# TwinTex: Geometry-aware Texture Generation for Abstracted 3D Architectural Models

WEIDAN XIONG, Shenzhen University, China
HONGQIAN ZHANG, Shenzhen University, China
BOTAO PENG, Shenzhen University, China
ZIYU HU, Guangdong Artificial Intelligence and Digital Economy Laboratory (SZ), Shenzhen University, China
YONGLI WU, Guangdong Artificial Intelligence and Digital Economy Laboratory (SZ), Shenzhen University, China
JIANWEI GUO, MAIS, Institute of Automation, Chinese Academy of Sciences, China
HUI HUANG*, Shenzhen University, China

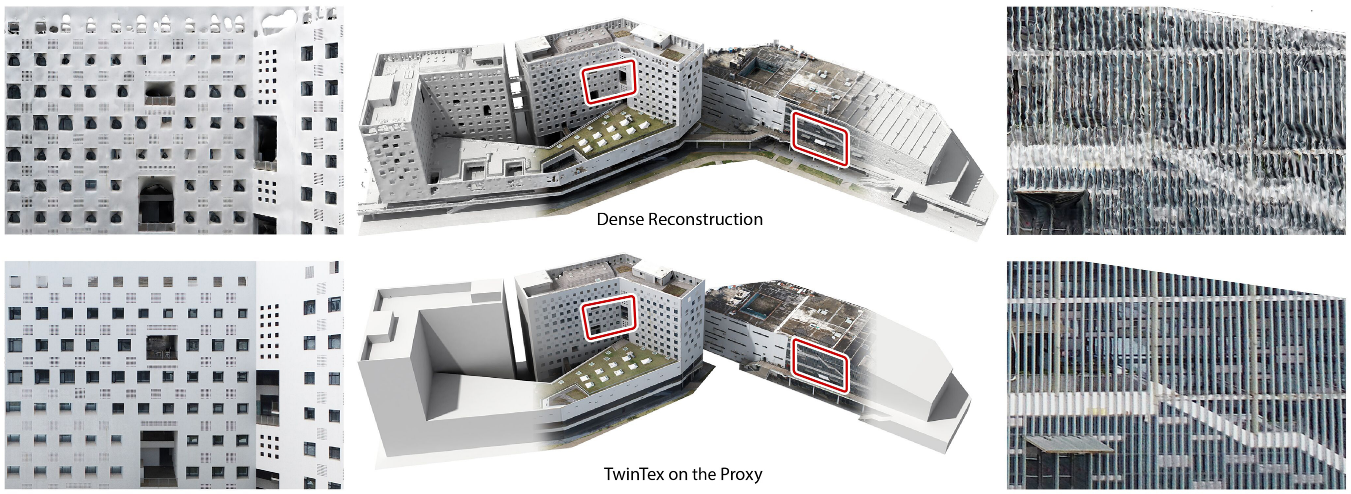


Fig. 1. We introduce **TwinTex**, a fully automatic method to produce a detailed texture map with high photometric and perceptual quality for a piece-wise planar 3D proxy mode. Such a proxy (1,489 vertices, 739 faces, shown in the bottom-middle) is an abstracted version of a 3D dense model (656,571 vertices, 1,313,550 faces, shown at the top-middle), which is reconstructed from multi-view aerial images captured by a drone.

Coarse architectural models are often generated at scales ranging from individual buildings to scenes for downstream applications such as Digital Twin City, Metaverse, LODs, etc. Such piece-wise planar models can be abstracted as twins from 3D dense reconstructions. However, these models typically lack realistic texture relative to the real building or scene, making them unsuitable for vivid display or direct reference. In this paper, we present *TwinTex*, the first automatic texture mapping framework to generate a photo-realistic texture for a piece-wise planar proxy. Our method addresses most challenges occurring in such twin texture generation. Specifically, for each primitive plane, we first select a small set of photos with greedy heuristics considering photometric quality, perspective quality and facade texture completeness. Then, different levels of line features (LoLs) are extracted from the set of selected photos to generate guidance for later steps. With LoLs, we employ optimization algorithms to align texture with geometry from local to global. Finally, we fine-tune a diffusion model with a multi-mask initialization component and a new dataset to inpaint the missing region. Experimental results on many buildings, indoor scenes and man-made objects of varying complexity demonstrate the generalization ability of our algorithm. Our approach surpasses state-of-the-art texture mapping methods in terms of high-fidelity quality and reaches a human-expert production level with much less effort.

*Corresponding author: Hui Huang (hhzhiyan@gmail.com)

Authors' addresses: Weidan Xiong, xiongweidan@gmail.com, Shenzhen University, China; Hongqian Zhang, Shenzhen University, China; Botao Peng, Shenzhen University, China; Ziyu Hu, Guangdong Artificial Intelligence and Digital Economy Laboratory (SZ), Shenzhen University, China; Yongli Wu, Guangdong Artificial Intelligence and Digital Economy Laboratory (SZ), Shenzhen University, China; Jianwei Guo, MAIS, Institute of Automation, Chinese Academy of Sciences, China; Hui Huang, hhzhiyan@gmail.com, Shenzhen University, College of Computer Science & Software Engineering, China.

0730-0301/2023/12-ART $15.00
https://doi.org/10.1145/3618328

CCS Concepts: • **Computing methodologies → Shape modeling**.

Additional Key Words and Phrases: Texture Mapping, 3D Architectural Proxy, View Selection, Image Stitching, Texture Optimization, Diffusion Model

**ACM Reference Format:**
Weidan Xiong, Hongqian Zhang, Botao Peng, Ziyu Hu, Yongli Wu, Jianwei Guo, and Hui Huang. 2023. TwinTex: Geometry-aware Texture Generation

for Abstracted 3D Architectural Models. *ACM Trans. Graph.* 42, 6 (December 2023), 14 pages. https://doi.org/10.1145/3618328

## 1 INTRODUCTION

With the advances in active sensors (*e.g.,* LiDAR), fully automated *Multiple View Stereo* (MVS) and aerial-based photogrammetry sensing technology, many mature algorithms have been developed to generate complete and high-fidelity 3D dense reconstruction [Musialski et al. 2013; Smith et al. 2018; Zhou et al. 2020] with textures [Waechter et al. 2014]. Architectural models with simplified topology and twin shapes of such dense models can be generated from procedural modeling, approximated from dense reconstruction, or created manually by artists and architects. These abstracted twin models are tractable in plenty of downstream applications due to their sharper features, much lower needs on storage, and more suitable for transmission and real-time rendering, as shown in Fig. 2.

