Vision, Modeling, and Visualization (2010)

Interactive Multi-View Façade Image Editing

Przemyslaw Musialski [1,2] Christian Luksch [1] Michael Schwärzler [1] Matthias Buchetics [1]
Stefan Maierhofer [1] Werner Purgathofer [1,2]

[1] VRVis Research Center, Austria [2] Vienna University of Technology, Austria

Figure 1: *Steps of the proposed multi-view image generation system. Top-left: one of typical perspective input photographs, please note the occlusion. Top-middle: the result of the proposed ortho-image generation method (note the pedestrians). Second row shows masks indicating source images of the composition by colors: automatic result (left) and interactively post-processed (middle). Right: the final result after interactive post-processing.*

Abstract

We propose a system for generating high-quality approximated façade ortho-textures based on a set of perspective source photographs taken by a consumer hand-held camera. Our approach is to sample a combined orthographic approximation over the façade-plane from the input photos. In order to avoid kinks and seams which may occur on transitions between different source images, we introduce color adjustment and gradient domain stitching by solving a Poisson equation in real-time. In order to add maximum control on the one hand and easy interaction on the other, we provide several editing interactions allowing for user-guided post-processing.

Categories and Subject Descriptors (according to ACM CCS): Computing Methodologies [I.3.3]: Computer Graphics—Picture/Image generation; Computing Methodologies [I.4.9]: Image Processing And Computer Vision—Applications;

1. Introduction

Modeling and reconstruction of urban environments is currently the subject of intensive research. There is a wide range of possible applications, including virtual environments like cyber-tourism, computer games, and the entertainment industries in general, as well as urban planning and architecture, security planning and training, traffic simulation, driving guidance and telecommunications, to name but a few. The research directions are spread across the disciplines of computer vision, computer graphics, image processing, photogrammetry and remote sensing, as well as architecture and the geosciences. Reconstruction is a complex problem and requires an entire pipeline of different tasks.

In this work we address the problem of texture generation, which remains a challenging task. The generation of high-quality façade imagery is a key element of realistic representation of urban environments. Ortho-rectified façades are also a prerequisite of several structure detection and segmentation algorithms [MZWvG07, MRM*10]. Our contribution is a system which provides the ability to create such images from a set of perspective photographs taken by a consumer hand-held camera. The novelty is a method for detailed removal of occluders by exploiting the multi-view information. It combines robust automatic processing steps with user interaction and is meant to resolve the still remaining weak points of fully automatic attempts and to improve the quality of the output.

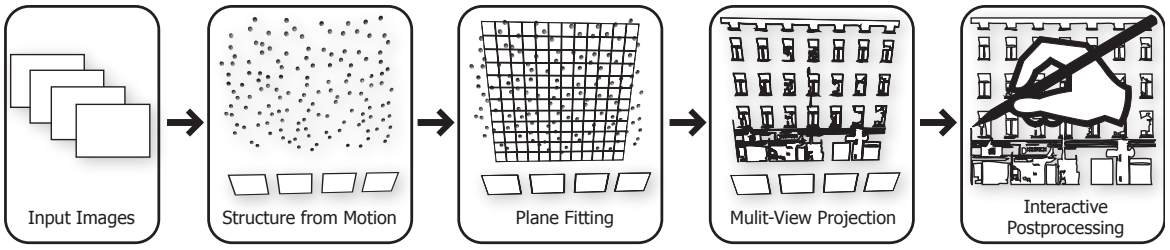

Figure 2: *Schematic overview of our system: we compute a sparse point cloud of the scene using structure-from-motion; then, we fit a dominant plane to the point cloud. Next, we project the images of the shots onto the plane and store their colors in a per pixel stack. Finally, we allow the user to brush over the stack in order to remove unwanted content by choosing the best source.*

1.1. Related Work

Projective Texturing and Image-Based Rendering. One of the pioneering works was the "Façade" system introduced by Paul Debevec *et al.* [DTM96]. Their paper proposes an interactive modeling tool that allows the user to model 3d architecture from photographs under the constraints of epipolar geometry, and to sample projective textures on building façades. There have been a number of parallel and follow-up publications aiming at urban modeling from images [LCZ99, CT99], which utilized the projection of photographs in order to obtain approximated ortho-images.

More recent approaches introduce semi-automatic systems that support the user during the modeling process. They are based on input from video [vdHDT*07] or image collections [SSS*08, XFT*08]. These systems introduce texture sampling as part of their modeling pipeline. Both latter approaches resort to user interaction in order to improve the quality of the results. Although similar to ours, they do not focus on textures so much as we do. We purely focus on texture generation and describe all details of this part of the urban reconstruction process.

