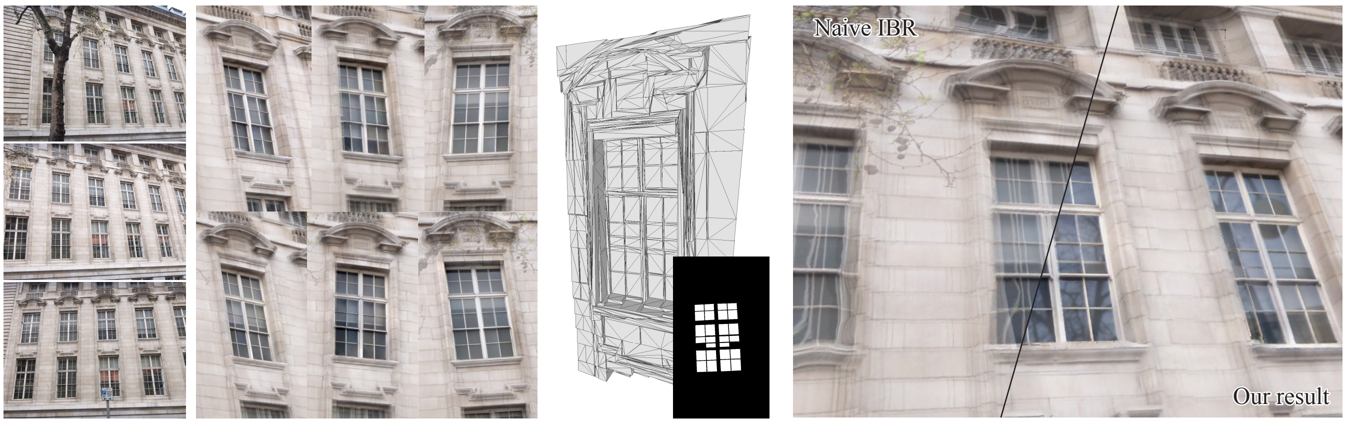
Eurographics Symposium on Rendering 2018
T. Hachisuka and W. Jakob
(Guest Editors)

Volume 37 (2018), Number 4

Exploiting Repetitions for Image-Based Rendering of Facades

Simon Rodriguez[1] and Adrien Bousseau[1] and Fredo Durand[2,1] and George Drettakis[1]

[1]Inria, Université Côte d'Azur
[2]MIT CSAIL



(a) Input (b) Repetitive Window (c) 3D Mesh and Reflection Mask (d) Image-Based Rendering

Figure 1: *Given a small number of pictures of a facade (a), we augment the number of views of the repetitive windows by combining the different views of each instance (b). We place the augmented set of viewpoints into a common space, allowing us to generate finer 3D geometry and to identify reflective areas (c). Compared to an image-based rendering of the input views, our solution produces sharper results and fewer popping artifacts (best seen in the accompanying video).*

Abstract

Street-level imagery is now abundant but does not have sufficient capture density to be usable for Image-Based Rendering (IBR) of facades. We present a method that exploits repetitive elements in facades – such as windows – to perform data augmentation, in turn improving camera calibration, reconstructed geometry and overall rendering quality for IBR. The main intuition behind our approach is that a few views of several instances of an element provide similar information to many views of a single instance of that element. We first select similar instances of an element from 3-4 views of a facade and transform them into a common coordinate system, creating a "platonic" element. We use this common space to refine the camera calibration of each view of each instance and to reconstruct a 3D mesh of the element with multi-view stereo, that we regularize to obtain a piecewise-planar mesh aligned with dominant image contours. Observing the same element under multiple views also allows us to identify reflective areas – such as glass panels – which we use at rendering time to generate plausible reflections using an environment map. Our detailed 3D mesh, augmented set of views, and reflection mask enable image-based rendering of much higher quality than results obtained using the input images directly.

CCS Concepts

•***Computing methodologies*** $\rightarrow$ ***Computer graphics; Image-based rendering; Reconstruction;***

1. Introduction

Recent Image-Based Rendering (IBR) algorithms [CDSHD13, OCDD15, HRDB16, PZ17] allow high-quality free-viewpoint navigation of cityscapes, but require relatively high capture density – typically a high-resolution image every meter or so. City-wide street level captures are now widely available (e.g., Google

Streetview, Microsoft Bing StreetSide), but are captured much more sparsely. Such low densities hinder all components of IBR: camera calibration, geometric reconstruction, and image interpolation for rendering. In this paper we develop solutions for IBR from sparse street-level capture by leveraging the repetitive nature of facades. The key idea in our work is to extract an idealized or *platonic* object corresponding to a given repetitive element, and use its multiple instances present in the facade to perform *data augmentation* allowing us to perform high-quality free-viewpoint IBR from sparse input. In particular, we focus on repetitive windows, which exhibit rich geometric and photometric details.

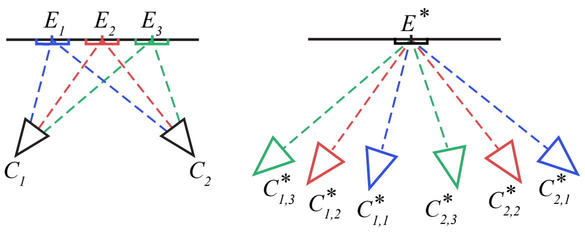


Figure 2: *A small number of views of similar – but physically distinct – elements E_i can be seen as a larger number of views of a unique element E^* if we align them into a common coordinate system.*

We start with a small number of views of a given facade (3-4 street-level pictures), and approximate cameras calibrated using Structure-from-Motion (SfM). We then semi-automatically extract cropped instances of the repetitive windows, which we call *subviews*. Our idea is to consider that all these subviews represent the same platonic window, as illustrated in Fig. 2. We use 3D information in all subviews during Structure-from-Motion to provide finer camera calibration in the common space, while running multi-view stereo in this space gives us a dense, albeit noisy 3D mesh of the platonic window. We refine this mesh by decomposing it into planar polygonal regions aligned with image edges, which typically correspond to the window panels, frames and surrounding bricks.

While the multiple subviews provided by similar instances improve 3D reconstruction, they include color variations that are view-dependent (such as reflections on glass panels) and instance dependent (such as different color or shape of blinds), which produce severe popping artifacts if used directly in IBR. We extract view-dependent variations by analyzing each instance separately, effectively removing reflections from the pictures. Aggregating the resulting reflection layers over all instances gives us a unique reflection mask for the platonic element, which we use to composite environment reflections during rendering. We treat instance-dependent variations by re-projecting each subview of an instance into the subviews of all other instances, only mixing information between different instances in the presence of occlusions.

In summary, we introduce the idea of exploiting repetitions to augment visual data in the context of image-based rendering of facades, and show how this augmentation improves camera calibration, 3D reconstruction, and reflection segmentation. These elements combined allow us to greatly improve visual quality for IBR of facades captured with a small number of pictures.

2. Related work

Our work targets Image-Based Rendering and the use of repetitions, which are both extensive research areas. We review here closely related work; more complete reviews can be found in surveys or books [LHOK*10, SCK08].

Image-based rendering. Image-Based Rendering (IBR) algorithms synthesize novel views by interpolating between photographs of the same scene. The Lumigraph [GGSC96] and Lightfield [LH96] methods generate novel views by interpolating light rays in a 4D light volume, but require the scene to be captured at very high density.

When the input images are sparser, additional information such as a 3D mesh of the scene can improve image interpolation [BBM*01]. However, the baseline of the input images still needs to be small enough to obtain good angular resolution, and avoid ghosting and other rendering artifacts. Wide baselines also hinder automatic 3D reconstruction of the mesh, which is particularly problematic along object silhouettes where foreground and background should not be mixed. Chaurasia et al. [CDSHD13] and Hedman et al. [HRDB16] address this latter challenge by respectively segmenting the image along strong contours or by aligning the 3D mesh to each input image independently. In contrast, we exploit repetitions present in facades to artificially augment the number of input views, which benefits both the 3D reconstruction and image interpolation steps of Image-Based Rendering.

While dense IBR representations like Lightfields naturally handle view-dependent effects, wide-baseline methods based on 3D meshes often produce strong popping in the presence of reflections. Proper handling of such effects requires a multi-layer representation where the reflected layer lies at a different depth than the reflective surface [SKG*12, KLS*13]. While we extract reflection layers from our input images, we cannot recover the depth of the reflected objects because they are rarely visible in multiple images due to our wide baseline. Instead, we combine the information provided by the reflection layers of all instances of a repetitive window to estimate a reflection mask for that window, which we use to render plausible reflections using image-based lighting.