However, the texture information contained in the original dense reconstruction is often not maintained after abstraction. Thus, with all the calibrated RGB images, how to texture the twin proxy for the level of realism then becomes a challenging problem. An intuitive way is to employ image-based texture generation methods [Lempitsky and Ivanov 2007; Waechter et al. 2014; Zhou and Koltun 2014]. However, these approaches would produce distorted and misaligned artifacts as illustrated in Fig. 3. Although patch-based optimization method [Bi et al. 2017] can reduce such artifacts to some extent by synthesizing photometrically-consistent aligned images to correct misalignments, it would possibly deliver blurring textures or missing regions. These issues are mainly caused by: i) the inaccurate camera parameters, ii) the significant geometric differences between the twin mesh and ground-truth models brought by the abstraction process, and iii) the large difference among viewing angles. The combination of these cases makes our problem even more challenging and complicated. The urban proxy models often exhibit large-scale planar structures. In practice, the modeling artists often manually create high-quality textures plane-by-plane. This is an extremely tedious, time-consuming, and labor-intensive process. It took quite a few days for a professional modeler to texture an abstracted model such as the Polytech shown in Fig. 1 with real photos. Texturing a model with a higher level of detail would take much longer.

In this paper, we propose a plane-based methodology and fully automated algorithms to generate high-quality texture maps for piecewise planar architectural models given a large set of unordered RGB photos. Our method addresses most challenges occurring in such twin texture generation: the large number and high resolution of input images, their drastically varying properties such as photometric quality, perspective quality (large variation of viewing angles), heavy occludes (*e.g.,* trees and buildings), and texture-geometry misalignment (introduced by incorrect camera parameters and model abstraction process). Given a coarse model, our algorithm decomposes it into a set of planar polygonal shapes, for each of which we aim at generating a high-quality and complete texture map. We first present a plane-oriented view selection approach to select views that best cover each planar shape with high photometric and perspective consistency. Based on the line segments extracted from each view, a line-guided texture stitching is introduced to create a single texture map to preserve geometric features. Finally, we perform texture

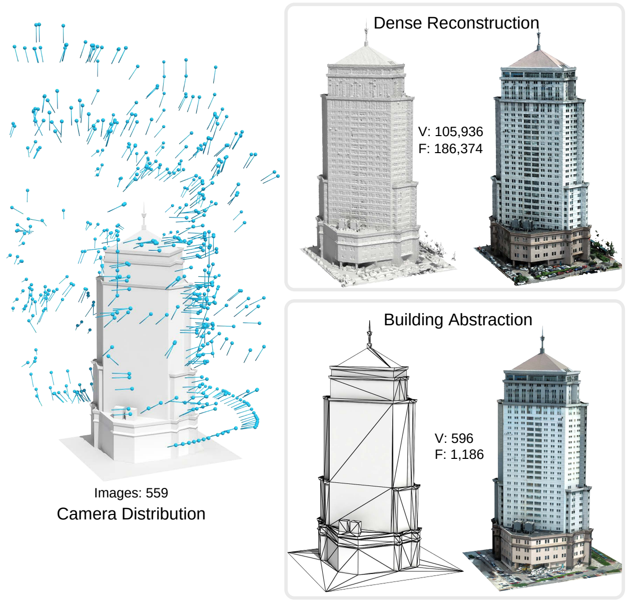


Fig. 2. Left: Aerial distribution of the drone that collected the input photos (559 images). Top right: 3D dense reconstruction of a Headquarter. Bottom right: An abstracted twin model with a much smoother appearance and smaller storage needs.

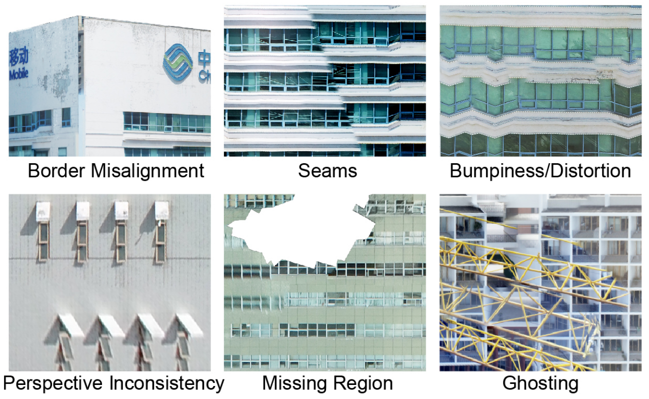


Fig. 3. Illustration of several typical issues occurred in texturing architectural proxies.

optimization to improve the illumination consistency and fill the missing regions in the texture map. Experiments are performed to show the effectiveness and robustness of our method. In summary, our work makes the following contributions:

- A fully automatic **TwinTex**, which produces high-quality and geometry-aware texture for a simplified proxy model.
- A view selection algorithm that can select a small number of candidate photos for each primitive plane considering texture completeness, perspective and photometric quality.
- A plane-based texture mapping methodology via geometry-aware image stitching guided by LoLs.

- An architectural dataset including various scenes and building components. We utilize this dataset and fine-tune a diffusion probabilistic model with a novel multi-mask initialization component to complete the texture maps.

## 2 RELATED WORK

### 2.1 Image-based Texture Mapping

Texture mapping is an important approach for authoring photo-realistic 3D objects without increasing their geometric complexity [Yuksel et al. 2019]. View-dependent texture mapping works [Debevec et al. 1998, 1996] composite multiple views of a scene by assuming the existence of a global mesh and storing all appearance variation in textures. Although they also use a view selection procedure to project a single image onto the model and then merge several image projections, they only determine for each polygon in the model in which photos it is visible. Instead, we present a novel view selection algorithm that considers texture completeness, perspective quality, and photometric quality.

To generate a single consistent texture map, early works on image-based texture mapping usually select multiple images for each face and blend them into textures by using different weighting strategies [Bernardini et al. 2001; Callieri et al. 2008]. However, such methods could generate blurry and ghosting results due to camera calibration errors and inaccurate geometry. To reduce those artifacts caused by multi-image blending, later approaches choose to select only one view per face, then conduct seam optimization to create a texture patch and to avoid visible seams between adjacent patches [Fu et al. 2020, 2018; Gal et al. 2010; Lempitsky and Ivanov 2007; Waechter et al. 2014; Wang et al. 2018]. They typically solve a discrete labeling problem by using different data terms and smoothness terms to construct the Markov random field to select an optimal texture image for each face. Whereas, these approaches still suffer from misaligned seams in challenging cases. Another category of warping-based methods jointly rectifies the camera poses and geometric errors. Zhou and Koltun [2014] use local image warping to optimize camera poses in tandem with non-rigid correction functions for all images. This approach is then extended by [Bi et al. 2017] to propose a patch-based optimization that handles larger geometry inaccuracies by correctly aligning the input images. This method estimates the bidirectional similarity of different images, which suffers from high computational costs.

