A SEMI-AUTOMATIC PROCEDURE FOR TEXTURING OF LASER SCANNING POINT CLOUDS WITH GOOGLE STREETVIEW IMAGES

J. F. Lichtenauer[a,*], B. Sirmacek[b]

[a] Laan der Vrijheid 92, 2661HM Bergschenhoek, The Netherlands - jeroenlichtenauer@gmail.com
[b] Department of Geoscience and Remote Sensing, Delft University of Technology,
Stevinweg 1, 2628CN Delft, The Netherlands - www.berilsirmacek.com

Commission VI, WG VI/4

KEY WORDS: Point Clouds, Terrestrial Laser Scanning, Structure from Motion (SfM), Texturing, Google Streetview

ABSTRACT:

We introduce a method to texture 3D urban models with photographs that even works for Google Streetview images and can be done with currently available free software. This allows realistic texturing, even when it is not possible or cost-effective to (re)visit a scanned site to take textured scans or photographs. Mapping a photograph onto a 3D model requires knowledge of the intrinsic and extrinsic camera parameters. The common way to obtain intrinsic parameters of a camera is by taking several photographs of a calibration object with a priori known structure. The extra challenge of using images from a database such as Google Streetview, rather than your own photographs, is that it does not allow for any controlled calibration. To overcome this limitation, we propose to calibrate the panoramic viewer of Google Streetview using Structure from Motion (SfM) on any structure of which Google Streetview offers views from multiple angles. After this, the extrinsic parameters for any other view can be calculated from 3 or more tie points between the image from Google Streetview and a 3D model of the scene. These point correspondences can either be obtained automatically or selected by manual annotation. We demonstrate how this procedure provides realistic 3D urban models in an easy and effective way, by using it to texture a publicly available point cloud from a terrestrial laser scan made in Bremen, Germany, with a screenshot from Google Streetview, after estimating the focal length from views from Paris, France.

1. INTRODUCTION

In recent years, a lot of progress has been made with the development of three dimensional (3D) sensors, 3D visualisation technologies and data storage-, processing- and distributions possibilities. This seems good news for one of its most useful applications: 3D mapping of our urban environments. However, the main limitation still left for large-scale realistic digitization of urban environments is the collection of data itself. Although flying- and Earth-orbiting cameras and 3D sensors allow coverage of the entire globe at relatively low cost, a view from above is severely restricted by occlusions. Unfortunately, the most relevant urban areas for daily use are mostly at ground level, rather than on rooftops or above tree canopies.

This means that terrestrial short-range measurements still need to be made to map urban environments realistically in 3D. While state-of-the-art mobile laser scanners can capture high quality 3D data together with coloured texture, their prices are high and the acquisition of texture data often requires significant extra scanning time over the capture of 3D data alone. It is mainly because of the cost of these sensors and the time it takes to use them to acquire measurements from all the required locations, that realistic 3D urban maps are still not widely available today.

Therefore, in order to make full use of the available 3D data technologies for realistic urban mapping, we also need to be able to include data from fast, low-cost scanning methods, as well as from previously recorded data, which do not always include colour texture information together with the 3D measurements. And even with 3D scans that include colour texture, unsatisfactory lighting conditions during a scan might still require new texture to be added afterwards.

*Corresponding author

To overcome this limitation, we propose a simple semi-automatic method to accurately texture 3D data with photographs, which consists of a novel combination of currently available free software implementations of pre-existing algorithms for Structure from Motion (SfM), camera pose estimation and texture mapping. Only three or more tie points per photo are needed to accurately align it to the corresponding 3D data. These tie points can either be obtained from a simple manual procedure that can be done by anyone without special training, or from an automatic 2D to 3D feature matching algorithm. To solve the problem of calibrating a panoramic viewer without being able to use an a priori known calibration pattern, we propose to obtain intrinsic camera parameters from a Structure from Motion algorithm applied to any suitable data that is already available from the viewer.

To demonstrate that our methods works effectively, we applied it to texture a laser scan of building faades at the Bremer Market Square in Germany with a screenshot from Google Streetview, using a focal length estimation made from views of the 'Arc de Triomphe' in Paris, France. The results indicate that our method allows to easily make realistic-looking digital urban 3D models by combining pre-recorded data from completely unrelated sources.