Repetitions. The presence of repetitions in images has been exploited for numerous applications in Computer Graphics. In particular, several methods build on the idea that different instances of a repetitive element can be seen as multiple views of the same platonic element. For example, Xu et al. [XWL*08] take a single picture of a flock of animals to generate an animation of that animal, effectively turning spatial repetition into temporal information. Aittala et al. [AWL*15] use repetitions in a flash picture of a material sample to extract patches of the material under different lighting conditions, recovering rich angular information for SVBRDF acquisition. Dekel et al. [DMIF15] extract local differences between similar patches to attenuate or accentuate small variations in an image. Closer to our context is the work of Alhalawani et al. [AYLM13], who rely on user interaction to detect window-specific details, such as blinds and shutters. Since these details

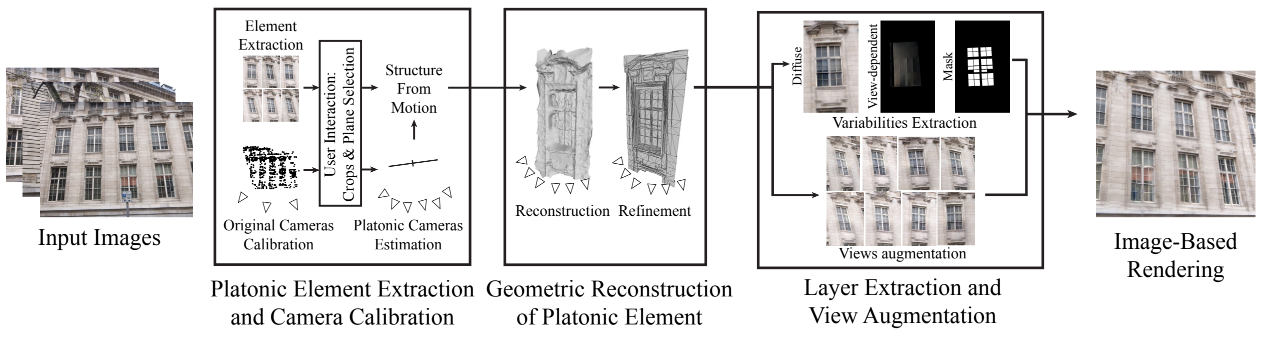

Input Images
Element Extraction
Original Cameras Calibration
User Interaction: Crops & Plane Selection
Structure From Motion
Platonic Cameras Estimation
Platonic Element Extraction and Camera Calibration
Reconstruction
Refinement
Geometric Reconstruction of Platonic Element
Diffuse
View-dependent
Mask
Variabilities Extraction
Views augmentation
Layer Extraction and View Augmentation
Image-Based Rendering

Figure 3: *Overview. From the input images, we first segment and select repetitive windows, followed by camera calibration of these subviews by a modified SfM algorithm. In the second step, we use the calibrated subviews to reconstruct a* platonic mesh *of the window using multi-view stereo, and subsequently refine it to obtain a piecewise planar geometry. In the third step we extract reflections from each subview and deduce a reflection mask of the window. We additionally generate images of all instances of the window from all available viewpoints. We finally re-insert the platonic geometry and images of each instance into the complete facade for improved image-based rendering.*

are present in different states in the various instances, they can be sorted and animated. We follow a similar strategy as the above approaches by turning spatial repetition into angular information, although we target the different application domain of image-based rendering.

Our approach is also inspired by 3D reconstruction methods for urban scenes. In particular, we build on the work by Wu et al. [WFP11], who detects repetitive facade elements inside a single image to estimate a depth map, and on the work by Xia et al. [XFT*08], who regularizes noisy depth maps by decomposing facades into rectangular regions with constant depth. We combine and extend these two ideas to reconstruct a detailed 3D mesh from multiple images of a repetitive element. Our use of repetitions for 3D reconstruction is also related to the work of Zheng et al. [ZSW*10] and Demir et al. [DAB15], who aggregate information from repetitive pieces of a 3D point clouds to consolidate it. Heinly et al. [HDF14] also use repetitions to improve camera calibration by detecting conflicting observations between viewpoints.

Detecting repetitions can also be a step towards inference of procedural rules from an existing building. Some methods extract repetitions and variations from a single fronto-parallel image of a facade, as a set of tiles and rules [MRM*10,DRSVG13]. New facade images can then be generated by following these procedural rules and mixing the different tiles. Starting from a single planar facade image, Muller et al. [MZWVG07] factorize irreducible architecural elements. These tiles are matched to a bank of 3D models (windows, doors, etc) that are inserted in the facade plane and textured. The goal of our method is similar, but does not rely on external 3D models. Given a set of input images and 3D model of a building, Aliaga et al. [ARB07] propose a user-assisted method to construct a procedural model of the building. The procedural grammar enables semantic edits, such as adding new floors, while the input images enable view-dependent texture mapping of each facade element. Jiang et al. [JTC09] attain similar results using a unique input image and user-drawn strokes, estimating the camera pose using the scene symmetries. Zhao et al. [ZYZQ12] also exploit repetitions to segment window elements on a facade. Jiang et al. [JTC11] extract lattice structures from repetitive facades and improve an image-based modeling process. In contrast, we focus on the components required for high-quality image based rendering of facade elements – camera calibration, geometric reconstruction, and image interpolation – rather than their use within a procedural modeling context.

3. Overview

Figure 3 illustrates the main steps of our approach. Our input is a small set of images (3-4) of a building, taken at street level alongside the facade. The pictures are acquired with a consumer camera, using fixed exposure.

We first automatically detect windows in the facade using a deep classification network, and ask the user to select the windows that are visually similar. We crop the input images around each such instance to obtain a set of images that all represent the same *platonic* window, under different viewpoints. We will refer to these crops as *subviews*.

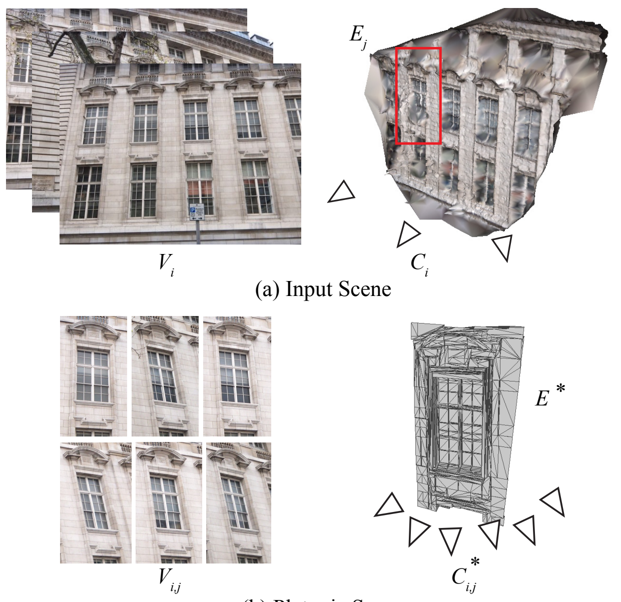
Due to the wide baseline of the input images, existing structure-from-motion methods only produce an approximate calibration of the input cameras. We use this information to compute an initial guess of the camera of each subview in a common platonic space, which we update by running a structure-from-motion algorithm on all subviews. The initial guess allows us to reject erroneous correspondences that may occur when a subview not only shows its central window, but also part of its neighboring windows. This filtering process allows us to improve "platonic" camera calibration for each of the subviews.

After platonic camera calibration, we run a multiview-stereo algorithm to reconstruct a 3D mesh of the platonic element. All subviews represent a similar window, but they often contain instance-specific details that disturb generic reconstruction algorithms, resulting in noisy surfaces. We regularize this geometry to obtain a piecewise planar mesh composed of polygonal regions aligned with the dominant lines of the window.

The 3D mesh we obtain allows us to re-project any subview into any other one. We use this feature to extract view-dependent variations between subviews of the same instance, and to generate images of each instance as seen from the cameras of all subviews. While the reflections extracted from a single instance may be sparse, combining the information provided by all instances allows us to estimate a *reflection mask* that indicates the reflective areas of the platonic window.