Although high-quality texture maps can be generated from the above methods, the resultant quality of texture mapping still greatly depends on the quality of reconstruction and accuracy of camera calibration. None of them can be naively applied to generate the realistic texture of a large-scale structured building or scene. Firstly, the abstracted proxy model has significant structural differences from the original dense mesh. Moreover, constrained by the complicated environment, the collected aerial data often has highly uneven camera distribution, large camera calibration errors and large variations in viewing angles. Current methods still cannot handle these problems well, resulting in obvious artifacts. Instead, we choose to generate texture in a plane-based manner that is more intuitive and human alike.

### 2.2 Structure-aware Scene Reconstruction and Texturing

Creating structured models from noisy raw data has become an ever-increasing demand in urban digitization. Ceylan et al. [2012, 2014] propose image-based 3D urban building reconstruction frameworks by exploiting the presence of linear and symmetric features (*e.g.,* lines, repeated elements) in facade for image registration and 3D reconstruction. To generate simplified meshes, a popular way is to first reconstruct a 3D dense mesh and then simplify it using geometry simplification [Garland and Heckbert 1997] or piece-wise planar structure approximation methods [Cohen-Steiner et al. 2004; Salinas et al. 2015; Verdie et al. 2015]. Another way is to directly detect a set of planar primitives from the point clouds, then explore the arrangements of primitives to provide a compact polygonal mesh [Bauchet and Lafarge 2020; Bouzas et al. 2020; Fang and Lafarge 2020; Guo et al. 2022; Monszpart et al. 2015; Nan and Wonka 2017; Pan et al. 2022]. In practice, such proxy models are usually generated manually or by procedural modeling [Sinha et al. 2008] with arbitrary topology (*e.g.,* manifolds or non-manifolds).

However, texturing such abstracted building models has drawn less attention. Several methods take into account the generation of texture maps while conducting a lightweight geometric reconstruction [Garcia-Dorado et al. 2013; Huang et al. 2017; Maier et al. 2017; Sinha et al. 2008; Wang and Guo 2018]. For example, Garcia-Dorado et al. [2013] computes a per-building volumetric proxy, then uses a surface graph-cut method to stitch aerial images and yield a visually coherent texture map. However, their reconstruction results are 2.5D which can only restore textures under limited perspectives, losing important structures. Recently, deep learning approaches based on image-image translation networks have been proposed to synthesize texture details for coarse meshes of urban areas [Georgiou et al. 2021; Kelly et al. 2018]. Although data-driven methods are able to generate various styles of textures, they require extensive training sets, and the synthesized textures lack realism comparing to the original real scenes. The technique of NeRF [Müller et al. 2022] shows great ability to generate high-quality 3D shapes and textures [Baatz et al. 2022; Metzer et al. 2022]. However, remember that we aim at very high-resolution input and output images which require extremely large memory, employing a small set of input images is not enough for NeRF-based methods to generate satisfying results, which makes them less applicable in our scenario.

### 2.3 Image Stitching and Completion

Image stitching is to combine multiple images with overlapping sections to produce a single panoramic or high-resolution image [Mehta and Bhirud 2011; Szeliski et al. 2007]. Early methods estimate an optimal global transformation for each input image [Brown and Lowe 2007]. However, they only work well for scenes near planar or with slight view disparity, and usually generate ghosting artifacts or projective distortion for general scenes. For more accurate stitching, many approaches adopt spatially-varying warps to process different regions of an image, including smoothly varying affine transformation [Lin et al. 2011], dual or quasi homographies [Gao et al. 2011; Li et al. 2017], shape-preserving half-projective warps [Chang et al. 2014; Lin et al. 2015], and smoothly varying homography

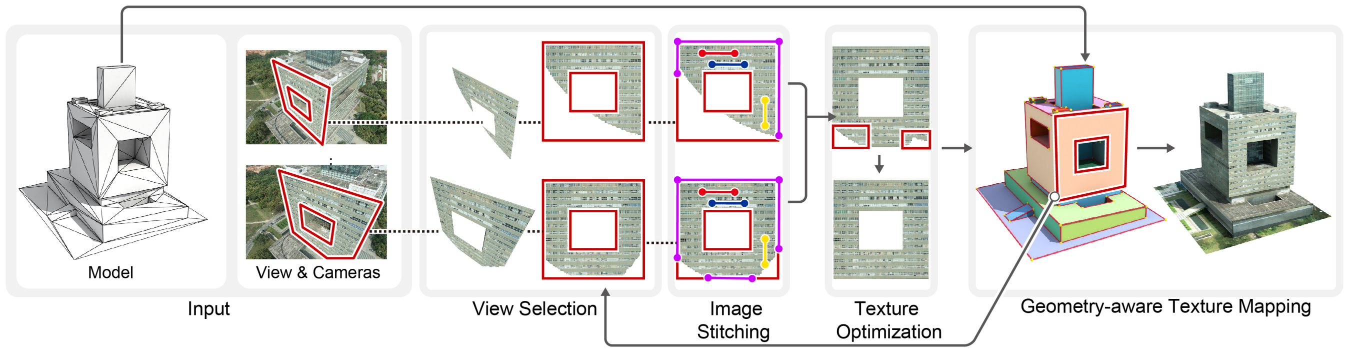


Fig. 4. Overview of our algorithm on the Hitech building. Taking an abstracted model and the UAV images as input, our framework detects the proxy polygons in the mesh and generates one stitched texture map for each proxy polygon. Firstly, a small subset of views is selected from all the visible views for each proxy polygon. Then, all the selected views are stitched together to form an enlarged texture map. Note that important geometric primitives such as line segments contained in the original images are well preserved after the image stitching. Finally, the texture map is optimized by performing global consistency across all the visible regions and inpainting the incomplete region of the underlying planar shape.

field [Zaragoza et al. 2013], etc. Aiming at building facades, Musialski et al. [2010] present a system for generating approximately orthographic facade textures from a set of perspective photographs. They perform multi-view image stitching over planar proxies by using a gradient-domain stitching to avoid visible seams.

However, they are usually time-consuming and unsuitable for scenes with large parallax and view disparity. Several methods are then proposed to obtain better alignment with less distortion, especially for large parallax [Lee and Sim 2020; Zhang et al. 2016], but linear structures in the image are not well maintained. After that, instead of just using point feature matching, line features are introduced to preserve both linear structures [Jia et al. 2021; Li et al. 2015; Liao and Li 2019; Xiang et al. 2018]. All these methods apply a uniform grid to warp the images, which are not flexible enough to align the regions with dense geometric features. Unlike them, we propose to use an adaptive grid to locally warp the images and better align the large-scale line features.

Image completion, also known as image inpainting, is the process of restoring missing or damaged areas in an image. Extensive research has been conducted on image completion, and existing methods can be classified into three types: 1) diffusion-based methods, 2) patch-based methods and 3) deep learning methods based on generative models. A detailed review is out of the scope of this paper, and we refer the reader to recent works [Dhariwal and Nichol 2021; Lugmayr et al. 2022] and surveys [Elharrouss et al. 2020; Guillemot and Meur 2014; Jam et al. 2021; Rojas et al. 2020].