2. PREVIOUS WORK

A promising way to easily generate photo-realistic urban 3D models is the use of Structure from Motion (SfM). Snavely et al. (2008) have shown that photos from the internet can be used with SfM to automatically reconstruct buildings and small parts of cities. This produces fully textured 3D models without even having to visit a site. Unfortunately, it requires having at least 3 good quality photos available on the internet of every building that needs to be reconstructed. This limits the application of

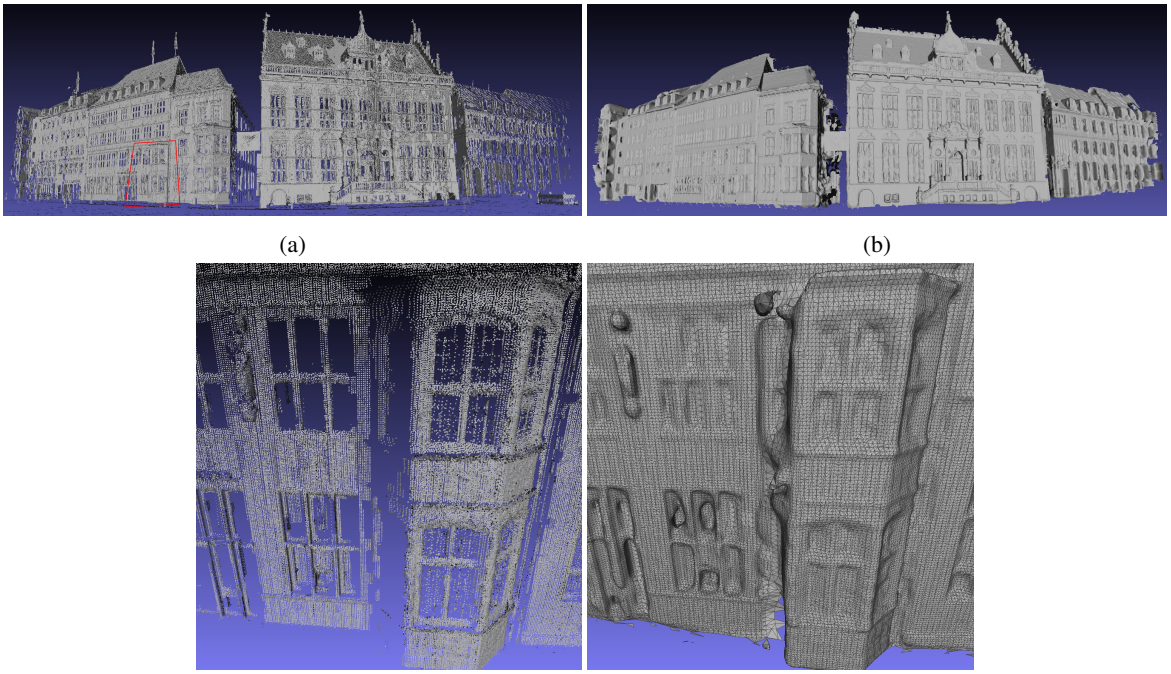


Figure 1: 3D Surface reconstruction from point clouds. (a) Point cloud extracted from a laser scan data set taken in Bremen, Germany. (b) Mesh surface obtained using Poisson surface reconstruction on the individual building faades. (c) Close-up view of the point cloud for the selected area in subfigure (a). (d) Close-up view of the triangulation.

this method to the reconstruction of popular tourist attractions. Furthermore, the 3D data from such reconstructions usually contain many missing parts and inaccuracies. Combining SfM results with more accurate 3D data from other sensors, such as laser scanners, still requires additional data alignment solutions.

Several methods have been suggested to automatically align a 2D image to a 3D model, by computing some sort of similarity measure between 2D image contents and structure of the 3D model. Lensch et al. (2000) proposed a silhouette-based alignment method that includes an automatic initialisation. However, the procedure requires an isolated object that is entirely visible in the photo, which is often not the case in 3D scans of urban environments. Viola and Wells III (1997) proposed to use surface normals to compute mutual information between an image and a 3D model. Corsini et al. (2009) improved on this by including more kinds of data to calculate mutual information. Although the mutual information registration methods can be applied to urban models, a sufficiently close pose and focal length initialisation has to be supplied manually for each photo, which comprises of the simultaneous adjustment of 7 parameters (3D location + 3D orientation + focal length). This might easily require more time and skill than annotating 3 or more tie points.

Morago et al. (2014) have chosen to automate the alignment of new photos with LIDAR scans by 2D feature point matching with photos that were taken together with the LIDAR scans, making use of their available 3D registration. However, this approach does not solve the problem of how to align photos with 3D scans for which no photos have been registered yet.