Various tools for interactive, projective texture generation, improvement and synthesis for architectural imagery has been also presented [PSK06,ELS08,MWR*09], but with different objectives as ours. Recently Xiao *et al.* presented an automated attempt at the modeling and texturing of street sites [XFZ*09], which suffers quality loss when compared to semi-interactive methods.

Another branch are feature-based sparse reconstruction methods, which also make use of projective imaging [SSS07,SGSS08]. They are related to our system also in that structure-from-motion is used for generation of the proxy geometry. This issue is handled in more detail in Section 2.1. Finally, there are methods which do not focus on architecture, but on the problem of projective texturing in general [NK01, TS08, GWOH10].

Image-based rendering methods [DYB98, EDM*08], in contrast to ours, aim more at real-time rendering performance than at high-quality images.

Multi-View and Panorama Imaging. The goal of those methods is to generate views with more than one viewpoint in order to provide novel insights into the given data. Often, the image is generated along a path of camera movement, referred to as strip panorama. One such variant are pushbroom images, which are orthogonal along the horizontal axis [GH97, SK03], and the similar x-slit images presented by Zomet *et al.* [ZFPW03]. Others proposed systems for generation of strip-panoramic images as well [Zhe03, RGL04]. Agrawala *et al.* [AAC*06] aims at the creation of long multi-view strip panoramas of street scenes. Optimal source images for particular pixels are chosen using a constrained MRF-optimization process. While our approach shares several ideas with them, our focus lies on as orthographic a projection as possible, and on the removal of all disturbing occluders as well, in order to provide high-quality façade texture.

Image Stitching. The stitching of two signals of different intensity usually causes a visible junction between them. An early solution to this problem were transition zones and multi-resolution blending [BA83]. Pérez *et al.* [PGB03] introduced a powerful method for this purpose: image editing in the gradient domain. There is a number of further papers tackling, improving, accelerating and making use of this idea [PGB03, ADA*04, Aga07, MP08]. Recently, McCann *et al.* [MP08] introduced an interactive painting system which allows the user to paint directly in the gradient domain, and the Poisson equation is solved online by a GPGPU solver. Also Jeschke *et al.* proposed a real-time solver [JCW09]. The foundations behind the gradient domain image editing method are described in the aforementioned papers as well as in the ICCV 2007 Course-Notes [AR07]. For the completeness, we shall provide a brief overview of this approach in Section 2.4.

1.2. Overview

The goal of this work is to provide a convenient and robust way to generate approximations of ortho-rectified images of building façades. The only input we use is a set

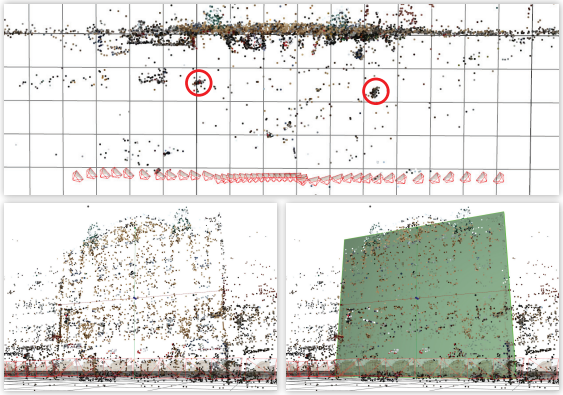

Figure 3: *Top: top view on the point cloud computed by the structure-from-motion (SfM) module. The dominant plane is clearly detectable. The circles indicate objects in front of the façade. Bottom left: frontal view of the point-cloud, right: with plane fit into it.*

of photographs of the targeted building taken from the ground using a hand-held, consumer-level camera. These images have to be registered to each other, thus we present a brief overview of multi-view registration and structure-from-motion in Section 2.1. We expect the object in front of the cameras to be approximately planar, like a single façade, such that it can be substituted by simple geometry, which we call *proxy geometry*. In Section 2.2 we propose one possible solution to this problem. In Section 2.3 we describe the details of the multi-view projection method. Our approach is straightforward: we span a grid of desired resolution over the façade-plane. Then, for each pixel in the target resolution we determine which camera shot is optimally projecting onto it, and we collect its color information. At this point two problems arise: The first occurs if two neighboring pixels in the target resolution are filled by color samples from different source images. Usually this results in a visible seam between them. To resolve this we propose color correction and gradient-domain stitching. This is handled in Section 2.4. The second problem relates to the actual image content. For some shots we might obtain color samples which belong to external objects that occlude the façade, like vehicles, vegetation, etc. We approach this in a semi-automatic manner in Section 2.5 and by turning to user interaction in Section 2.6. Ultimately, the final image is composed according to the automatic and manual corrections in the gradient-domain and an online Poisson solver provides the result (Section 3). Figure 2 provides an overview over the mentioned pipeline.