## 3 PROBLEM STATEMENT AND OVERVIEW

The objective of our approach is to automatically generate realistic and geometry-aware texture maps for a piece-wise planar model, *i.e.,* urban building or scene. Our algorithm takes a set of calibrated camera views $\{\mathcal{I}\}$ and an abstracted model $\mathcal{M}$ as input, and outputs high-quality texture maps. The proxy model $\mathcal{M}$ is a polygonal mesh consisting of a set of planar polygonal shapes $\mathcal{P}_i$, each of which is referred to as a *proxy polygon.*

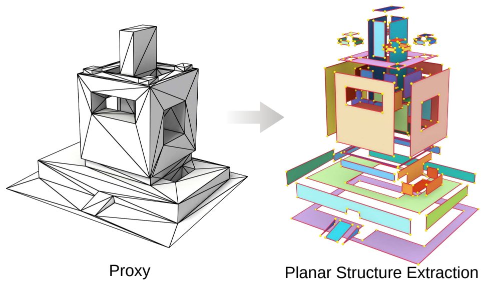


Fig. 5. The input mesh $\mathcal{M}$ is first segmented into a set of planar regions represented with proxy polygons $\{\mathcal{P}_i\}$. The shape of each proxy polygon is then extracted as proxy boundary $\mathcal{B}_i$ (visualized with red border lines).

To produce a texture map for each proxy polygon $\mathcal{P}_i$, the first step should be to select one or multiple images as candidate maps for $\mathcal{P}_i$. However, unlike previous texturing problems dealing with dense models (small triangles), it is usual that none of the given images can cover such large proxy polygons in abstracted versions. Hence, we employ the latter scheme and select multiple images for each polygon shape. However, there are still several challenges that exist in our scenario: i) Photometric issue. There are seams, illumination resolution inconsistency between adjacent patches. For large scale scenes, the frequent ghosting effect is mainly introduced by the dynamic instances, such as cars and tower crane. ii) Perspective issue. Even with photometric consistent patches, it would still be apparent that the stitched texture comes from several images with largely varied viewing angles. iii) Image-to-image and image-to-geometry misalignment. This is introduced by both the inaccuracy of camera parameters and the simplification process. iv) Facade incompleteness. In the absence of semantic guidance, how to fill a largely empty region with geometrically and semantically consistent content remains a problem.

To address the above problems, we propose a plane-based and geometry-aware texture mapping methodology, as shown in Fig. 4. First, we extract high-quality views and visible regions for each

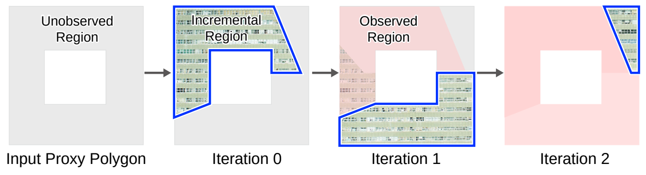


Fig. 6. The view selection process of a proxy polygon $\mathcal{P}_i$. The red polygons denote the observed regions, while the grey ones denote the unobserved regions. The polygon with blue outline denotes the incremental region $A_i^j$ of selected image $\mathcal{R}_j$ in the current iteration.

proxy polygon $\mathcal{P}_i$. An effective greedy heuristic method is proposed to select a small set of views $\{\mathcal{I}\}_i$ that best cover each $\mathcal{P}_i$ with high photometric and perspective consistency (Sec. 4). Next, in Sec. 5.1, the rich line features contained in the selected images $\{\mathcal{I}\}_i$ are detected and organized into Level of Lines (LoLs). After that, all the selected images are sequentially warped to their relative proxy boundary based on LoLs and stitched into a single texture map. Sec. 5.2 provides details. Then, each texture map is optimized by a series of operators to reveal illumination consistency. Finally, we provide an architectural image dataset and train a multi-mask Diffusion Model to inpaint the missing regions in texture maps, see Sec. 5.3. Our algorithm has been implemented as a customized plug-in of a widely used graphics engine called Houdini[1].

## 4 PLANE-ORIENTED VIEW SELECTION

*Proxy polygon extraction.* The input proxy $\mathcal{M} = (\mathcal{E}, \mathcal{F})$ can be created manually, generated procedurally, or abstracted from a dense reconstruction. $\mathcal{M}$ is a polygonal mesh where each basic element of $\mathcal{E}$ is an edge and $\mathcal{F}$ is either a polygonal or triangular facet, as shown in Fig. 5. We first segment $\mathcal{M}$ into a set of planar polygonal shapes, each of which is represented as a *proxy polygon* $\mathcal{P}_i$. We input $\mathcal{P}_i$ by utilizing a region-growing algorithm based on face normal. We then detect the corresponding proxy boundary $\mathcal{B}_i$ by choosing and sorting the boundary edges of $\mathcal{P}_i$. Fig. 5 shows the result of this step on the scene of Hitech.

**View selection.** Given a large set of input calibrated photos, the goal of this step is to select a small set of camera views $\{\mathcal{I}\}_i$ as candidate feature maps to best cover each proxy polygon $\mathcal{P}_i$. We first apply frustum-culling and visibility detection to filter out the views and pixels which are invisible, far away from $\mathcal{P}_i$, or have extremely inclined viewing angles. The filtered images are then projected onto $\mathcal{P}_i$, and the projected regions $\{\mathcal{R}\}$ covering the proxy polygon are extracted as candidate images, as shown in Fig. 4.

Next, we propose an iterative view selection algorithm to select a smaller set of candidate views from the projected images covering $\mathcal{P}_i$ (see Fig. 6). Starting from an empty set $\{\mathcal{I}\}_i^0 = \emptyset$, we select the best projected image $\mathcal{R}_j$ with the highest quality (denoted by $Q_j$) in each iteration $k$ to create $\{\mathcal{I}\}_i^k = \{\mathcal{I}\}_i^{k-1} \cup R_j$. This process converges when there is no projected image left to examine, or the $\{\mathcal{I}\}_i^k$ can fully cover $\mathcal{P}_i$, i.e., the area of unobserved regions in $\mathcal{P}_i$ is smaller than a given threshold $\tau$. We measure the quality $Q_j$ of each $\mathcal{R}_j$ by its incremental coverage ratio of the proxy polygon, its photometric and perspective quality and consistency with $\{\mathcal{I}\}_i^{k-1}$. The score $Q_j$ is calculated as the weighted sum of two terms:

$$Q_j = Q_{photo} + \lambda_p Q_{persp}, \tag{1}$$

where $Q_{photo}$ and $Q_{persp}$ represent the photometric and perspective quality, respectively. $\lambda_p$ is the weight of $Q_{persp}$.