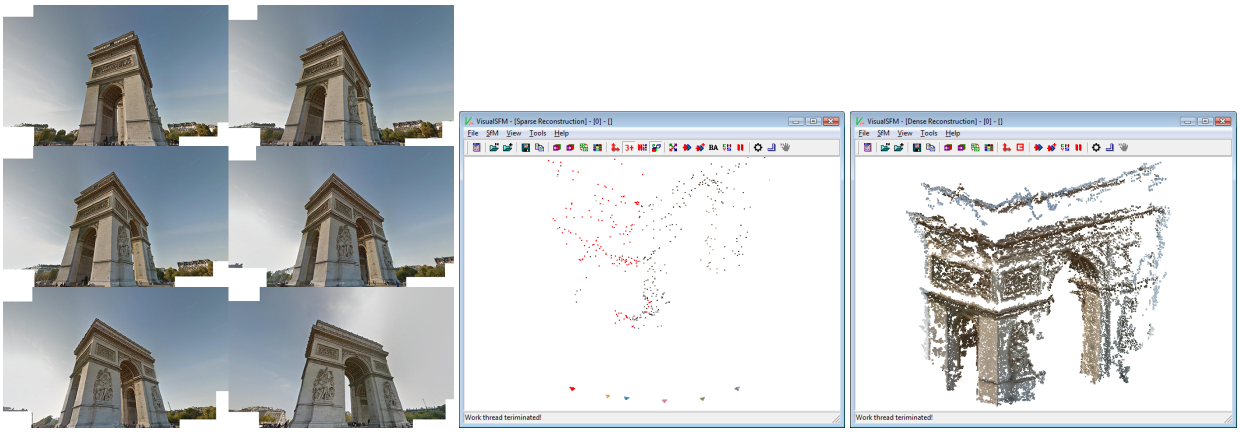
Using tie points to align 2D images to 3D models is a well-studied problem, for which many effective solutions already have been found such as by Hesch and Roumeliotis (2011) and Zheng et al. (2013). Accurate 2D-3D alignment can already be achieved reliably with only three or more well-spread tie-points (or at least four to prevent multiple ambiguous solutions). These methods make use of the perspective projection (pinhole camera) model and therefore require calibration of a camera's intrinsic parameters. A requirement which also holds for projecting a photo onto a 3D model once its correct pose is estimated. For self-made photographs, intrinsic camera calibration may be done by taking several photographs of a calibration pattern with known structure. However, this cannot be done for photos from an image database such as Google Streetview.

3. MAIN CONTRIBUTIONS

The main contributions of our proposed procedure that set it apart from previously proposed methods are as follows: - Our method does not only work with self-made photographs, but also with screen shots from panoramic viewers such as Google Streetview, which already contain a large coverage of terrestrial urban photographs. - Since it doesn't depend on a 3D reconstruction from photogrammetry other than for camera calibration, it works with any source of 3D data and requires only one photo to texture any surface area visible in that photo. - Three or more tie points per camera are sufficient, without the need for a manual initialisation of camera pose. - Since the method is fully tie-point-based, as long as 3 or more tie points can be provided accurately, it works on any type of scene, any photo quality and doesn't require segmentation of individual buildings or objects. - Free software implementations are currently available for all of the steps that our procedure requires to convert a 3D point cloud into a textured 3D mesh. To our knowledge, we are the first to propose a procedure that covers a complete work flow with all of these advantages that are crucial for obtaining large-scale urban 3D maps.

4. TEXTURING A 3D MESH SURFACE WITH A PHOTO

In principle, texturing a 3D model from a photo is the inverse of how the light from the scene was projected onto the photo.



(a) (b) (c)

Figure 2: Structure from Motion for calibrating the focal length of Google Streetview. (a) Google Streetview screen shots with the on-screen information covered with white rectangles. (b) Reconstruction of the 6 camera poses and 3D feature point locations using VisualSFM. (d) Denser surface reconstruction using the CMVS plugin of VisualSFM.

By projecting the image points back into the 3D scene from the same relative position and orientation, the visible surface of the 3D model can be coloured correctly from the photo. This can be done using the freely available Meshlab software by Cignoni et al. (2008). The procedures described below are demonstrated in more detail in our three video tutorials: Lichtenauer and Sirmacek (2015a,b,c).