2. Multi-View Ortho-Rectification

2.1. Structure From Motion

We resort to the classic sparse stereo *structure-from-motion* (SfM) method to register the images to one another and to orient and position them in 3d space. This method is based on feature matching, pose estimation, and bundle adjustment [PvGV*04]. Multiple photographs are provided to the modlue and from each one a sparse set of SIFT feature-points is extracted [Low04]. Once multiple images with corresponding features have been established, the extrinsic (i.e., pose in 3d space) properties of their cameras can be determined. Since we are dealing with mostly planar objects, we use a calibrated approach for unstructured photographs, such as the one described by Irschara *et al.* [IZB07]. In accordance with epipolar geometry given known camera parameters, the 3d positions of the corresponding 2d features in the photos can be triangulated, which provides a cloud of 3d space points.

2.2. Proxy Geometry

Plane Fitting. The SfM procedure delivers a sparse point-cloud of the triangulated points in 3d space. If we have not encountered any serious mismatches between the photographs, the points are distributed such that they form a more-or-less coherent planar manifold of the 3d space (cf. Figure 3). In order to compute the proxy geometry, we introduce a rudimentary plane detection algorithm based on RANSAC [FB81] for outlier removal followed by least squares fitting. Let the set of the 3d points be $\mathbf{X} = \{\mathbf{x}\}_{i=1}^{n}$. In the following, we perform RANSAC on the set such that we obtain only a thin layer of the points $\mathbf{X}^* \subseteq \mathbf{X}$. The "thickness" of the layer is controlled by the distance threshold ε of the RANSAC procedure. Next, the plane is defined by a 4d vector π composed of the normal $\mathbf{n}$ and the distance to the origin d. We perform a least squares fit by minimizing the sum of squared distances of all points $\mathbf{x} \in \mathbf{X}^*$ to π:

$$E_\pi = \sum_i \|\mathbf{n}^T \mathbf{x}_i - d\|^2 \quad \longrightarrow \min \ .$$

Façade Boundary. So far we have a set of registered shots including their camera properties, a sparse point cloud in 3d space and a dominant plane fitted into the cloud. At this stage there arises the problem of defining the actual façade extent. While there have been attempts to solve such problems automatically, these are error prone and not well defined. On the

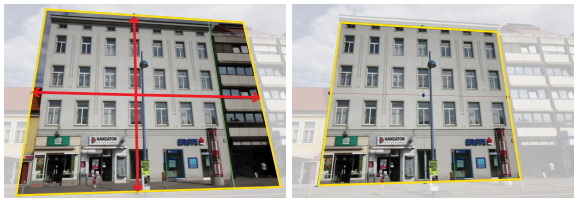

Figure 4: *View at the façade plane through one of the projecting cameras. In this view it is easy to adjust the façade-bounds interactively. Left: during the adjustment. Right: final result.*

other hand, this is quite an easy task for a human provided with an appropriate user interface. For this reason, we propose a GUI that allows the user to (1) navigate in 3d though the scene, (2) look at the scene from the computed shot positions, (3) preview the texture by projecting best single-shot image onto the plane, (4) adjust the 3d plane by resizing and rotating it (see Figure 4) and, finally, (5) align the coordinate system of the scene with the one of the proxy plane. After the adjustment of the façade boundary, the application is ready for the next step: multi-view projective texturing.

2.3. Viewpoint Projection.

Scene Geometry. We distinguish different cases of camera placement, where only one is valid and the others are classified as invalid and shots of this class are rejected. Figure 5 depicts this issue: the invalid cases occur when the camera is behind the plane ($\mathbf{C}_3$ and $\mathbf{C}_4$) or when it is in the front, but not all four rays from its center through the corners of the frustum intersect the image plane ($\mathbf{C}_1$). The valid case is when the camera is in front of the façade plane and all rays intersect the image plane in a finite distance, such that the projected shape is a finite trapezoid that intersects the façade rectangle (cf. Figure 5, left). If not all rays intersect the plane, only a part of the image is finitely projected onto the plane and a part meets the plane at a line at infinity. Pixels from such a projection are very strongly elongated along the plane and thus prone to cause sampling artifacts. Since we expect to have enough information from the valid cameras anyway, we simply reject them as invalid ones.