*Photometric quality.* This term favors $\mathcal{R}_j$ with large incremental projection region, high resolution, and high photometric similarity to all the projected images and previously selected images. In detail, the coverage ratio of $\mathcal{R}_j$ against proxy polygon $\mathcal{P}_i$ at iteration $k$ is calculated as $A_i^j / {A_i}^k$. It is determined by the area of the incremental region $A_i^j$ normalized with the unobserved area ${A_i}^k$. Second, we measure the resolution of $\mathcal{R}_j$ as $G(\mathcal{R}_j)$ by computing the average of the gradient magnitude over all pixels within the projection region [Gal et al. 2010]. The photometric similarity, $C(\mathcal{R}_j, \{\mathcal{R}\})$, is defined by checking the photo-consistency of $\mathcal{R}_j$ against all the other projected images. It is computed by a multi-variate Gaussian function [Waechter et al. 2014]. Furthermore, we introduce a new smoothness term to give a more confident value to a projected image $\mathcal{R}_j$ when $\mathcal{R}_j$ is consistent with the current set of selected images $\{\mathcal{I}\}_i^{k-1}$. The final photometric quality term is organized as:

$$Q_{photo} = [\lambda_g G(\mathcal{R}_j) + \lambda_c C(\mathcal{R}_j, \{\mathcal{R}\})] \frac{A_i^j}{{A_i}^k} + \lambda_s Q_{smooth}, \tag{2}$$

$$Q_{smooth} = 1 - D_c(\mathcal{R}_j, \{\mathcal{I}\}_i^{k-1}), \tag{3}$$

where $\lambda_g$ and $\lambda_c$ are the respective weight of $G(\mathcal{R}_j)$ and $C(\mathcal{R}_j, \{\mathcal{R}\})$, $\lambda_s$ is the weight of $Q_{smooth}$. $D_c(\mathcal{R}_j, \{\mathcal{I}\}_i^{k-1})$ represents the mean of pixel-wise color difference between the overlapping pixels of image $\mathcal{R}_j$ and all the images in $\{\mathcal{I}\}_i^{k-1}$ after normalization. $Q_{smooth} = 1$ if iteration $k$ is zero or there exists no overlapping regions. The quality term $Q_{photo}$ tends to select a set of projected images with the maximum photometric and content consistency, which helps reducing the ghosting effect. Note that both photometric and smoothness terms consider overlap regions. $C(\cdot)$ directly uses the whole projected regions (including overlap regions) of all the projected images $R$. Meanwhile, the smoothness term only considers the overlap region compared to the candidate projected image $I$.

*Pespective quality.* However, only considering the photometric quality will possibly cause significant perspective inconsistency among selected regions/patches. An example is shown at the bottom left in Fig. 3, where the partial region is from the left viewing angles and the other regions come from the right viewing angles. One solution is to evaluate one patch at a time and choose patches with viewing directions as close to the normal of the current proxy polygon as possible [Wang et al. 2018]. However, the set of candidate photos usually has a large variation in viewing direction and extremely uneven distribution, especially for those proxy polygons with large planes. Strong front-parallel constraint could introduce missing regions while loose constraint could cause significant perspective differences. To address this issue, we present a novel quality measurement to select a set of perspective consistent images, while satisfying the front-parallel constraint as much as possible.

The perspective quality term tends to select the next candidate image with a viewing direction similar to selected images in $\{\mathcal{I}\}_i^{k-1}$,

[1]https://www.sidefx.com/

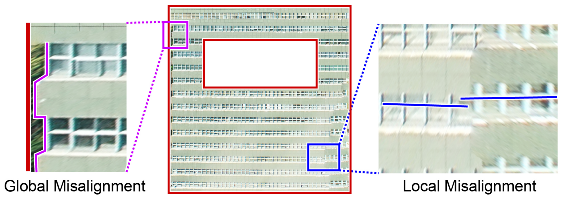


Fig. 7. Simply stitching selected images and projecting to proxy polygon produces obvious geometric inconsistency. Left: global texture-to-geometry misalignment, where the magenta textural feature lines should match the proxy geometric boundary (visualized in dark red). Right: local texture-to-texture misalignment, where the blue feature lines should be connected and collinear.

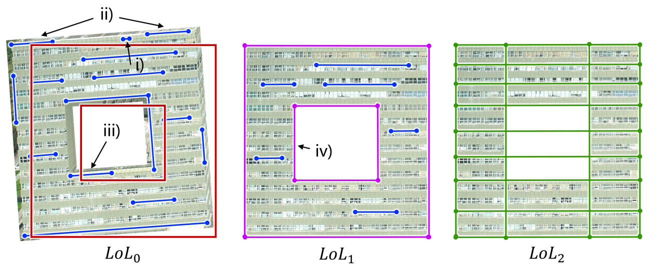


Fig. 8. Three levels of the line features extracted from a facade image from local to global (left to right). The respective proxy boundary is visualized in red. Some refinement operations i)-iv) on $LoL_1$ are pointed with arrows.

and with a viewing angle perpendicular to $\mathcal{P}_i$ as much as possible. Hence, the perspective property is measured with front-parallel constraint and viewing angle consistency against selected photos. Unlike previous methods that only consider the viewing angle of a single candidate at a time [Wang et al. 2018], we propose to select a set of candidate images with high viewing angle consistency and as many front-parallel properties. Photo collections usually have a large variety of viewing angles. In practice, it is difficult to find a set of $N$ photos from the input all with a high level of front-parallelism covering a large plane. Compared to focusing on front-parallelism, we hope to stress more on viewing direction consistency as $N$ gets larger. Hence, we introduce the inverse facade normal as the first selected viewing direction. We define the perspective quality as:

$$Q_{persp} = 1 - \frac{1}{N+1}[D_n(\mathbf{v}_j, -\mathbf{n}_i) + \sum_n D_\theta(\mathbf{v}_j, \mathbf{v}_n)], \qquad (4)$$

where $N$ is the number of selected images in $\{\mathcal{I}\}_i^{k-1}$, $\mathbf{n_i}$ is the plane normal of $\mathcal{P}_i$, $\mathbf{v}_j$ is the viewing direction of $R_j$, $\mathbf{v}_n$ is the viewing direction of the $n$-th image in $\{\mathcal{I}\}_i^{k-1}$. $D_n(\mathbf{v}, \mathbf{n}) = \frac{2}{\pi}\cos^{-1}(\mathbf{v} \cdot \mathbf{n})$ calculates the normalized angular distance between facet normal $\mathbf{v}$ and plane normal $\mathbf{n}$. $D_\theta(\mathbf{u}_1, \mathbf{u}_2) = \frac{1}{\pi}\cos^{-1}(\mathbf{u}_1 \cdot \mathbf{u}_2)$ calculates the normalized angular distance between facet normal $\mathbf{u}_1$ and $\mathbf{u}_2$.