Given these preprocessing steps, we create a complete scene representation by inserting the 3D mesh of the platonic window at the location of each of its instances over the facade mesh. We composite each view of a given instance to be consistent with the corresponding view of the entire facade, generating an enriched set of subviews by combining views and instances via reprojection. We use the resulting mesh and images in an image-based rendering algorithm, which we enhance with image-based reflections using our reflection mask. An additional use of our method output data is the generation of a multi-view textured mesh.

Terminology. In what follows, the term *input scene* refers to the original 3-4 photos of the entire facade, along with calibrated cameras and a coarse 3D mesh computed using multiview stereo. The *platonic scene* refers to the scene containing the calibrated cameras and reconstructed geometry of the platonic element, computed from multiple subviews of that element. All elements of the platonic scene are denoted with a superscript $*$. We denote the initial images as V_i, and the input *physical instances* of the repetitive element as E_j. The cropped *subviews* $V_{i,j}$ use a double-index notation, indicating the input view i they were extracted from, and the physical instance j they contain. The associated cameras in the common platonic space are denoted $C^*_{i,j}$ and the platonic model itself E^*. We illustrate these quantities in Fig. 4.



(b) Platonic Scene

Figure 4: *From given input views V_i of a facade and their associated cameras C_i (a), we extract subviews $V_{i,j}$ of similar windows E_j. From these, we can reconstruct a 3D model of the* platonic element E^* *along with* platonic cameras $C^*_{i,j}$ *(b) in a common space.*

4. Windows Extraction and Platonic Camera Calibration

The first step of our method consists in cropping repetitive windows in the facade to form a multi-view dataset of the corresponding platonic window. Calibrating the cameras of each crop within a common space then enables 3D reconstruction of the window with higher accuracy than using solely the input images.

4.1. Window extraction

We identify the repetitive windows in a semi-automatic manner, where we automatically detect candidate windows and let the user select the ones that should be considered similar (Fig. 5).

We generate the candidate windows by running the rectified images through a deep classification network. We use a U-Net architecture [RFB15], trained on the CMP Facade Database [RT13], containing 600 rectified images of facades with ground truth labels of architectural elements. We only predict two labels – windows and background. We obtain the rectified facade using the vanishing point method by Wu et al. [WFP10] We then process the classification to extract its connected components, which should correspond to individual windows. We compute the bounding box of each component and scale it by a factor of 1.75 to include its surrounding, and re-project these boxes into the input image to be shown to the user.

This user interaction is very easy, and only takes a few seconds per dataset (please see video). Automating this process should be possible but would require similarity metrics robust to differences between instances of the same window while being sensitive to differences between different windows, e.g., top parts of the first and second row of windows in Fig. 5. We leave the exploration of robust similarity detection, for instance based on deep features, to future work.

4.2. Platonic Camera Calibration

We now have a list of cropped subviews, each representing an instance of the platonic window seen from a different viewpoint. However, our automatic crops sometimes contain parts of neighboring windows on their sides, as shown in Fig. 6(a). These duplicates challenge structure-from-motion algorithms since points on the central window in one subview may be matched to points in the side window in another subview.

We filter out such erroneous matches by deriving an approximate camera for each subview, which we compute from the known position and size of the corresponding crop in the input image, as well as from the camera pose of that image. This computation also requires knowledge of the facade geometry in order to position the approximate cameras such that they all point to the same element in platonic space. We approximate this geometry as a plane, fit to the point cloud of the facade, computed by running multiview-stereo reconstruction on the input images (Fig. 7). In most cases, a RANSAC fit is sufficient; when this fails we ask the user to specify the plane by selecting three points on the point cloud. Given

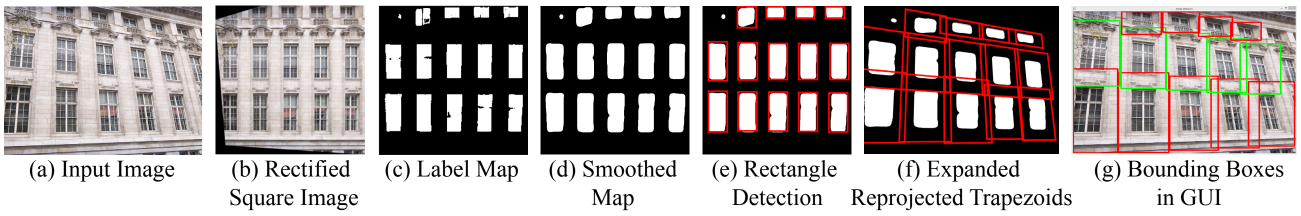

(a) Input Image
(b) Rectified Square Image
(c) Label Map
(d) Smoothed Map
(e) Rectangle Detection
(f) Expanded Reprojected Trapezoids
(g) Bounding Boxes in GUI

Figure 5: *We rectify each input image (a,b) before feeding it to a deep classification network that detects window pixels (c). We smooth the label map (d) and compute the bounding box of each connected component (e). We then expand these boxes to include the surroundings of each window and project them back into the image (f). We ask users to select the boxes corresponding to a group of similar windows (g).*

these cameras, we modify an SfM algorithm [MMMO] to reject matches for which a point, when reprojected into the other subview, lands further away than half the image dimension from its correspondence. This filtering operation greatly improves the quality of platonic camera calibration, as shown in Fig. 8.

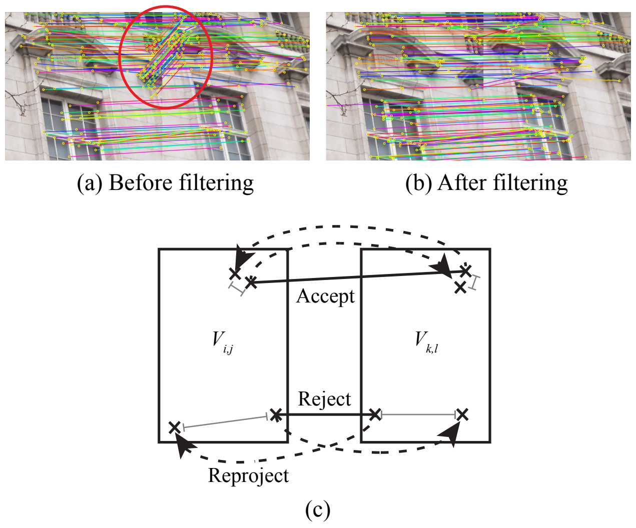


Figure 6: *(a) Candidate matches between two views using standard approaches. The two views represent a similar window: the first view contains parts of another window on its side, which are matched to the central window of the second view. (b) We use an approximate camera derived from the input scene to filter these outliers. (c) We reproject each pair of matched points into each other subview using the approximate cameras. If both reprojections land close to their correspondences, the match is kept.*

5. Geometry Reconstruction

We first describe how to reconstruct a 3D mesh of the platonic window, before explaining how we repeat it over the entire facade.

5.1. Platonic Window

The platonic cameras estimated by our modified SfM algorithm can now be used for dense 3D reconstruction of the platonic window. However, we found that generic multi-view stereo algorithms

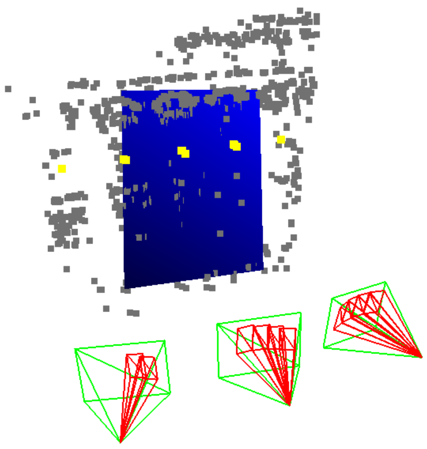


Figure 7: *The estimated facade plane is displayed (blue) along with the input scene sparse point cloud. Input and crop cameras are shown. The yellow points are the intersection of each crop subview with the plane.*

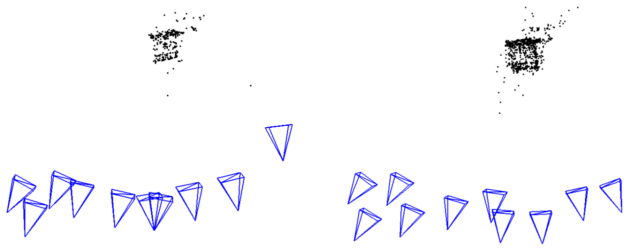


Figure 8: *Camera calibration using standard approach (left), and the result with our approach (right). Note how in the standard approach, the rightmost camera does not see the window, and that three cameras are incorrectly co-located.*

[Rea18] tend to produce noisy 3D point clouds on such input, possibly due to the per-instance and per-view variations present in the cropped subviews. Inspired by prior work on image-based facade modeling [XFT*08], we refine this noisy reconstruction by decomposing it into flat polygonal regions parallel to the facade, and aligned with image edges.