Shot Selection. Our approach is based on the fact that we have multiple projective centers along the horizontal axis in world space (since we are using ground-based hand-held cameras). This allows us to compose the target image I in such a way that each pixel is chosen from an optimal camera. As a measure for this optimality, we use an objective function composed of the camera to plane-normal incidence angle φ and a term which expresses the area covered by the footprint of the original pixel projected onto the proxy plane.

From the *law of sines* we know that the length of a projected segment depends on the distance of the camera center to the plane and the projection angle. Figure 6, left hand side, depicts this relation, where the length of the segment $\overline{BC}$ depends on the angles α, φ_1, and the length of $\overline{AM}$.

We denote the distance of each camera $\mathbf{c}_k$ to each pixel in the target image $\mathbf{p}_i$ as d_{ik}, then we approximate the projection area as $A_{ik} = (d_{ik}/d_{max})^{-2}$. We normalize d_{ik} such that it lies between 0 and 1, which is a chosen maximum distance d_{max} (i.e. the most distant camera). For the angular term, we use the dot product of the plane normal and the normalized vector $\mathbf{v}_{ik} = \|\mathbf{c}_k - \mathbf{p}_i\|$, such that: $B_{ik} = \mathbf{n}^T \mathbf{v}_{ik}$. This value is naturally distributed in the range $0 \ldots 1$. Both terms are weighted by the empirical parameters $\lambda_1 + \lambda_2 = 1$, such that the final objective function is:

$$E_I = \sum_i \sum_k \lambda_1 A_{ik} + \lambda_2 B_{ik} \quad \longrightarrow \max , \qquad (1)$$

where i iterates over all target pixels and k over all valid cameras. We choose $\lambda_2 = 0.7$ in our experiments.

Image Stacks. In order to accumulate the projections, we span a grid of desired resolution over the detected and bounded façade plane. Then, for each pixel in the target resolution, we determine a set of cameras which project optimally according to the aforementioned constraints. We store these values in a volume of the size *width* $\times$ *height* $\times$ *number of shots* attached to the proxy, which we call *image stack* due to its layered nature. Right hand side of Figure 6 shows a schematic, 2d top view of this idea.

2.4. Seamless Stitching

One remaining problem are the visible seams along transitions between pixels from different sources, which we address by a gradient-domain stitching algorithm.

GPU Poisson Solver. As presented in Section 1.1, Poisson image editing dates back to [PGB03]. The beauty of this method manifests itself in both the elegance of its formulation and the practical results. It is based on the insight that

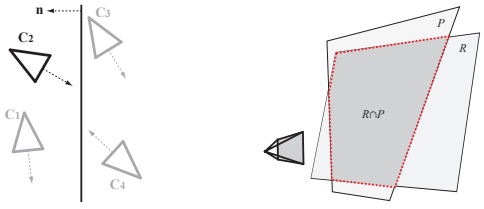

Figure 5: *Left: Example of valid ($\mathbf{C}_2$) and invalid cameras in the system. Right: the area of the intersection $R \cap P$ in determines the "quality" of the projecting camera.*

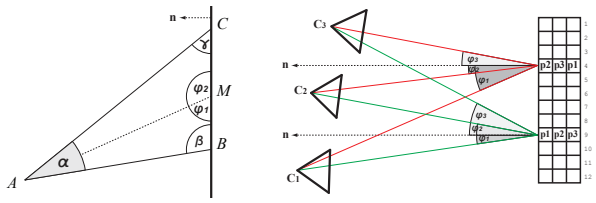

Figure 6: *Left: The relations of the projection, where the length of $\overline{BC}$ only depends on the angles α, φ_1, and the length of $\overline{AM}$. Right: Projection from the shots onto the image stack. For each pixel indicated by the numbers on the right, the best cameras are chosen, and the projected value is stored in the respective layer of the stack.*

one can stitch the derivatives of two signals instead the signals themselves. The derivative functions have the advantage that the intensity differences between them are relative, and not absolute as in the original signals. Thus, any differences in the amplitude of the original signals vanish in their gradient fields. We can compute them in the discrete case of an image I as forward differences:

$$\begin{aligned} \partial I/\partial x &= I_{(x+1,y)} - I_{(x,y)} \\ \partial I/\partial y &= I_{(x,y+1)} - I_{(x,y)} \,. \end{aligned}$$