## 5 LINE-GUIDED TEXTURE MAPPING

After determining a set of high-quality images $\{\mathcal{I}\}_i$ for each proxy polygon $\mathcal{P}_i$, our next goal is to generate a realistic and geometry-aware texture map for $\mathcal{P}_i$. Urban buildings usually exhibit rich structural information, such as local/global line features (e.g., window border/windows layout) in facades, and structural line features of the overall building (e.g., facade border). Different from most existing texturing methods that are dealing with overall smooth models, the proxy model in our case is only $C0$ continuous with obvious sharp structural features. It makes our texture mapping a challenging problem. Even tiny texture-to-texture or texture-to-geometry misalignments will draw great visual attention, which can be observed in Fig. 7.

Instead of relying on semantic label masks to synthesize textures [Sinha et al. 2008], we make full use of the rich line features extracted from candidate views and the proxy boundary extracted from the planar polygon to guide the texture mapping. Our algorithm begins by extracting three levels of line features (LoLs) from selected candidate images. A line-guided texture stitching approach is then proposed to improve the visual effects based on LoLs. Finally, we perform a texture optimization step to further improve the illumination consistency and texture completeness. To further eliminate the seams across patches, we employ graph-cut image synthesis [Boykov et al. 2001; Kwatra et al. 2003] and Poisson-Image-Editing [Pérez et al. 2003] to obtain one single enlarged texture for each proxy polygon.

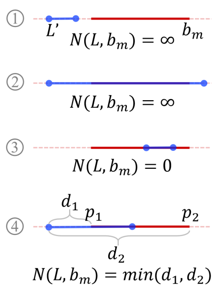
### 5.1 Levels of Line Features

To generate proper line guidance at different scales we extract and optimize line segments from the level of local to global and finally define three levels of line features (LoLs).

*$LoL_0$.* We first employ the ELSED algorithm [Suárez et al. 2022] to detect rich but discrete line segments from the projected images $\mathcal{R}$ as local features at the finest scale. Then we organize these line segments into a set of global lines defined as $LoL_0$, in which each basic element encodes a meaningful primitive, *e.g.*, facade boundary, window boundary. The $LoL_0$ is employed for rigid registration (the details will be described in Sec. 5.2). The registered images are then fed to later steps for generating $LoL_1$ and $LoL_2$.

*$LoL_1$.* After registering image $\mathcal{I}_n$ via rigid alignment, we obtain its $LoL_1$ which has higher accuracy than $LoL_0$. However, the registered line segments still cannot well represent the polygon boundary. Our next step is to obtain $LoL_1$ which can maximally match $\mathcal{B}_i$.

We first measure the partial matching relationship of a line segment $L \in LoL_1$ compared to a boundary edge $b_m \in \mathcal{B}_i$ considering: a) $\theta(L, b_m)$, the angle between two lines. b) $D_l(L, b_m)$, the maximum point-to-line distance from end-points to another line. Moreover, it is useful in matching line segments with similar slopes and spatial distance, but large variances in length. c) $N(L, b_m)$, the level of non-overlap between $L$ and $b_m$. To compute it, $L$ is first projected to $b_m$ to get $L'$. We compute $N(L, b_m)$ based on four types of relationship between $L'$ and $b_m$ after line projection, see the inset figure.



In case 1 and 2, we assign infinity to $N(L, b_m)$. In case 3, $N(L, b_m)$ is set to zero. A point projected on $b_m$ with end-points $p_1$ and $p_2$ can be denoted as $p_1 + w(p_2 - p_1)$. In the last case, for the end-point of $L^{'}$ locating outside $b_m$ ($w < 0$ or $w > 1$), we calculate its distance to the end-points of $b_m$ and assign the minimum value to $N(L, b_m)$. $L$ with value of a)-c) smaller than given thresholds will be denoted as a boundary matching line $\hat{L}$.

The set of lines of $LoL_1$ are further refined by: i) eliminating tiny noisy $\hat{L}$ matching any $b_m$, ii) merging the set of lines $\hat{L}$ matching the same $b_m$ into one single line segment, iii) extending $\hat{L_1}$ and $\hat{L_2}$ until they share a same end-point, if the matching boundary $b_p$ and $b_q$ of $\hat{L_1}$ and $\hat{L_2}$ are neighbors, and iv) connecting $\hat{L_1}$ and $\hat{L_2}$ whose matching boundary are $b_p$ and $b_q$ respectively, if the inserted $L$ can match $b_o$ whose neighbors are $b_p$ and $b_q$. The above operations are illustrated in Fig. 8 and the refined $\hat{L}$ are visualized in purple. The $LoL_1$ can serve as a guidance for texture stitching in Sec. 5.2.

*$LoL_2$.* Finally, we cluster all the line segments in $LoL_1$ to extract structural lines with a dynamic K-means algorithm, according to the slope and the distance from the image center to the line segment. After that, we can obtain $LoL_2$ to represent the structural layouts of facade components and to serve as a reference for later inpainting in Sec. 5.3. Until now, for each selected image, we obtain LoLs for different operators. Fig. 8 illustrates the LoLs from a facade image of Hitech.

## 5.2 Line-guided Texture Stitching

For each proxy polygon $\mathcal{P}_i$, we have obtained a subset of views $\{\mathcal{I}\}_i$ along with their LoLs in previous steps. For simplicity, we denote the first selected image $\mathcal{I}_1$ in $\{\mathcal{I}\}_i$ as a reference image $\mathcal{I}^{ref}$ and the others as target images $\mathcal{I}^{tar}$. To generate an enlarged texture map, we need to find the matching relationship between two images to stitch them together. Based on extracted LoLs, one solution is to find the correspondence between the lines (*e.g.,* using LBD descriptor [Zhang and Koch 2013]) by considering both image and geometry features. However, there exists plenty of repetitive patterns which make such pixel-based methods less effective. Therefore, we develop an algorithm for matching line segments by utilizing geometric information.

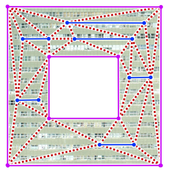
Due to the camera and geometry inaccuracy, there exists obvious texture-to-geometry displacement which increases the difficulty in accurate image local matching and stitching. We start by globally registering $\mathcal{I}^{ref}$ towards $\mathcal{B}_i$ with a rigid transformation and several warping operators. Next, following the order of their insertion into $\{\mathcal{I}\}_i$, all the $\mathcal{I}^{tar}$ are sequentially deformed to match $\mathcal{B}_i$ globally, then locally stitched to $\mathcal{I}^{ref}$. The $n$-th inserted image is denoted as $\mathcal{I}_n$. The stitching process is done in three steps: rigid alignment, line matching and texture warping.