4.1 Generating a Surface Model

Texturing requires a surface, so a point cloud must be converted into a 3D mesh first, as shown in figure 1. This can be done with the Poisson surface reconstruction method in Meshlab. We assume that the point cloud is clean and only includes points which are representing the faade surface samples (fig. 1(a)). If the point cloud is too noisy to represent the faade surface correctly, or else if there are occluding objects, the point cloud must be pre-processed before mesh generation. This pre-process must remove the points coming from noise and occlusion effects. The best results are achieved by reconstructing relatively straight parts individually. Therefore, it will help to segment building point clouds into their separate sides, reconstruct their surfaces separately and then combine the surface meshes of all the sides again afterwards (fig. 1(b)). Note that the Poisson reconstruction method in Meshlab requires surface normals of the point cloud to be estimated first.

4.2 Obtaining intrinsic camera parameters from SFM

To correctly estimate camera pose and to project an image accurately onto a 3D surface, we need to know the intrinsic parameters of the camera that was used to generate the photo. The intrinsic camera parameters are coefficients of a mathematical approximation that relates the pinhole camera model to the actual pixel locations in a photograph. In a pinhole camera model, a point in a 3D scene is represented by the 2D location of the intersection of the straight line between the scene point and the camera's focal point (the pin hole) with a flat surface at focal distance f from the focal point.

The most common way to obtain intrinsic camera parameters is by performing a separate calibration procedure in which additional photographs are taken from a known calibration pattern or -structure. However, such procedures require extra time, equipment and skills. Furthermore, in some cases a separate calibration might not even be possible at all. For instance, when the camera that has been used for the texture photographs cannot be accessed, or when a camera is used that has a variable focal length (zoom function). An example of such a situation might be when someone wants to texture a 3D mesh with a photograph from a public database such as Google Streetview Google Streetview - Google Maps (n.d.), which doesn't include details about the viewer's intrinsics.

A panoramic viewer such as the one for Google Streetview uses a single 360-degree spherical per location. Since all source photos have already been converted to the same spherical camera model that the panoramic viewer requires, it doesn't matter where the photos have been taken and what equipment has been used. All we need to do is to calibrate the viewer's own intrinsic parameters. Note that the view that is being shown to the user is only a part of the whole spherical image, converted to a perspective view centred around the viewing direction chosen by the user. Since Google Streetview currently has no radial distortion added, The only intrinsic parameter we need to estimate is the focal length. Because focal length is measured in screen pixels, it depends on the user's screen resolution. This means that the focal length has to be estimated separately for each screen resolution at which screen shots are taken from Google Streetview for texturing.

To estimate the intrinsic parameters for a public database such as Google Streetview, without being able to perform a regular camera calibration procedure, we propose to use Structure from Motion (SfM). Besides generating a 3D structure of the photographed scene, SfM software also provides the estimates of the intrinsic and extrinsic camera parameters for each of the source photographs. And since the focal length of Google Streetview does not depend on the location, we can choose a suitable scene anywhere in the world. For our example, we have chosen the Arc de Triomphe in Paris, France. This structure has views available from all around at small steps, which is exactly what is required for SfM.

Figure 2(a) shows the 6 screen shots we have taken from Google Streetview, with the on-screen information boxes removed to prevent false feature point matches between the images. Figure 2(b) shows the result of estimating structure (3D feature point locations) from motion from these images with the free VisualSFM software created by Wu (2011); Wu et al. (2011). We have disabled radial distortion in this VisualSFM, since the images were already distortion-free. By forcing the different camera views to

have the same intrinsic parameters estimated, accuracy can be increased (since all images of come from the same panoramic viewer). However, performing the initial sparse SfM reconstruction in Visual SFM with 'shared calibration' enabled prevents the SfM algorithm from converging to the right solution. Instead, the initial SfM result without shared calibration can be refined afterwards using the Bundle Adjustment (BA) step with the 'shared-calibration' option enabled.

After the sparse SfM reconstruction, VisualSFM can also make a denser reconstruction, by using the CMVS option. For the purpose of texturing a 3D mesh with a photograph, we don't need a 3D reconstruction from SfM for any other purpose than to estimate the intrinsic camera parameters. However, the dense reconstruction is useful to visually inspect whether SfM has succeed. If some angles that should be straight have not been reconstructed perfectly straight, it means that the intrinsic camera parameters also have not been estimated correctly. Figure 2(c) shows the dense reconstruction for our example of the Arc de Triomphe. Furthermore, executing the CMVS procedure in VisualSFM also generates some output files that are useful for loading aligned texture photos into Meshlab.