After editing (e.g., deleting, amplifying) and combining (e.g., blending, averaging) of the derivatives of one or more images, one obtains a modified gradient field $G = [G_x\ G_y]^T$. Unfortunately, this is usually a non-integrable vector field, since its curl is not equal to zero, and thus one cannot reconstruct the original signal by a trivial summation. This problem is addressed by solving for the best approximation of the primitive (original) signal by minimizing the following sum of squared differences:

$$E_U = \|\nabla U - G\|^2 \quad \longrightarrow \min .$$

In other words, we are looking for a new image U, whose gradient field ∇U is closest to G in the least squares sense. This can be formulated as a Poisson equation:

$$\nabla^2 U = \frac{\partial G_x}{\partial x} + \frac{\partial G_y}{\partial y} ,$$

which results in a sparse system of linear equations that can be solved using least squares. Since we strive for real-time performance, we adapt a GPU solver proposed by [MP08], which is a multi-grid solution [AR07]. It performs at real-time rates with up to four mega pixel images (on an NVIDIA GeForce GTX 285), which allows not only for the stitching of precomputed layers but also interactive editing of the layers. We elaborate this in Section 2.6.

Color Correction. Despite the fact that we are using a Poisson image editing approach, we perform a simple color correction procedure before the actual stitching process. This provides better initial values and has turned out to be useful in cases where we have slight transition in the illumination of the façade. In practice this happens very often, since the global illumination (sun, clouds) changes. We resort to a simple approach presented by Reinhard *et al.* [RAGS01], where we just shift the mean μ and the standard deviation σ of all images in the stack to common values. Unlike their method, we perform the linear shift in the RGB color space, since we do not aim for an appearance change but just for slight color correction:

$$c_{out} = \frac{\sigma_{key}}{\sigma_{in}} (c_{in} - \mu_{in}) + \mu_{key} ,$$

where c stands for each color channel separately. The key-values are chosen from an input shot with the largest projected area on the bounded façade plane.

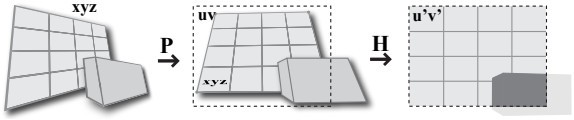

Figure 7: *Left: projection of the 3d scene by a shot-camera* $\mathbf{P}_k$*. Note the occluder in front. Middle: We compute a homography* $\mathbf{H}_k$ *of the façade-plane to the view-port. Right: in the vertex shader the scene is transformed by the shot view projection* $\mathbf{P}_k$ *and* $\mathbf{H}_k$*.*

2.5. Occlusion Handling

The described multi-view projection delivers optimal color samples for the ortho-façade pixels as long as the proxy geometry of the scene is visible from the cameras. However, in real-life data we usually encounter a number of obstacles between the camera and the façade: pedestrians, street signs, vehicles, vegetation, etc. These, if projected on the plane provide unwanted and disturbing artifacts. To counter this, we introduce two ways to integrate the occlusion into the scene.

Point-Footprint Projection. The first idea is based on the observation that many 3d points of the SfM point cloud do not belong to the proxy, but to other objects in front of the camera (see Figure 3, top, red circles). Hence, they represent potential obstacles and we splat these points onto the image-plane, such that their footprints provide an additional visibility term V_{ik} to the source-selection function presented in Equation 1:

$$E_I = \sum_i \sum_k (\lambda_1 A_{ik} + \lambda_2 B_{ik}) \cdot V_{ik} \quad \longrightarrow \max , \qquad (2)$$

In our implementation, we introduce the V_{ik} term as a per-shot mask, which contains per-pixel visibility information from the splatted 3d points (shown in Figure 8). According to this value, a shot might be considered as an occluded one, even if its score from Equation 1 is high.

Geometric Occluders. One further way to include the occluding objects into the scene is to explicitly model their geometry. We do so by allowing the user to model bigger objects roughly by primitive shapes such as cuboids. An example is shown in Figure 11, where a shop in front of the façade has been approximated by a 3d box and entirely removed. We add this information in the same manner as with the 3d points above. However, we assign the modeled occluder maximum confidence value.

Implementation. We implement the occlusion test in hardware. Let us denote the shot-camera projection by $\mathbf{P}_k$. For each shot we compute the homography $\mathbf{H}_k$ that maps the façade proxy projected by $\mathbf{P}_k$ to the target image space. In the vertex shader we transform the entire scene by $\mathbf{P_k}$ and $\mathbf{H_k}$, such that we obtain the result in the target resolution

(see Figure 7). In the pixel shader, the interpolated depth of the projection of the scene is tested with the proxy plane. In a second pass, 3d points in front of the proxy are splatted by the same mapping as above onto the target. The radius of their footprints depends on the distance to the target and is weighted using a radial falloff-kernel (see Figure 12). The results are accumulated in a per shot mask, which acts as the occlusion term V_{ik} in Equation 2.