*Rigid alignment.* To reduce the large texture-to-geometry misalignment, each image $\mathcal{I}_n$ is transformed toward the proxy boundary $\mathcal{B}_i$. Firstly, we extend each image with a margin of $N$ pixels and extract complete $LoL_0$ from the extended version. We compute the bounding box of $\mathcal{B}_i$ and set $N$ to 5% of the length of its diagonal. This can be observed in Fig. 8. Next, we convert all the endpoints of $LoL_0$ in $\mathcal{I}_n$ into a source point cloud $C_n$. We then convert the endpoints of $\mathcal{B}_i$ into a target point cloud $C_i$. A rigid transformation is calculated via ICP [Besl and McKay 1992] to align $C_n$ toward $C_i$. This rigid transformation is employed to transform and update $\mathcal{I}_n$.

*Line segment matching.* We denote $LoL_1$ of $\mathcal{I}^{tar}$ as $\mathcal{L}^{tar}$ and $\mathcal{I}^{ref}$ as $\mathcal{L}^{ref}$. For each line segment $L^{tar} \in \mathcal{L}^{tar}$, it can be registered to line segment $L^{ref} \in \mathcal{L}^{ref}$ if satisfying: i) the angular distance $\theta(L^{tar}, L^{ref})$ is smaller than $5°$, ii) the point-to-line distance $D_l(L^{tar}, L^{ref})$ is closer than 10 pixels, and iii) the level of non-overlap $N(L^{tar}, L^{ref})$ is smaller than 100 pixels if $L^{ref} \in \mathcal{B}_i$. We denote $\mathcal{L}$ as the set of all pairs of matched line segments between $\mathcal{L}^{tar}$ and $\mathcal{L}^{ref}$.

*Texture warping.* With the matched lines in $LoL_1$, we can warp $\mathcal{I}^{tar}$ to $\mathcal{I}^{ref}$ while preserving the matched line segments. The warping of $\mathcal{I}_n$ to $\mathcal{B}_i$ shares the same process. Previous work [Jia et al. 2021] converts the transformation into a global line-guided mesh deformation problem. They rely on deforming the vertices of a grid-mesh uniformly sampled in the target image to maintain line structures by solving a global least-square energy function.

However, the global optimal deformation method may not guarantee the strict straightness of colinear line features in facade images. In this work, we employ an adaptive-mesh data structure to explicitly maintain the line segments after warping. First, we construct a constrained Delaunay triangulation $M^{tar} = (V^{tar}, E^{tar}, F^{tar})$ based on $LoL_1$, keeping all the inserted segments as constraint edges, as shown in the right wrapped figure. Unlike uniformly sampling grid-mesh data structure, every basic element in our adaptive mesh encodes a meaningful geometric primitive in the target image, *i.e.,* a corner point or a line segment. Based on this, all the vertices $V^{tar}$ are deformed to new positions $\widetilde{V}^{tar}$ while satisfying the following two constraints:



- The matched line segment pairs in $\mathcal{L}$ are well aligned after the mesh deformation.
- All the line segments in $\mathcal{L}^{tar}$ preserve straightness after the mesh deformation.

We search for the optimal deformation offset $\Delta V = \widetilde{V}^{tar} - V^{tar}$ by embedding these two constraints and a normalization term into an energy minimization formulation:

$$E(\Delta V) = \lambda_a E_a(\Delta V) + \lambda_l E_l(\Delta V) + \lambda_r E_r(\Delta V). \quad (5)$$

$E_a(\Delta V)$ represents the line-alignment term, which guarantees the alignment between all pairs of matched line segments in $\mathcal{L}$ after deforming vertices by $\Delta V$ in the form of

$$E_a(\Delta V) = \sum_{t=1}^{|\mathcal{L}|} d(V_{t1}^{tar} + \Delta V_{t1}, l_t^{ref}) + d(V_{t2}^{tar} + \Delta V_{t2}, l_t^{ref}), \quad (6)$$

where $l_t^{ref}$ is the $t$-th matched line segment in $\mathcal{L}^{ref}$, and $(V_{t1}^{tar}, V_{t2}^{tar})$ are end points of $t$-th matched line segment in $\mathcal{L}^{tar}$. Moreover, the line-preserving term denoted as $E_l(\Delta V)$ ensures the straightness of each line segment in $\mathcal{L}^{tar}$ after mesh deformation by

$$E_l(\Delta V) = \sum_{j=1}^{|\mathcal{L}^{tar}|} [(V_{j1}^{tar} + \Delta V_{j1}) - (V_{j2}^{tar} + \Delta V_{j2})]^T \cdot \overrightarrow{n}_j^{tar}, \quad (7)$$

where $(V_{j1}^{tar}, V_{j2}^{tar})$ are end points of $j$-th line segment $l_j^{tar}$ in $\mathcal{L}^{tar}$ and $\overrightarrow{n}_j^{tar}$ is the normal vector of $l_j^{tar}$. The normalization term to prevent exaggerate deformation is defined as $E_r(\Delta V) = \sum_{t=1}^{|\mathcal{L}|} ||\Delta V||^2$. $\lambda_a$, $\lambda_l$, and $\lambda_r$ in Eq. 5 is set to 0.5, 0.5, and 0.025 in all experiments.

Instead of optimizing the vertex position via LSQM [Jia et al. 2021], we calculate the vertex offset by applying the conjugate gradient algorithm which is more stable and efficient. Unlike using the calculated vertex position to update enclosing grid vertices with bilinear interpolation [Jia et al. 2021], each line segment represents a linear feature of a facade component in our case. We directly apply the optimal vertex offsets to the end-points of lines for the preservation of linear features. There can appear topological issues of the updated mesh after that. Hence, we introduce a refinement step to address the topological issues such as line crossing and non-manifolds. Finally, with an affine transformation constructed by the refined vertices of adaptive mesh facet $f \in F^{tar}$, each pixel $p^{tar}$ inside the $f$ is warped to the new position $\widetilde{p}^{tar}$. With the alignment and warping operators, all the visible regions are globally warped to $\mathcal{B}_i$ and locally warped to $\mathcal{I}^{ref}$ forming an enlarged texture image for each proxy polygon $\mathcal{P}_i$.

## 5.3 Texture Optimization

*Illumination adjustment.* Previous steps enable us to generate a texture map $\mathcal{I}_i$ for each proxy polygon $\mathcal{P}_i$. Since $\mathcal{I}_i$ is composed of patches extracted from various viewing directions, the brightness may be inconsistent across neighbouring patches. Assuming that the brightness distribution over the whole texture map should hold a specific pattern, we perform a histogram specification operation to improve the illumination consistency of $\mathcal{I}_i$ in HSV color space. To be precise, we first calculate a distribution histogram of the channel $V$ on the non-overlapping region of $\mathcal{I}^{tar}$ and overlapping region of $\mathcal{I}^{ref}$, respectively. Next, we match these two histograms and transfer the brightness of the overlapping region on $\mathcal{I}^{ref}$ to the non-overlapping region on $\mathcal{I}^{tar}$.