The file 'bundle.rd.out' can be opened with Meshlab to load aligned rasters for texturing. For a description of the Bundler file format, see Snavely (2008-2009). It contains a list of intrinsic parameters for each camera view used in SfM, followed by a list of all structure points. When loading a Bundler file into Meshlab, it also asked for a text file containing a list of images corresponding to the cameras defined in the 'rd.out' file. The focal length of Google can also be found in the 'bundle.rd.out' file as the first value for each camera. The focal lengths for all views should be almost the same after having used Bundle Adjustment with the 'shared-calibration' option.

When using self-made photos for texturing which might include radial distortion, another possibility is to use VisualSFM on several photos taken from the scene that needs to be textured. VisualSFM will then not only calculate the intrinsic parameters for your camera automatically, but also rectify any radial distortion that might be present. After performing the CMVS procedure in VisualSFM, the rectified versions of your photos can be found in the 'visualise' directory of the output from VisualSFM, which correspond to the camera parameters in the 'bundle.rd.out' file in the order specified in the list.txt file. These rectified photos are suitable for texturing 3D lasers scans.

VisualSFM uses the following camera model that relates pinhole model coordinates $(\mathrm{n}_x, \mathrm{n}_y)^\top$ to image coordinates $(\mathrm{m}_x, \mathrm{m}_y)^\top$ (which are relative to the image center):

$$\begin{bmatrix} \mathrm{n}_x \\ \mathrm{n}_y \end{bmatrix} = f \begin{bmatrix} \mathrm{X}_c/\mathrm{Z}_c \\ \mathrm{Y}_c/\mathrm{Z}_c \end{bmatrix} \quad (1)$$

$$= \left(1 + r(\mathrm{m}_x^2 + \mathrm{m}_y^2)\right) \begin{bmatrix} \mathrm{m}_x \\ \mathrm{m}_y \end{bmatrix}, \quad (2)$$

where $(\mathrm{X}_c, \mathrm{Y}_c, \mathrm{Z}_c)^\top$ are the 3D coordinates of the visible point with respect to the camera frame. The above camera model has only two intrinsic camera parameters: the focal length f and a radial distortion coefficient r. This model assumes that the image center coincides with the optical center, that the camera pixels are perfectly square and that the lens distortion is quadratic, without higher order components.

By using the f and r estimated by the SfM software, equations 1 and 2 can also be used to obtain the texture colour of a 3D point that is visible in the image at pixel coordinates $(\mathrm{m}_x, \mathrm{m}_y)^\top$. For faster texturing of many 3D points, a rectified image can be generated using equation 2. When the SfM procedure includes the image that is to be used for texturing, a rectified image is already generated by VisualSFM itself. In the rectified image, the pixel coordinates are corresponding to the pinhole model coordinates $(\mathrm{n}_x, \mathrm{n}_y)^\top$. Therefore, texturing of 3D points can be done with a rectified image by only applying equation 1. Furthermore, this also allows to use third-party software, such as the texture function of Meshlab, which we use for our example.

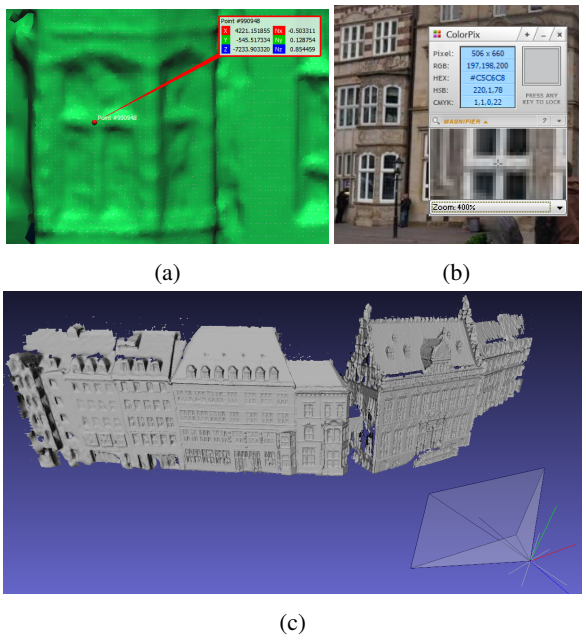


Figure 3: Pose estimation from tie point annotation. (a) Mesh vertex selected to see 3D point location and surface normal information. (b) Retrieving corresponding image coordinates of the same point. (c) Camera pose estimated from multiple tie points.