2.6. User Interaction

Finally, our system allows the user to directly edit on the projected façade image. To accomplish this we introduce several brushing-modi which can be applied locally and precisely in order to repair small details. The brush operations exploit the fact that we have multiple information per pixel stored in the image stack. On the top of the stack (and thus visible) lies the color taken from the camera that best maximizes Equation 2. However, neither the automatic, 3d point footprint method, nor the interactive geometry modeling method presented above ensure the removal of all outliers. With the help of interactive brushing in the gradient domain, our system provides the user convenient editing tools to control the final result. The following brushes relax the results provided by Equation 2 and change the order in the stack.

Growing Brush. This brush is thought to "grow" a region projected from one image over an other region. It captures the shot where the user starts to brush (by clicking). While holding the mouse button down, the captured shot is propagated interactively to others. As a visual aid, the user can overlay the multi-view image with a colored indication layer, such that regions stemming from different sources are highlighted by different colors, as shown in Figure 9.

Eraser Brush. The idea behind this brush is to use pixel samples lying behind the visible stack layer. Each time the user clicks, the next layer is chosen and its information can be brushed on the top of the stack. If the last layer is active, it rotates on click over the stack modulo the number of layers. In this way it is possible to bring information from another cameras to the front by just clicking on one position. Since other shots have a different viewpoint, they often do not contain the potential occluder on the same pixels, but shifted due to the parallax. In other words, this brush brings the next layer information at current mouse position to the front and gives the user a simple way to switch between the layers (Figure 9).

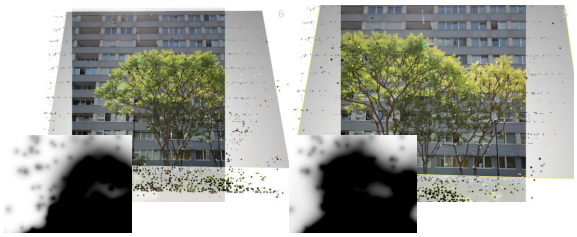

Figure 8: *Occlusion masks of two shots generated by splatting the 3d points onto the proxy plane. Shots are looking at the proxy, the overlayed masks are in proxy-plane space. The final result of this scene is shown in Figure 12.*

3. Results

The table on the right shows timings of the system with 22 input images (8 MP each) measured at two target resolutions (Intel Quad Core with NVIDIA GeForce GTX 285). Brushing runs on the same data set at approx. 40 fps. In Figures 1, 10, 11 and 12 we present visual results of our system. Additionally, we refer to the accompanying video material. We usually work with a target resolution of 2 mega pixels, mainly due to hardware limitations. However, since our system allows the user to freely define the extent of the projected façade, it is easily possible to focus only on selected parts and apply the maximum resolution to this subregions only. This "zoom" is of course limited by the source resolution, which can have up to 16 mega pixels on current hardware with DX9.

operation	2 MP	3 MP
accumul.	0.05s	0.06s
color corr.	6.0s	8.0s
sampling	9.0s	11.5s

Limitations. Our method fails in cases, where in all input images the actual façade is occluded. In such cases we want to resort to methods that utilize similarity present in the image. A problem of our current implementation is the limitation of the stack to four layers due to hardware-API constraints (DX9). We plan to switch to DX10 to resolve this. Finally, our method is quite hardware intensive, such that it requires graphics cards with 1GB video RAM to perform well.

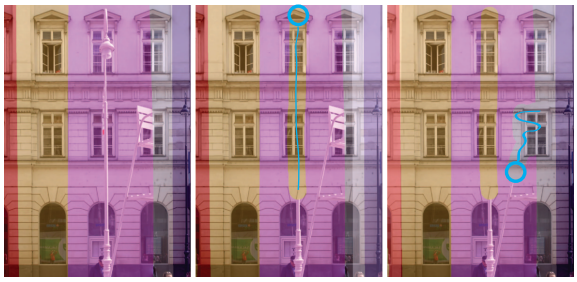

Figure 9: *Interactive brushing. Left: color stripes indicate regions stemming from different cameras. Middle: the eraser brush brings the yellow layer to the front (over the purple). Right: the growing brush pulls the gray layer over the purple one. Blue storks indicate the user actions.*

4. Conclusions

We present a system for generating approximately orthographic façade textures. We pay particular attention to high-quality, high-resolution and obstacle-free images. Most steps of our method are fully automatic: image registration, pose estimation, plane fitting as well as per-pixel projection. On the other hand, some tasks have proven difficult to solve automatically with adequate quality. For these cases we introduce interactive tools. For the problem of bounding the actual façade, we provide the user with an easy method to define the extent. Another difficult problem is the detection and removal of possible occluders in front of the façades. To solve this, we propose two approaches: projection of SfM outliers and modeling of additional geometry. The major contribution of our system is the detailed removal of occluders by exploiting the multi-view information. For the future, we are considering to extend the system in a way that allows the user to operate in moderate resolutions for real-time interaction while calculating higher resolutions offline. Furthermore, we want to extend the geometry modeling part of the solution. Our system is intended to serve as part of a complex urban reconstruction pipeline.