Our refinement method operates in the subview for which the view direction is the most orthogonal to the facade plane. We segment this subview into convex polygons aligned with image edges using the recent algorithm of Bauchet and Lafarge [BL18], as shown in Fig. 9. In a nutshell, this algorithm detects small line segments in the image and extends them until they intersect other segments to form closed regions.

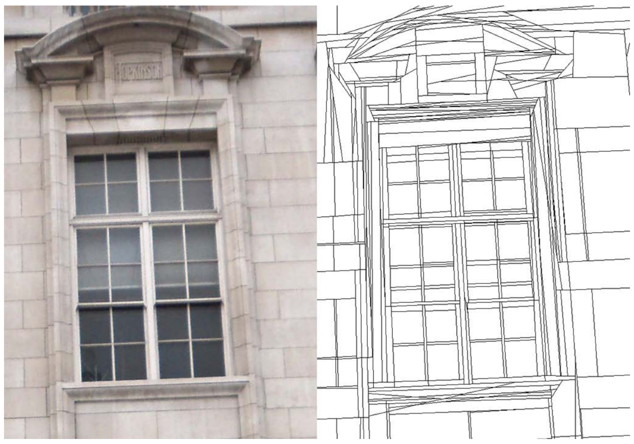


Figure 9: *Left: Input image. Right: Polygonal segmentation.*

We then formulate an optimization that displaces each region along the facade normal. Our formulation combines a data term, which seeks to keep the region close to the noisy point cloud it covers, and a smoothness term, which encourages neighboring regions to align if they share similar colors. Denoting r a polygonal region, d_r its displacement along the facade normal ($d_r = 0$ at the facade plane), d_{ref} the average distance to the facade plane of the reconstructed points covered by r, $\mathcal{N}(r)$ the two-ring neighborhood of r, and c_r the average color of r, we want to minimize

$$E(d_r) = |d_r - d_{ref}| + \lambda \frac{1}{W} \sum_{n \in \mathcal{N}(r)} e^{-\|c_r - c_n\|_2} |d_r - d_n| \qquad (1)$$

where $W = \sum_{n \in N(r)} e^{-\|c_r - c_n\|_2}$ is a normalization factor and λ balances the two terms. We set $\lambda = 15$ in all our experiments.

Given the small number of regions, we solve this optimization problem using a simple mean field algorithm [ZC93], where we iteratively update the displacement of each region to minimize Eq. 1 given the displacement of its neighbors. At each iteration, we try to assign to a given region all displacements of its neighbors, as well as a regular sampling of the depth interval of the noisy reconstruction with respect to the facade plane. We initialize this optimization by setting $d_r = d_{ref}$. At the end of the optimization, we connect the displaced polygons by additional faces to handle depth discontinuities between neighbors. Figure 10 illustrates the result of this optimization on two 3D meshes, the first obtained by running multi-view stereo on the input images, and the second obtained from all subviews. The best result is achieved with the latter option, where our refinement removes bumps and captures well the flat parts of the window and wall.

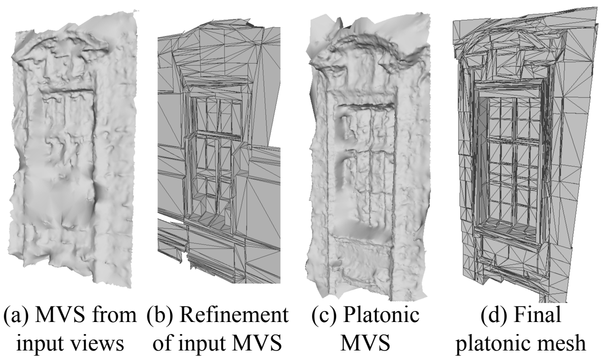


Figure 10: *From left to right: noisy surface obtained by applying multi-view stereo on the input images, refinement of this noisy surface, noisy surface obtained by applying multi-view stereo on the cropped subviews, refinement of this noisy surface. The cropped subviews provide additional information that yield a more detailed mesh after refinement.*

5.2. Complete Facade

Given the reconstructed geometry of the platonic window, we generate a mesh for the entire facade by replicating the platonic mesh on the facade plane at the positions of all window instances. However, since the facade plane and the platonic mesh have been generated by separate executions of the structure-from-motion algorithm, they are not in the same coordinate system and do not have the same scale. We deduce the 3D rigid transformations from the platonic scene back to the input scene by matching the platonic cameras with the cropped cameras previously estimated in Sec. 4.2, as illustrated in Fig. 11. For each match we compute a candidate transformation and make it independent from the instance position. We average all match transformations to obtain the final one. We provide the detailed computation of this transformation in Appendix A.

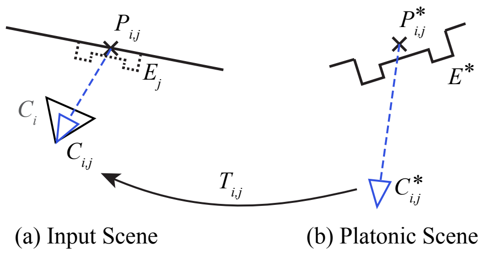


Figure 11: *For each platonic camera, we estimate the transformation to align it back to the corresponding estimated camera in the input scene. We use this transformation to copy the platonic mesh at each window occurrence in the facade.*

While the above process results in an improved 3D mesh of the facade (see Fig.12), it is not perfectly aligned with the input images due to the numerous geometric transformations involved and to the approximate calibration of the input cameras. Such misalignements produce significant ghosting if the original images are

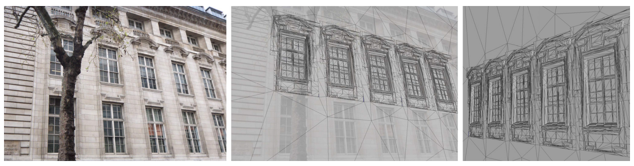


Figure 12: *Visualization of the generated facade mesh (center) from the camera of an input view (left). We overlay the input image onto the mesh view to assess alignment. Another viewpoint on the same mesh (right).*

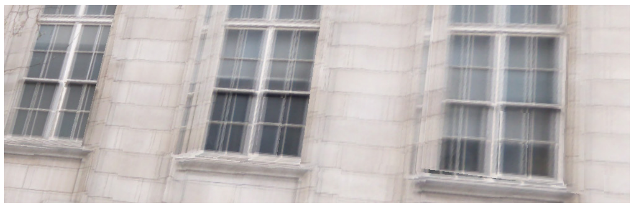


Figure 13: *Mis-alignments introduce ghosting when performing image-based rendering using the original images and the refined mesh.*

used along the refined mesh in an image-based rendering algorithm, as shown in Fig. 13. We next describe how to improve rendering quality by leveraging the more precisely calibrated subviews of the windows. We use these subview cameras, placed back into the input scene, along with the approximate input cameras for the final rendering. The subviews also provide complementary information gathered from different instances of the window, allowing us to better handle reflections than when using only the input images.

6. Data Factorization and Augmentation

While the cropped subviews all represent the same platonic element, they each contain view-dependent and instance-dependent variations which would yield significant popping artifacts if used directly for image-based rendering. We treat these two sources of variations separately, as view-dependent variations mainly correspond to reflections over the window panels, while instance-dependent variations correspond to changes of shape or color of the window frame.

6.1. View-Dependent Variations

Each instance of a window is seen in several of our input images. While window frames are mostly diffuse, the panels are typically reflective and exhibit strong variations between these input views. However, the input camera baseline is often too wide to observe any overlap in reflections, which prevents them to be rendered with existing solutions based on 3D reconstruction of the reflected scene [KLS*13]. Instead, we propose to remove these view-dependent variations to later replace them by plausible reflections from an environment map.