4.3 Obtaining extrinsic camera parameters

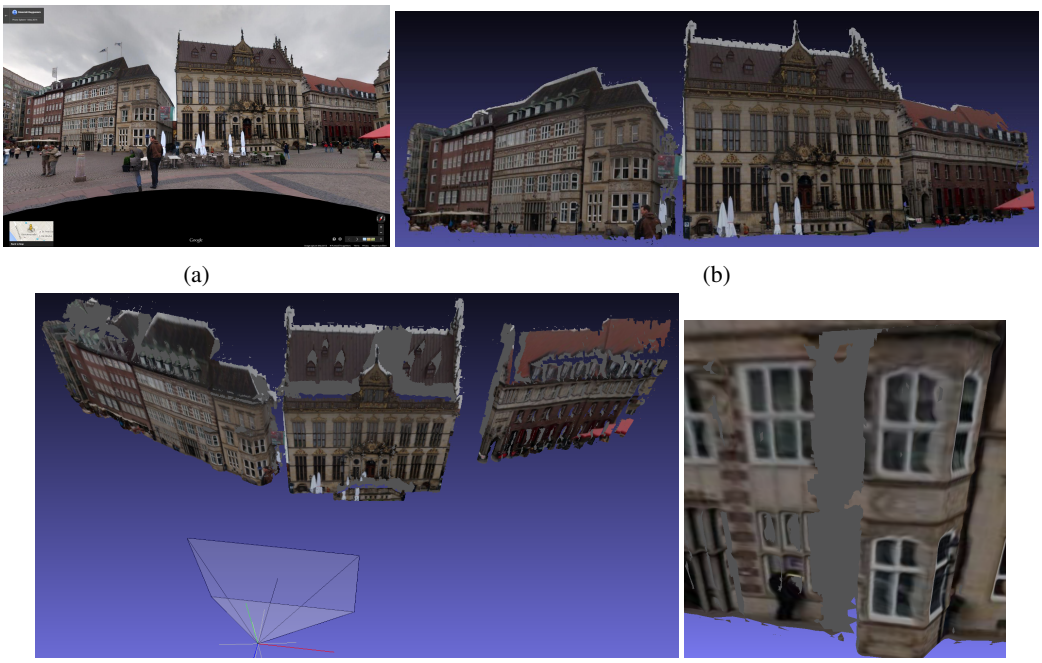
Equation 1 requires the 3D coordinates of points to be defined in the reference frame of the camera. The coordinates of a point in the scanned 3D structure $(\mathrm{X}_s, \mathrm{Y}_s, \mathrm{Z}_s)^\top$ are converted to the reference frame of the texture camera by:

$$\begin{bmatrix} \mathrm{X}_c \\ \mathrm{Y}_c \\ \mathrm{Z}_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} \mathrm{X}_s \\ \mathrm{Y}_s \\ \mathrm{Z}_s \end{bmatrix} + \mathbf{T}, \quad (3)$$

$$\text{where} \quad \mathbf{R} = \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} & \mathrm{R}_{13} \\ \mathrm{R}_{21} & \mathrm{R}_{22} & \mathrm{R}_{23} \\ \mathrm{R}_{31} & \mathrm{R}_{32} & \mathrm{R}_{33} \end{bmatrix}, \mathbf{T} = \begin{bmatrix} \mathrm{T}_1 \\ \mathrm{T}_2 \\ \mathrm{T}_3 \end{bmatrix}.$$

The rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{T}$ are the extrinsic camera parameters (or 'camera pose') of the texturing camera with respect to the 3D mesh that has to be textured.

The SfM procedure described above also computes extrinsic camera parameters $\mathbf{R}$ and $\mathbf{T}$. However, these are only suitable for alignment with the 3D reconstruction made by SfM. They cannot be used for texturing another 3D model, such as a laser scan, which will have an entirely different scale and alignment. Fortunately, the camera pose with respect to any 3D structure can be accurately estimated from as little as three point correspondences between the photo and the surface of the 3D structure. To prevent multiple solutions from which the correct one has to be validated by visual verification, at least four tie points are needed. Hesch



(a) (b) (c) (d)

Figure 4: Texturing a laser scan with a Google Streetview screen shot (a) The screen shot from Google Streetview. (b) The textured 3D mesh seen from the view of the Google Streetview screen shot. (c) The textured 3D mesh from a different angle, with parts occluded from the camera view coloured in gray. (d) Close up of the textured 3D mesh from a different angle.

and Roumeliotis (2011) presented a Direct Least-Squares (DLS) method for computing camera pose which, unlike some other methods, doesn't require any initial (rough) pose estimate. The Matlab code can be downloaded from their website. Although the authors propose to refine the result of their DLS method with an additional Bundle Adjustment (BA) step, the accuracy of the DLS method is already very accurate by itself. For the purpose of texturing a 3D surface, the improvement of the additional BA refinement will likely be unnoticeable.