Acknowledgments. We would like to thank the Aardvark-Team, especially Robert F. Tobler and Mike Hornacek. Further, we thank Stefan Jeschke for providing his source-code.

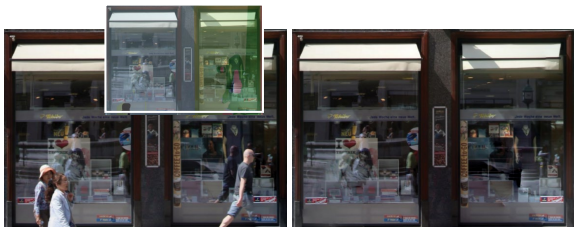

Figure 10: *A close-up of the image shown in Figure 11. Pedestrians and their reflections visible in the left image have been removed (middle).*

References

[AAC*06] AGARWALA A., AGRAWALA M., COHEN M., SALESIN D., SZELISKI R.: Photographing long scenes with multi-viewpoint panoramas. *ACM Transactions on Graphics 25*, 3 (July 2006), 853. 2

[ADA*04] AGARWALA A., DONTCHEVA M., AGRAWALA M., DRUCKER S., COLBURN A., CURLESS B., SALESIN D., COHEN M.: Interactive digital photomontage. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 294. 2

[Aga07] AGARWALA A.: Efficient gradient-domain compositing using quadtrees. *ACM Transactions on Graphics 26*, 3 (July 2007), 94. 2

[AR07] AGRAWAL A., RASKAR R.: Gradient domain manipulation techniques in vision and graphics. ICCV 2007 Course (http://www.umiacs.umd.edu/~aagrawal/ICCV2007Course/index.html), 2007. 2, 5

[BA83] BURT P. J., ADELSON E. H.: A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics 2*, 4 (Oct. 1983), 217–236. 2

[CT99] COORG S., TELLER S.: Extracting textured vertical facades from controlled close-range imagery. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)* (1999), IEEE Comput. Soc, pp. 625–632. 2

[DTM96] DEBEVEC P. E., TAYLOR C. J., MALIK J.: Modeling and rendering architecture from photographs. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96* (New York, New York, USA, 1996), ACM Press, pp. 11–20. 2

[DYB98] DEBEVEC P. E., YU Y., BORSHUKOV G.: Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering techniques' 98: proceedings of the Eurographics Workshop in Vienna, Austria, June 29-July 1, 1998* (1998), Springer Verlag Wien, p. 105. 2

[EDM*08] EISEMANN M., DE DECKER B., MAGNOR M. A., BEKAERT P., DE AGUIAR E., AHMED N., THEOBALT C., SELLENT A.: Floating Textures. *Computer Graphics Forum 27*, 2 (Apr. 2008), 409–418. 2

[ELS08] EISENACHER C., LEFEBVRE S., STAMMINGER M.: Texture Synthesis From Photographs. *Computer Graphics Forum 27*, 2 (Apr. 2008), 419–428. 2

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (June 1981), 381–395. 3

[GH97] GUPTA R., HARTLEY R.: Linear pushbroom cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 9 (1997), 963–975. 2

[GWOH10] GAL R., WEXLER Y., OFEK E., HOPPE H.: Seamless Montage for Texturing Models. *Computer Graphics Forum 29*, 2 (2010), to appear. 2

[IZB07] IRSCHARA A., ZACH C., BISCHOF H.: Towards Wiki-based Dense City Modeling. In *2007 IEEE 11th International Conference on Computer Vision* (Oct. 2007), IEEE, pp. 1–8. 3

[JCW09] JESCHKE S., CLINE D., WONKA P.: A GPU Laplacian solver for diffusion curves and Poisson image editing. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 1. 2

[LCZ99] LIEBOWITZ D., CRIMINISI A., ZISSERMAN A.: Creating Architectural Models from Images. *Computer Graphics Forum 18*, 3 (Sept. 1999), 39–50. 2