Reflection Separation. Given all the subviews $V_{\cdot,j}$ of a given instance, we remove the reflections in each subview by reprojecting all other subviews onto it, and computing the median gradient of the resulting image stack. We use the 3D mesh computed in Sec. 5 to perform this reprojection. As observed by Weiss in the context of shadow removal [Wei01], the median gradient preserves the visual content shared by the aligned images, while it discards content that only appears in one of the images. Integrating the resulting gradient fields gives images where most reflections have been removed, as shown in Fig. 14. We refer to these images as *diffuse layers*, and to the removed information as *view-dependent layers*.

In practice, we only perform reflection separation for pixels identified as windows by the deep classification network (Sec. 4.1), since those are the most likely to be reflective.

We noticed that median filtering is also effective at removing occluders present in one of the input images (trees, sign posts). Assuming that these occluders have a very different color than the scene they occlude, we also run the above process on pixels that exhibit a high variance in hue over the images.

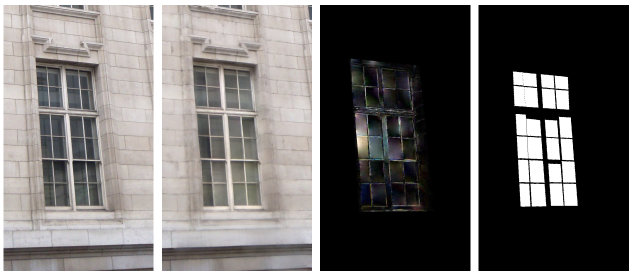


Figure 14: *From left to right: input cropped image, extracted diffuse layer, extracted variation layer (intensity), reprojected reflection mask.*

Reflection Mask. We are now equipped with one view-dependent layer per subview, which gives us indications of which parts of the platonic window are reflective. We aggregate this information across subviews to generate a unique *reflection mask*, which we use at rendering time to composite the environment map reflections over the window. Since the extracted reflections often only cover part of the window panels, we again leverage the polygonal segmentation of the cropped subviews to obtain clean, regular masks.

We first reproject all view-dependent layers into the subview from which the polygonal segmentation has been computed (Sec. 5). We then count the number of pixels in each region for which the reprojected layers have non-zero intensities, and we compute the sum of all such pixel intensities. We estimate the average value and variance of these quantities over all regions. We consider that a region is in the reflection mask if both values differ by more than 0.25 standard deviation from their average over all regions. Intuitively, this approach selects regions that are well covered by intense view-dependent effects. The resulting reflection mask (shown in Fig. 14) is then reprojected to all subviews. The entire view-dependent variations extraction process is illustrated in Fig. 15.

Reflected Environment. The reflections present in the input images are too sparse to be used directly for rendering, but they provide some information about the environment surrounding the facade, see Fig. 16. In particular, we use the extracted reflections to

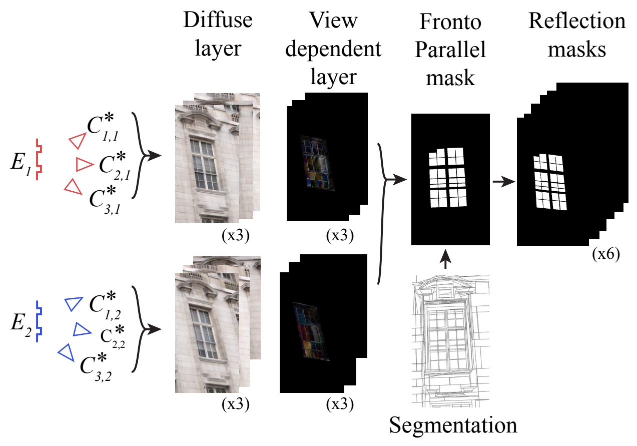


Figure 15: *For each instance, we decompose the associated cropped views into diffuse and view-dependent effects layers. All view-dependent layers are then combined to estimate a fronto-parallel mask guided by the segmentation. The resulting mask is then reprojected in all views.*

construct an incomplete environment map, which we use to manually select a similar environment map from a lightprobe library [XEOT12].

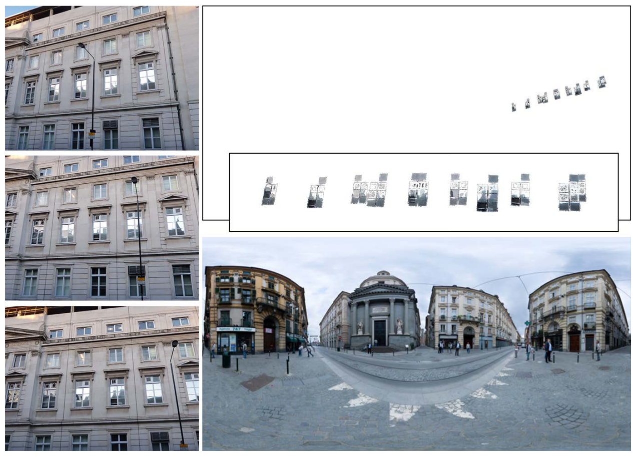


Figure 16: *Input images (left) and the resulting environment map after reprojection of the extracted reflections (top right, zoom as inset). We use this incomplete environment map to select a similar one from a lightprobe dataset (bottom right).*

6.2. Instance-Dependent Variations

The key idea behind our approach is to consider that a few views of different instances of a window can be seen as multiple views of a single platonic window. While we have shown the potential of this idea to generate a precise platonic mesh and reflection mask, using all subviews $V_{i,j}$ to render a single window yields severe popping of visual content specific to each instance. For instance, the window blinds may change height or color over different instances, producing distracting animations as we rotate around the window. Each instance thus only has a subset of coherent subviews, which correspond to the crops of that instance in the original images.

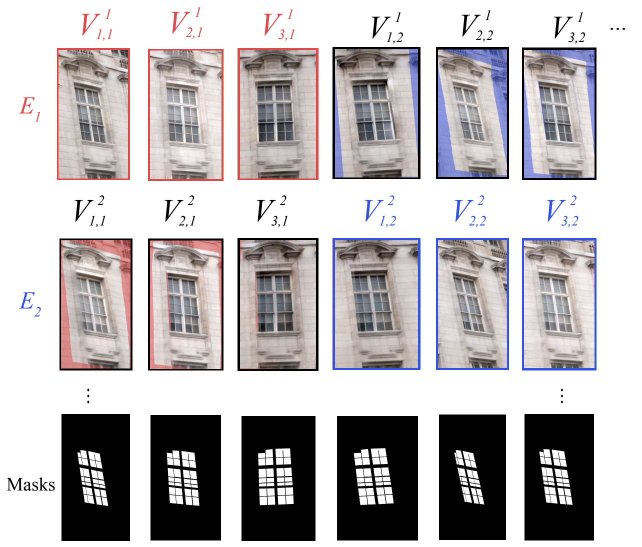


Figure 17: *Each instance E_i of a window is only seen in a subset of subviews (images with colored border). We augment this coherent subset by re-projecting the available subviews in other viewpoints, and fill-in occlusions and missing parts using the corrsponding subviews from another instance (colored areas in images with black border). We also re-project the reflection mask in all subviews.*

Our solution consists in augmenting the coherent set of each instance by reprojecting the available subviews into their closest missing subviews. Since we only have a few available subviews per instance, parts of the window may be occluded in other subviews. We fill in these parts using content from another instance for which this particular subview is available. We introduce additional notation to indicate the *target* instance k: $V^k_{i,j}$. When the physical instance j is equal to the target instance k, all pixels are covered in the sub-view, see Fig. 17. When the target instance $k \neq j$ some pixels need to be reprojected from other views to complete the cropped image.

Fig. 17 illustrates the resulting data augmentation on two window instances, each seen in three subviews, yielding six coherent subviews for each instance. Note that this overall process is similar in spirit to the re-projection performed at runtime by the image-based rendering algorithm to generate novel virtual views. However, augmenting the subviews in an offline pre-process allows us to employ a costly gradient-domain fusion [PGB03] when combining parts from different instances at occlusions. The color coding in Fig. 17 indicates the source of the pixels in each subview: colored $V^k_{i,j}$ with outlined subiews are cases where target and and physical instance are the same, while those in black include reprojection. Note that the colored pixels in the reprojected views correspond to the source instance used; in the general case these can come from several instances. The effect of this augmentation is shown in Fig. 18.