The tie points can either be determined by manual annotation, or by an automatic procedure that matches corner points in the image with corner points of the 3D structure. Figure 3(a) and (b) show how a tie point can be found through manual annotation. The 3D location of a vertex on a 3D mesh surface can be found using the point picker option of the free software CloudCompare $EDF\ R\&D$ Telecom ParisTech, CloudCompare (version 2.6.1).

To obtain the most accurate texture mapping, it is best to choose the points as far apart as possible, unless a significant lens distortion is present, in which case points close to the photo border might not be accurately approximated by the radial distortion model. For ease of computation, use the rectified version of the texture photograph to determine the image coordinates of the points (so that the radial distortion represented by equation 2 can be ignored). Note that the DLS method requires normalised image coordinates with unit focal length: $\frac{1}{f}(\mathrm{n}_x, \mathrm{n}_y)^\top$.

The Matlab code from Hesch and Roumeliotis (2011) to estimate camera pose from tie points can be used with the free software Octave Eaton et al. (2015). To convert the estimated pose to the format of Bundler, it has to be flipped over, using:

$$\mathbf{R}_b = \mathbf{F}\mathbf{R} \quad (4)$$

$$\mathbf{T}_b = \mathbf{F}\mathbf{T} \quad (5)$$

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (6)$$

4.4 Projecting a 2D Image on a 3D Mesh Surface

Texturing from a photo can be done with the Meshlab software, using the option 'parameterization + texturing from registered rasters'. To load the (rectified) texture photos and their corresponding focal length and extrinsic calibration into Meshlab, the Bundler file format can be used, as mentioned above. This allows to load multiple photos at once, for combined texturing. The texturing procedure of Meshlab has several options of how to combine overlapping photos with possibly different illumination conditions. Figure 4(a) shows the screenshot from Google Streetview that we used for our example. Figure 4(b) shows the result from texturing from the same viewing angle as the Google Streetview screen shot. Figure 4(c) shows the texture result under a different angle, which includes parts that were occluded from the view and thus have no texture information. These occluded parts have been coloured gray. Figure 4(d) shows a close up of the texture result.

5. SHOWCASE EXAMPLE AND DISCUSSION

Our 3D texturing procedure consists of a novel combination of free software implementations of pre-existing algorithms for Structure from Motion, pose estimation and texture mapping. For an evaluation of the performances of each of the selected methods, please refer to our references to their respective literature in the previous sections.

To show the effectiveness of our complete procedure and give an impression of the quality that can be obtained, we have demonstrated the required steps above using a challenging example that we find representative of a common practical situation for which our procedure is intended to work. The 3D point cloud in our example was derived from terrestrial laser scans made with the Riegl VZ-400 scanner of a row of buildings at the Market Square in Bremen, Germany. It was part of a larger dataset collected by Borrmann and Nüchter (n.d.) and available for download. The

show case example shows that the method is easy to use and that the accuracy of the method can be very high depending on the tie point selection.

The accuracy of the final texture alignment depends on several factors. First of all, the accuracy of the 3D model. When the photo for texturing is not made from the same location as the 3D scan, an error in the estimated distance from the 3D scanner will result in a shift in the projection from the photo.

Secondly, the tie points must be selected accurately at the same location in the photo that corresponds to the location selected on the 3D model. A tie point misalignment results in an inaccuracy of the camera pose estimation. Random errors in tie point alignment can be compensated by taking a larger number of tie points.