[Low04] LOWE D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision 60*, 2 (Nov. 2004), 91–110. 3

[MP08] MCCANN J., POLLARD N. S.: Real-time gradient-domain painting. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 1. 2, 5

[MRM*10] MUSIALSKI P., RECHEIS M., MAIERHOFER S., WONKA P., PURGATHOFER W.: Tiling of Ortho-Rectified Façade Images. In *Spring Conference on Computer Graphics (SCCG'10)* (Budmerice, 2010). 1

[MWR*09] MUSIALSKI P., WONKA P., RECHEIS M., MAIERHOFER S., PURGATHOFER W.: Symmetry-Based Façade Repair. In *Vision, Modeling, Visualisation (VMV'09)* (2009), Magnor M. A., Rosenhahn B., Theisel H., (Eds.), DNB, pp. 3–10. 2

[MZWvG07] MÜLLER P., ZENG G., WONKA P., VAN GOOL L.: Image-based procedural modeling of facades. *ACM Transactions on Graphics 26*, 3 (July 2007), 85. 1

[NK01] NEUGEBAUER P. J., KLEIN K.: Texturing 3D Models of Real World Objects from Multiple Unregistered Photographic Views. *Computer Graphics Forum 18*, 3 (Sept. 2001), 245–256. 2

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Transactions on Graphics 22*, 3 (July 2003), 313. 2, 4

[PSK06] PAVIĆ D., SCHÖNEFELD V., KOBBELT L.: Interactive image completion with perspective correction. *The Visual Computer 22*, 9-11 (Aug. 2006), 671–681. 2

[PvGV*04] POLLEFEYS M., VAN GOOL L., VERGAUWEN M., VERBIEST F., CORNELIS K., TOPS J., KOCH R.: Visual Modeling with a Hand-Held Camera. *International Journal of Computer Vision 59*, 3 (Sept. 2004), 207–232. 3

Figure 11: *Top left: multi-view stitching without constraints. Top right: multi-view stitching with geometry constraints. Bottom from left to right: one of the original perspective shots, occluding geometry has been modeled into the scene, source-indication masks without and including the geometry occlusion.*

Figure 12: *Automatic removal of occluding objects by utilizing the information from structure-from-motion points. Left: image and its mask after multi-view stitching without the occlusion term. Middle: results with occlusion term. Right: result with occlusion term post-processed by interactive brushing. Note that lens flares have been removed as well.*

[RAGS01] REINHARD E., ADHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Computer Graphics and Applications 21*, 4 (2001), 34–41. 5

[RGL04] ROMAN A., GARG G., LEVOY M.: Interactive design of multi-perspective images for visualizing urban landscapes. *IEEE Visualization 2004* (2004), 537–544. 2

[SGSS08] SNAVELY N., GARG R., SEITZ S. M., SZELISKI R.: Finding paths through the world's photos. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 1. 2

[SK03] SEITZ S. M., KIM J.: Multiperspective imaging. *IEEE Computer Graphics and Applications 23*, 6 (Nov. 2003), 16–19. 2

[SSS07] SNAVELY N., SEITZ S. M., SZELISKI R.: Modeling the World from Internet Photo Collections. *International Journal of Computer Vision 80*, 2 (Dec. 2007), 189–210. 2

[SSS*08] SINHA S. N., STEEDLY D., SZELISKI R., AGRAWALA M., POLLEFEYS M.: Interactive 3D architectural modeling from unordered photo collections. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 1. 2

[TS08] THORMÄHLEN T., SEIDEL H. P.: 3D-modeling by ortho-image generation from image sequences. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 1. 2

[vdHDT*07] VAN DEN HENGEL A., DICK A., THORMÄHLEN T., WARD B., TORR P. H. S.: VideoTrace: rapid interactive scene modelling from video. *ACM Transactions on Graphics 26*, 3 (July 2007), 86. 2

[XFT*08] XIAO J., FANG T., TAN P., ZHAO P., OFEK E., QUAN L.: Image-based façade modeling. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 1. 2

[XFZ*09] XIAO J., FANG T., ZHAO P., LHUILLIER M., QUAN L.: Image-based street-side city modeling. *ACM Transactions on Graphics (TOG) 28*, 5 (2009). 2

[ZFPW03] ZOMET A., FELDMAN D., PELEG S., WEINSHALL D.: Mosaicing new views: the crossed-slits projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 6 (June 2003), 741–754. 2

[Zhe03] ZHENG J. Y.: Digital route panoramas. *IEEE Multimedia 10*, 3 (July 2003), 57–67. 2