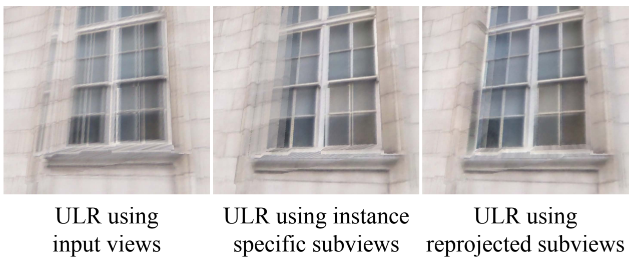

ULR using input views
ULR using instance specific subviews
ULR using reprojected subviews

Figure 18: *IBR comparison on our full scene mesh, using resp. the three input views, each instance original subviews, and the reprojected subviews for each instance.*

6.3. Complete Facade

We now need to create a complete model of the facade, requiring us to make the window-scale subviews compatible with the facade-scale input images. The main challenge we face at this stage is that the facade-scale images are not perfectly aligned with the subviews and refined 3D mesh, which results in visible seams if the two sets of images are combined naively during image-based rendering (see Fig. 19(a)). For each subview, our solution is to project the closest input views into it, and stitch the two images such that the subview is kept over the window, while the input image is used over the surrounding walls. We apply Digital Photomontage [ADA*04] to achieve a seamless composite, using the blurred bounding box of the subview as the foreground/background unary term (see Fig. 19(b)).

This approach also helps with color differences between the subviews and the facade due to exposure variations in the scene. As a result, subviews are seamlessly composited and we can place the instances at different locations on the facade, creating a novel facade (see Fig. 20).

7. Implementation and results

The final augmented dataset contains the matrix of augmented views, $V^k_{i,j}$, the corresponding cameras $C^*_{i,j}$ and the augmented geometry. In addition, we have the selected environment map that will be used for IBL.

7.1. Rendering

We render the scene using an extension of the Unstructured Lumigraph Rendering (ULR) method [BBM*01], that computes blending weights between input images for each pixel of the desired output. Our expanded set of cameras C^* and images V^k_{ij} augment the image data available for the regions in the windows. In the remaining areas of the facade plane, it is equivalent to a ULR render using solely the three input images. In the regions of the platonic elements, only the platonic cameras are used, to avoid any ghosting caused by the unprocessed input images.

To restore plausible view-dependent effects that are important to the perceived accuracy and realism of the scene, we overlay a reflection layer over the output image, using the environment map

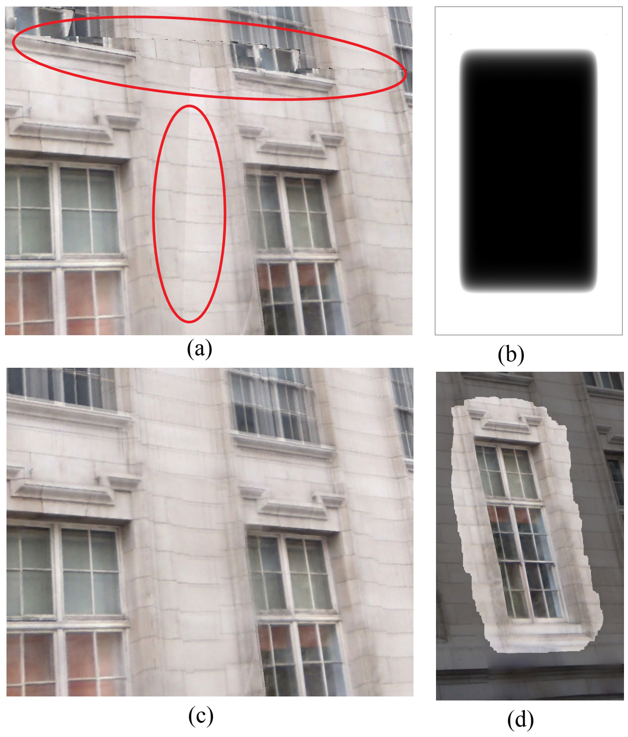


Figure 19: *(a) If the crop is blended naively, we observe visible seams. (b) Unary terms weighting with distance from boundaries. (c) After the graph cut, visible seams are eliminated. (d) Resulting cut.*

previously selected. For each pixel lying in one of the areas delimited by the reflection masks (Sec. 6.1), we reflect the view-to-point vector with respect to the facade plane normal and use the resulting direction to query the environment map. The resulting color is additively blended with the underlying diffuse color.

7.2. Implementation

We implemented our method in our custom C++ framework and render in an OpenGL pipeline. For the individual components we used the implementation of [WFP11] to rectify input images, and a Tensorflow implementation of the U-net for window segmentation. The entire preprocessing pipeline requires between 10 and 30 minutes for all our scenes; camera calibration and reconstruction requires less than 5 minutes both for input and platonic scenes. All timings are reported on an Intel Xeon 12 core 2.6GHz machine with 32Gb of memory. Rendering is real-time.

7.3. Results

We validate our results on five scenes, two with facades from London (3 input images each, from Ceylan et al. [CMZP13]) and three from Paris (3 input images for Paris1 and 4 for the others). Scenes are processed as described in the previous sections. Scenes Paris2 and Paris3 required manual definition of the plane. In Fig. 21, for each scene we show the input augmented geometry and compare

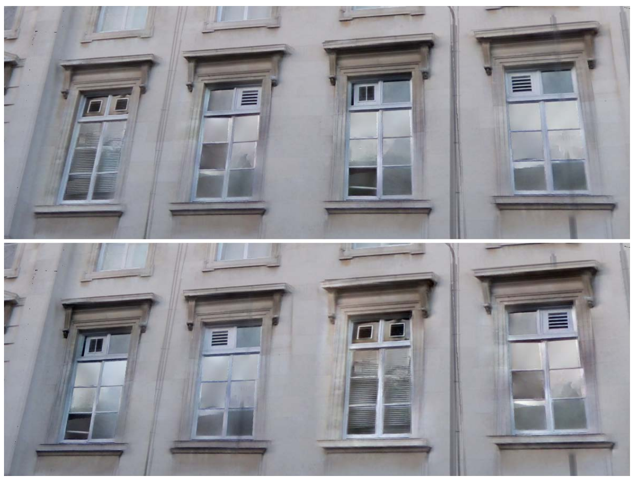


Figure 20: *Our method allows for instances of the platonic element to be swapped and reinserted in the final scene in a coherent manner. Top: original facade, bottom: novel facade.*

our result to the baseline result rendered with ULR rendering on the MVS reconstruction using the input images. The resulting improvement in quality is more clearly visible in the supplemental video, when changing the novel view and for the IBL rendering of reflections. Our contributions could also benefit rendering algorithms based on a diffuse textured mesh. In particular, the refined mesh yields better alignment of the re-projected images, which could benefit multi-view texturing. We present a comparison between textured meshes generated from our data and our image-based rendering approach in Fig. 22. Additional comparisons, illustrating the relative advantages of each part of our method can also be found in the accompanying videos. As shown in Fig. 20, a novel facade can be generated from an existing scene by swapping window instances.

8. Limitations and future work

Limitations. We currently rely on user intervention for subview selection. A clustering algorithm with a specific similarity metric that allows for minor differences between instances of a group could be used. Our algorithm is sensitive to the precision of the original calibration and reconstruction of the input images. In some cases, even though the MVS reconstruction fails completely we are able to produce a usable result, albeit with some artifacts (Fig. 23). Improved resolution street-side captures (e.g., [Eth17]) should provide sufficient quality that will allow our approach to be used. Eliminating the manual steps of subview selection and plane extraction (when needed) would allow the approach to be used at a much larger scale.

Future Work. In future work, it would be interesting to generalize our approach to sub-blocks of a given platonic element, e.g., a pediment shared by a door and a window. Such a generalization would allow our method to treat elements that are not necessarily repetitive, e.g., a door that shares sufficient sub-elements with repetitive windows. This would also allow our approach to treat elements other than windows. Another interesting avenue of future work would be the usage of our approach as a component for procedural modelling and generation. Our augmented subiews could be used as part of a procedural modelling system, allowing the generation of different combinations of the different instances. Such a method would need to include a way to generate the rest of the facade in a procedural manner, while being consistent with the requirements of IBR.

