features (bottom row)

In figure 6 the sub- and final results of our approach are shown for data set *g*. In the preprocessing step a massive amount of keypoints are found in each image due to especially the significant high frequency information coming from the bricks. To avoid such repetitive keypoints we enforce a conservative threshold when matching keypoints and hence only keep the most distinctive keypoints. The quality of the structure from motion (RMS reprojection error) estimated using our method is below 0.4 pixels, which is acceptable for reconstruction. In figure 6 a representative example is shown.

In six of the eleven data sets, the number of correctly detected facade features equals ground truth. For the remaining data sets, the percentage of correctly detected features is between $65\%$ and $86\%$, and for all data sets on average $89\%$ of the ground truth features were correctly detected. The main source of error is occlusion. That is, when a foreground object (tree or lamppost) is occluding parts of the building.

Since all pictures were taken from the ground some edges cannot be textured (illustrated by a yellow/black pattern). We handle this by copying the texture from one of the other recessed edges. This can be seen in figures 7 and 8 where close ups of different facade features are shown with and without refining the model. These figures also illustrate the benefit of modeling detailed features.

**Fig. 8.** Three different synthetic viewpoints of the building in data set *c* with recessed features inserted (top row) and without recessed features (bottom row)

## 8 Conclusion

We have presented a method for detecting facade features and including that in a refined 3D textured model of a building. The approach is based on structure from motion to estimate a 3D point cloud wherein a coarse facade model is defined. We segment the rectified texture of the coarse model using a clustering approach and hereby find the local facade features. Their depths are estimated using an analysis-by-synthesis approach. The approach is tested on eleven data sets and detects $89\%$ of the facade features correctly. Future work includes a texture selection mechanism for occlusion handling and an automated process for finding the coarse model, e.g., by finding intersections of planes fitted to the 3D data.

## References

1. Pollefeys, M., Van Gool, L.: From Images to 3D Models. Communications of the ACM 45, 50–55 (2002)
2. Frahm, J., Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y., Dunn, E., Clipp, B., Lazebnik, S., Pollefeys, M.: Building Rome on a Cloudless Day. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 368–381. Springer, Heidelberg (2010)
3. Benitez, S., Denis, E., Baillard, C.: Automatic Production of Occlusion-Free Rectified Facade Textures using Vehicle-Based Imagery. In: International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Commission XXXVIII, Saint-Mande, France (2010)
4. Sinha, S.N., Steedly, D., Szeliski, R., Agrawala, M., Pollefeys, M.: Interactive 3D Architectural Modeling from Unordered Photo Collections. In: SIGGRAPH, Asia (2008)
5. Boulaassal, H., Landes, T., Grussenmeyer, P., Tarsha-Kurdi, F.: Automatic Segmentation of Building Facades using Terrestial Laser Data. In: International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Commission V, Munich, Germany (2007)
6. Becker, S., Haala, N.: Grammar Supported Facade Reconstruction From Mobile LiDAR Mapping. In: International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Commission XXXVII, Paris, France (2009)
7. Rippererda, N., Brenner, C.: Application of a Formal Grammar to Facade Reconstruction in Semiautomatic and Automatic Environments. In: AGILE International Conference on Geographic Information Science, Hannover, Germany (2009)
8. Recky, M., Wendel, A., Leberl, F.: Facade Segmentation in a Multi-View Scenario. In: International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, Hangzhou, China (2011)
9. Gool, L., Zeng, G., Borre, F., Muller, P.: Towards Mass-Produced Building Models. In: International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Commission III, Munich, Germany (2007)
10. Hartley, R.I.: In Defence of the 8-point Algorithm. In: ICCV, Cambridge, USA (1995)
11. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2004) ISBN: 0521540518
12. Nister, D.: An Efficient Solution to the Five-Point Relative Pose Problem. PAMI 26(6), 756–770 (2004)
13. Schwartz, C., Klein, R.: Improving Initial Estimations for Structure from Motion Methods. In: The 13th Central European Seminar on Computer Graphics, CESCG 2009 (2009)
14. Lourakis, M.I.A., Argyros, A.A.: SBA: A Software Package for Generic Sparse Bundle Adjustment. ACM Transactions of Mathematical Software 36 (2009)