Thirdly, the intrinsic camera parameters must be estimated accurately. Lens distortion must be removed completely. This is not possible if the lens that was used has a more complex distortion than what is modelled by the (radial) lens distortion model. Unfortunately, Google Streetview contains a lot of misalignments in the multiple views that have been combined to generate the spherical images. This will cause inaccuracies in the tie point alignment (leading to inaccurate pose estimation), as well as misalignment of the mapped texture. The effects of this can be mitigated by carefully selecting the tie points at places that seem undistorted and by combining multiple views from different positions, to not use the edges of the photos, which will be effected the worst. An error in the focal length estimation from Structure from Motion will also cause texture misalignment, but by using the same inaccurate focal length for pose estimation as for texturing, the effect of an error in focal length is relatively small and distributed over the middle and outer parts of the photo.

Lastly, high quality texturing obviously depends on the quality of the photos that are used. A complete texturing of all parts of a 3D model requires well-placed views that do not leave any parts of the 3D surface unseen. Furthermore, differences between lighting conditions and exposures of the different photos can cause edges to be visible where the texture from one photo transitions into that of another.

6. CONCLUSION

Generation of realistic 3D urban structure models is valuable for urban monitoring, planning, safety, entertainment, commercial and many other application fields. After having the 3D model of the structure by laser scanning, photogrammetry or using CAD design software tools, the most realistic views are obtained after texturing the 3D model with real photos. Google streetview provides opportunity to access photos of urban structures almost all around the world. In this study, we introduce a semi-automatic method for obtaining realistic 3D building models by texturing them with Google streetview images. We discuss advantages and also the limitations of the proposed texturing method.

ACKNOWLEDGEMENTS

This research is funded by the FP7 project IQmulus (FP7-ICT-2011-318787) a high volume fusion and analysis platform for geospatial point clouds, coverages and volumetric data set.

References

Borrmann, D. and Nüchter, A., n.d. 3d scan data set recorded in city center of bremen, germany as part of the thermalmapper project. http://kos.informatik.uni-osnabrueck.de/3Dscans/.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F. and Ranzuglia, G., 2008. Meshlab: an open-source mesh processing tool. In: Sixth Eurographics Italian Chapter Conference, pp. 129–136.

Corsini, M., Dellepiane, M., Ponchio, F. and Scopigno, R., 2009. Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties. Computer Graphics Forum 28(7), pp. 1755–1764.

Eaton, J. W., Bateman, D., Hauberg, S. and Wehbring, R., 2015. Gnu octave version 4.0.0 manual: a high-level interactive language for numerical computations. http://www.gnu.org/software/octave/doc/interpreter.

EDF $R\&D$ Telecom ParisTech, CloudCompare (version 2.6.1), n.d. http://www.cloudcompare.org/.

Google Streetview - Google Maps, n.d. https://www.google.com/maps/views/streetview.

Hesch, J. and Roumeliotis, S., 2011. A direct least-squares (dls) method for pnp. In: Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 383–390.

Lensch, H. P. A., Heidrich, W. and peter Seidel, H., 2000. Automated texture registration and stitching for real world models. In: in Pacific Graphics, pp. 317–326.

Lichtenauer, Jeroen, F. and Sirmacek, B., 2015a. Texturing a 3d mesh with google streetview 1 of 3: Manual alignment. http://youtu.be/Nu3VaeBxGHc.

Lichtenauer, Jeroen, F. and Sirmacek, B., 2015b. Texturing a 3d mesh with google streetview 2 of 3: Estimate focal length with visualsfm. http://youtu.be/0HZDuI48IHc.

Lichtenauer, Jeroen, F. and Sirmacek, B., 2015c. Texturing a 3d mesh with google streetview 3 of 3: Calculate pose from tie points. http://youtu.be/8i86ys9Boqc.

Morago, B., Bui, G. and Duan, Y., 2014. Integrating lidar range scans and photographs with temporal changes. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pp. 732–737.

Snavely, N., 2008-2009. Bundler v0.4 user's manual. http://www.cs.cornell.edu/~snavely/bundler/.

Snavely, N., Seitz, S. and Szeliski, R., 2008. Modeling the world from internet photo collections. International Journal of Computer Vision 80 (2), pp. 189–210.

Viola, P. and Wells III, W. M., 1997. Alignment by maximization of mutual information. Int. J. Comput. Vision 24(2), pp. 137–154.

Wu, C., 2011. Visualsfm: A visual structure from motion system. http://ccwu.me/vsfm/.

Wu, C., Agarwal, S., Curless, B. and Seitz, S. M., 2011. Multicore bundle adjustment. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR).

Zheng, Y., Kuang, Y., Sugimoto, S., Astrom, K. and Okutomi, M., 2013. Revisiting the pnp problem: A fast, general and optimal solution. In: Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 2344–2351.