9. Acknowledgements

This research was partially funded by the European Union Horizon 2020 research and innovation programme under grant agreement No. 727188 EMOTIVE, http://www.emotiveproject.eu/, and by generous donations from Technicolor and Adobe. The authors thank J.-P. Bauchet and F. Lafarge for their help with the KIPPI segmentation and J. Philip for his plane fitting implementation.

References

[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 (ToG) 23*, 3 (2004), 294–302. 9

[ARB07] ALIAGA D. G., ROSEN P. A., BEKINS D. R.: Style grammars for interactive visualization of architecture. *IEEE transactions on visualization and computer graphics 13*, 4 (2007). 3

[AWL*15] AITTALA M., WEYRICH T., LEHTINEN J., ET AL.: Two-shot svbrdf capture for stationary materials. *ACM Trans. Graph. 34*, 4 (2015), 110–1. 2

[AYLM13] ALHALAWANI S., YANG Y.-L., LIU H., MITRA N. J.: Interactive facades analysis and synthesis of semi-regular facades. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 215–224. 2

[BBM*01] BUEHLER C., BOSSE M., MCMILLAN L., GORTLER S., COHEN M.: Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 425–432. 2, 9

[BL18] BAUCHET J.-P., LAFARGE F.: KIPPI: KInetic Polygonal Partitioning of Images. In *Proc. of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, US, 2018). 6

[CDSHD13] CHAURASIA G., DUCHENE S., SORKINE-HORNUNG O., DRETTAKIS G.: Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG) 32*, 3 (2013), 30. 1, 2

[CMZP13] CEYLAN D., MITRA N. J., ZHENG Y., PAULY M.: Coupled structure-from-motion and 3d symmetry detection for urban facades. *ACM Transactions on Graphics* (2013). 9

[DAB15] DEMIR I., ALIAGA D. G., BENES B.: Procedural editing of 3d building point clouds. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 2147–2155. 3

[DMIF15] DEKEL T., MICHAELI T., IRANI M., FREEMAN W. T.: Revealing and modifying non-local variations in a single image. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 227. 2

[DRSVG13] DAI D., RIEMENSCHNEIDER H., SCHMITT G., VAN GOOL L.: Example-based facade texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 1065–1072. 3

[Eth17] ETHERINGTON D.: Google's street view cameras get a high-res update focused on ai. https://techcrunch.com/2017/09/05/googles-street-view-cameras-get-a-high-res-update-focused-on-ai/, 2017. 10

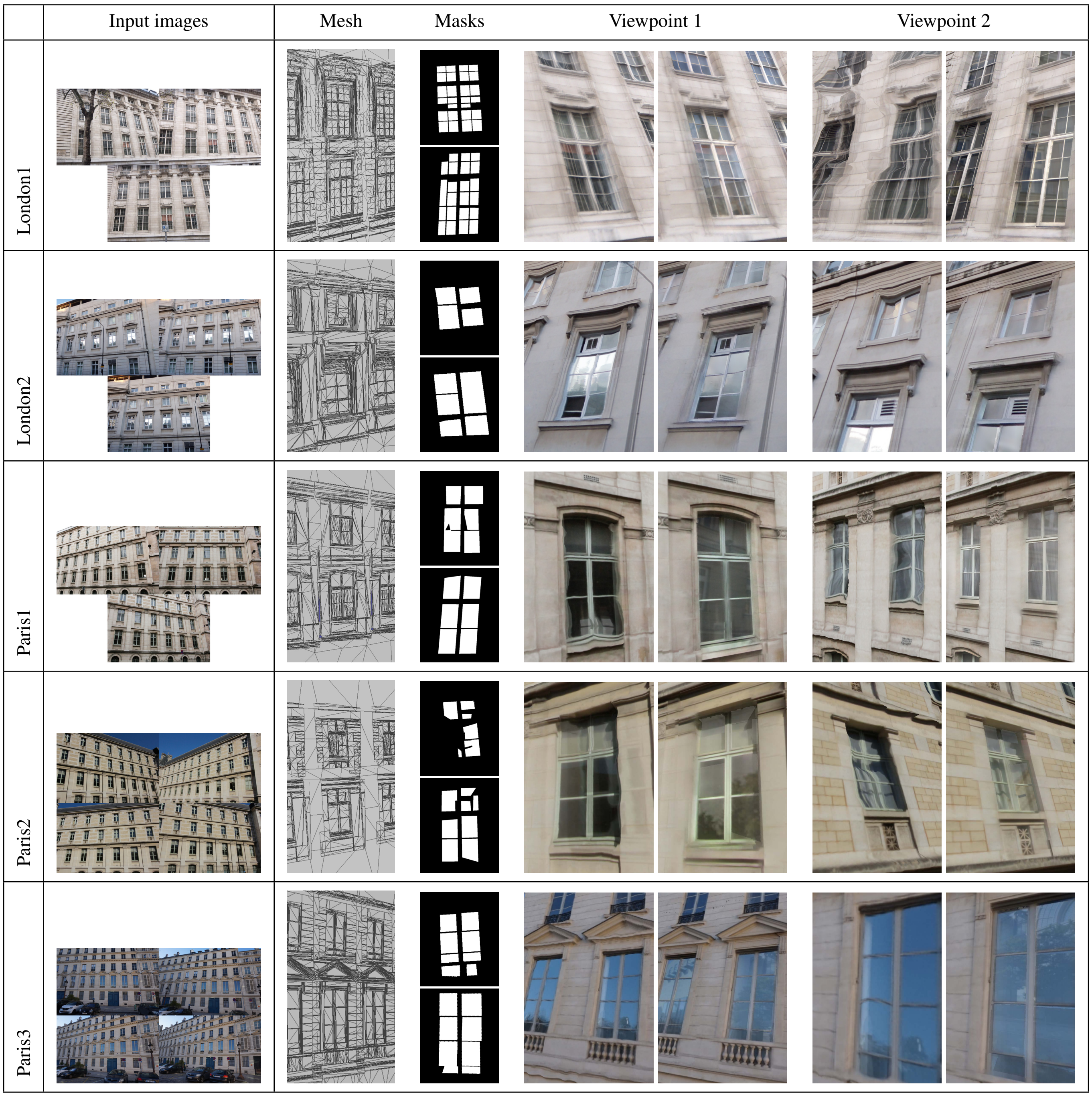


Figure 21: *Results on 2 scenes from London and 3 from Paris. Left to right: input images, augmented geometry, extracted reflection masks, baseline rendering (ULR) and our result on a first viewpoint, baseline rendering (ULR) and our result on a second viewpoint.*

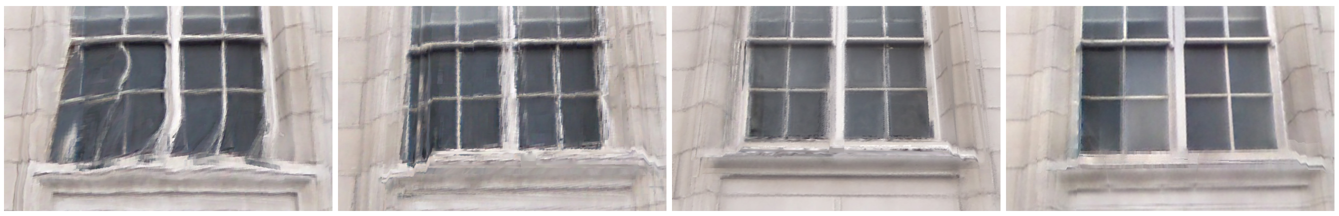


Figure 22: *Multi-view texturing in this figure was performed using the texturing module of RealityCapture [Rea18]. From left to right: textured mesh reconstructed using the 3 input views, our refined mesh textured using the 3 input views, our refined mesh textured with all our generated subviews, and our IBR solution for the same view.*

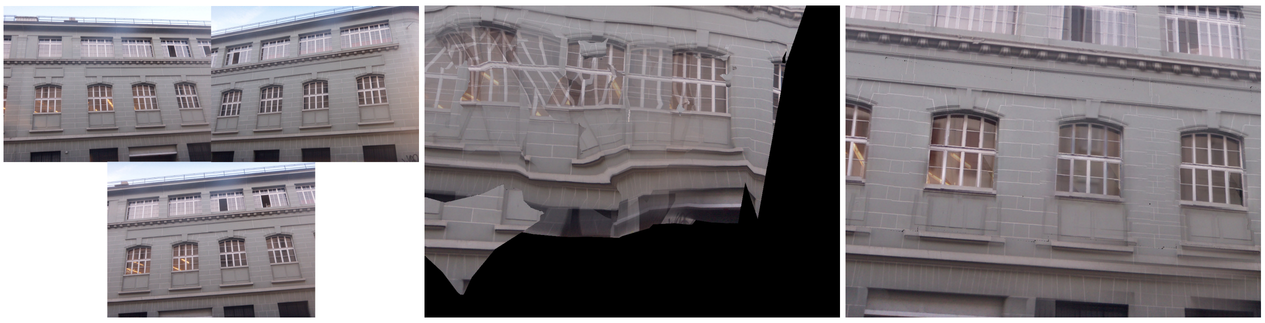


Figure 23: *On this scene where standard MVS reconstruction fails (middle row), we obtain a usable result (right).*

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 43–54. 2

[HDF14] HEINLY J., DUNN E., FRAHM J.-M.: Correcting for duplicate scene structure in sparse 3d reconstruction. In *European Conference on Computer Vision* (2014), Springer, pp. 780–795. 3

[HRDB16] HEDMAN P., RITSCHEL T., DRETTAKIS G., BROSTOW G.: Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 231. 1, 2

[JTC09] JIANG N., TAN P., CHEONG L.-F.: Symmetric architecture modeling with a single image. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 113. 3

[JTC11] JIANG N., TAN P., CHEONG L.-F.: Multi-view repetitive structure detection. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 535–542. 3

[KLS*13] KOPF J., LANGGUTH F., SCHARSTEIN D., SZELISKI R., GOESELE M.: Image-based rendering in the gradient domain. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 199. 2, 7

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 31–42. 2

[LHOK*10] LIU Y., HEL-OR H., KAPLAN C. S., VAN GOOL L., ET AL.: Computational symmetry in computer vision and computer graphics. *Foundations and Trends® in Computer Graphics and Vision 5*, 1–2 (2010), 1–195. 2

[MMMO] MOULON P., MONASSE P., MARLET R., OTHERS: Openmvg. an open multiple view geometry library. https://github.com/openMVG/openMVG. 5

[MRM*10] MUSIALSKI P., RECHEIS M., MAIERHOFER S., WONKA P., PURGATHOFER W.: Tiling of ortho-rectified facade images. In *Proceedings of the 26th Spring Conference on Computer Graphics* (2010), ACM, pp. 117–126. 3

[MZWVG07] MÜLLER P., ZENG G., WONKA P., VAN GOOL L.: Image-based procedural modeling of facades. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 85. 3

[OCDD15] ORTIZ-CAYON R., DJELOUAH A., DRETTAKIS G.: A Bayesian Approach for Selective Image-Based Rendering using Superpixels. In *International Conference on 3D Vision - 3DV* (Lyon, France, Oct. 2015). URL: https://hal.inria.fr/hal-01207907. 1

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. In *ACM Transactions on graphics (TOG)* (2003), vol. 22, ACM, pp. 313–318. 8

[PZ17] PENNER E., ZHANG L.: Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 235. 1

[Rea18] REALITY C.: Realitycapture reconstruction software. https://www.capturingreality.com/Product, 2018. 5, 12

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241. 4

[RT13] RADIM TYLEČEK R. Š.: Spatial pattern templates for recognition of objects with regular structure. In *Proc. GCPR* (Saarbrucken, Germany, 2013). 4

[SCK08] SHUM H.-Y., CHAN S.-C., KANG S. B.: *Image-based rendering*. Springer Science & Business Media, 2008. 2

[SKG*12] SINHA S. N., KOPF J., GOESELE M., SCHARSTEIN D., SZELISKI R.: Image-based rendering for scenes with reflections. *ACM Trans. Graph. 31*, 4 (2012), 100–1. 2

[Wei01] WEISS Y.: Deriving intrinsic images from image sequences. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001), vol. 2, IEEE, pp. 68–75. 7

[WFP10] WU C., FRAHM J.-M., POLLEFEYS M.: Detecting large repetitive structures with salient boundaries. *Computer Vision–ECCV 2010* (2010), 142–155. 4

[WFP11] WU C., FRAHM J.-M., POLLEFEYS M.: Repetition-based

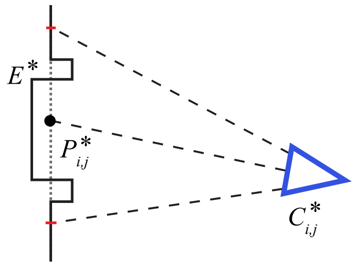


Figure 24: *We cast rays from the borders of the view associated to $C^*_{i,j}$, intersecting the platonic model (red points). These samples can be used to estimate the facade plane (dotted grey line).*

dense single-view reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 3113–3120. 3, 9

[XEOT12] Xiao J., Ehinger K. A., Oliva A., Torralba A.: Recognizing scene viewpoint using panoramic place representation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 2695–2702. 8

[XFT*08] Xiao J., Fang T., Tan P., Zhao P., Ofek E., Quan L.: Image-based façade modeling. In *ACM transactions on graphics (TOG)* (2008), vol. 27, ACM, p. 161. 3, 5

[XWL*08] Xu X., Wan L., Liu X., Wong T.-T., Wang L., Leung C.-S.: Animating animal motion from still. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 117. 2

[ZC93] Zerubia J., Chellappa R.: Mean field annealing using compound gauss-markov random fields for edge detection and image estimation. *IEEE Transactions on Neural Networks 4*, 4 (1993), 703–709. 6

[ZSW*10] Zheng Q., Sharf A., Wan G., Li Y., Mitra N. J., Cohen-Or D., Chen B.: Non-local scan consolidation for 3d urban scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 29*, 4 (2010), 94:1–94:9. 3

[ZYZQ12] Zhao P., Yang L., Zhang H., Quan L.: Per-pixel translational symmetry detection, optimization, and segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 526–533. 3

Appendix A: Transformation from platonic scene to input scene

We need to estimate a transformation T from the platonic frame to the input scene frame, in order to place the platonic model and cameras back into the input scene frame. We evaluate a series of candidate transformations $T_{i,j}$ using each platonic camera separately (see Fig. 11).

For each platonic camera $C^*_{i,j}$, we estimate an approximate corresponding camera $C_{i,j}$ in the input scene frame. As in Sec. 4.2, we compute it from the input camera C_i and the cropped image $V_{i,j}$ location in V_i. We compute a transformation from camera $C^*_{i,j}$ to $C_{i,j}$ by aligning their positions, and their direction, up and right vectors. This transformation only characterizes a rotation and translation, leaving an unknown scaling factor between the two frames.

To lift this ambiguity, we use the distances between the camera and the facade, in the platonic and input frames (respectively $d^*_{i,j}$ and $d_{i,j}$).

The optical center of $C_{i,j}$ intersects the known facade plane (and thus E_j) at $P_{i,j}$. Similarly the optical center of $C^*_{i,j}$ intersects the platonic facade plane at $P^*_{i,j}$ (Fig. 24). This point does not necessarily belong to the platonic mesh, due to geometric details such as window recess. We estimate the plane of the facade in the platonic frame by sampling pixels near the borders of $V_{i,j}$, and casting rays towards the mesh. The motivation is that such intersection points will belong to the facade wall. From these points we can estimate the facade plane. We intersect a central ray with it to obtain $P^*_{i,j}$.

Then, $d^*_{i,j}$ is the distance between $P^*_{i,j}$ and $C^*_{i,j}$, $d_{i,j}$ the distance between $P_{i,j}$ and $C_{i,j}$. We estimate the scaling factor s from the ratio of these distances, taking into account the field of view values of $C^*_{i,j}$ and $C_{i,j}$.

$$s = \frac{d_{i,j}}{d^*_{i,j}} \frac{\tan(fov_{C_{i,j}}/2)}{\tan(fov_{C^*_{i,j}}/2)}$$

In practice, we express the transformation from platonic scene space to $C^*_{i,j}$ image space, then estimate the scaling factor; we compose this result with the transformation from $C_{i,j}$ image space to input scene space. The candidate $(T_{i,j})$ transformations are centered (from E_j to the origin) and averaged to obtain the final transformation T. This transformation is used to place the platonic mesh and cameras back into the input scene frame, before duplicating and translating them at each E_j position using $P_{i,j}$.