*Texture inpainting.* In practice, some regions in $\mathcal{P}_i$ may not be observed by any of the selected views or covered with high-quality patches, leading to holes in the generated texture map. To reduce this visual artifacts, we need to fill the missing regions (referred to as masks) to generate a complete texture map. However, the arbitrary sizes and irregular shapes of the mask pose great challenges to previous inpainting methods (*e.g.,* patch-based approaches [Barnes et al. 2009; Huang et al. 2014]). Inspired by the strong capability of diffusion model in generating semantic and geometric harmonious results, we employ a denoising diffusion probabilistic model called RePaint [Lugmayr et al. 2022] for free-form inpainting.

However, RePaint is still not robust enough to generate structurally consistent results for a large missing region. To alleviate this issue, we further introduce a line-guided version of RePaint called Multi-Mask RePaint model (MMRP) to preserve the geometric consistency. As shown in Fig. 9, the input of MMRP contains the

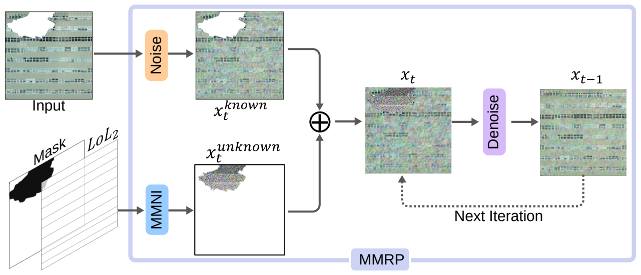


Fig. 9. Overview of our proposed texture inpainting approach, MMRP.

mask, $LoL_2$, and the texture to be completed. We then propose a multi-mask noise initialization (MMNI) component. Firstly, we compute the principal directions of the lines in $LoL_2$ by using principal component analysis (PCA). Our MMNI divides the unobserved region into several parts following the second max principal direction. As mentioned by Croitoru et al. [2023], the results will show better Frechet Inception Distance values and faster convergence with a mixture of Gaussian noises. Inspired by this observation, unlike the original RePaint that initializes the entire unobserved region with a single random Gaussian noise, we randomly initialize each divided part with a random Gaussian noise $N(\mu, \sigma)$, where $\mu$ ranges from [0, 10], $\sigma$ ranges from [1, 50]. The ranges are chosen from experimental results that deliver good performance.

After that, the parts initialized with multiple noises along with observed content are merged into one image, $x_t$, which will be harmonized together into a new inpainting. With MMNI, our MMRP can increase the possibility of generating result that is more consistent with the observed region.

We find that the quality of training data is crucial for diffusion model to generate feasible results for a specific scenario, such as the architecture in our case. To adapt RePaint to our task, we create a TwinTex dataset and fine-tune Repaint which was originally trained on ImageNet [Russakovsky et al. 2015]. Starting from $44k$ high-resolution images collected with aerial drone covering 17 outdoor architectural scenes, we first project each image to all its visible $\mathcal{P}_i$ and extract the pixels inside the bounding box of visible regions. Each extracted image is then cropped to a set of images with the size of $512 \times 512$. Next, we remove the images with heavy blurring issues or containing a large portion of non-architectural contents (*e.g.,* vegetation, human, scaffolds, etc) [Zhao et al. 2017]. Our final TwinTex dataset (TwinTexSet) contains $46k$ images of various scene components and building categories (school, office building, etc). Note that all the image regions with heavy blurring issue and extreme inclined angles have been removed from our data set. Note that there are some overlaps between the training and testing scenes. However, the TwinTex dataset contains the processed photos of 17 outdoor scenes, where we have rectified the photos, removed blurring regions and large non-architectural objects. After these processes, the training images and the test texture maps have no overlap. The missing regions in test images come from frustum-culling, visibility detection, or are not included in input photos. Therefore, the regions to be infilled never appear in the training images, thus never seen by the trained model in our tasks.

We make use of our TwinTex dataset to fine-tune the pre-trained RePaint model. The total training time is about 9 hours. We only need to pay the time cost for training once. For an image with high resolution, we will resize it to 512 × 512. And for an image with large missing area, we choose to scale it according to the largest dimension while preserving the image ratio. The resized image will be fed into MMRP to receive a semantic and geometric consistent inpainting. Finally, we upsample the resultant inpainting to the original size with a bi-linear interpolation operation. Please refer to the supplemental for more details and experimental results.

## 6 RESULTS

To validate the proposed algorithm, we conduct a series of experiments on real-world scenes with varying building styles and functions. We first demonstrate the robustness of our approach and the effectiveness of each main ingredient through a stress test. Then, we evaluate our approach by comparing it against state-of-the-art texture mapping methods. Ablation studies are performed as well to show the effectiveness of our main technical ingredients (please refer to the supplemental material for the details). Texturing results by our method and manual work on proxy models at different levels of detail are also provided in the supplemental material to demonstrate the high quality of our generated texture maps. Our algorithm is implemented in C++ and also well-organized as a customized plug-in for Houdini. All the presented experimental results are obtained on a desktop computer equipped with an Intel i9-10900k processor with 3.0 GHz and 128 GB RAM. Note that the time cost for fine-tuning the RePaint model are not included in the statistics.

*Dataset.* For the performance evaluation and comparisons, we carry out experiments on 18 real-world scenes, including 15 outdoor buildings, two indoor rooms and a man-made object. These scenes come from public sources (*i.e.,* Library [Zhang et al. 2021], Hitech [Zhou et al. 2020], Polytech and ArtSci [Lin et al. 2022]), a handheld device (*i.e.,* Machine Room, Lab, Cabinet), or are captured by a single-camera drone (*i.e.,* Highrise, School, Center, CSSE, Sunshine Plaza, Hall, Center, Factory, Hisense, Bank, Apartment). Please refer to the supplemental material for detailed statistics on the photo collections and models. Then we use the commercial software RealityCapture[2] (RC) to reconstruct the 3D high-precision models from the captured images. All the abstracted versions are either the 2.5D models for path planning [Zhou et al. 2020], or created by in-field modelers. All of the dense reconstructions and proxy models are utilized to demonstrate the performance of our algorithm.

### 6.1 Stress Test

We first evaluate the robustness of our approach on a complicated Hall example with 519 input views. This stress test is performed via randomly removing partial photos from the original photo set and feeding the rest views into our TwinTex. The textured proxies of Hall example generated with three different sets of remaining views are shown in Fig. 10. We also show the intermediate texture maps after each step, *e.g.,* view selection, image stitching along with blending and texture inpainting.

[2]https://www.capturingreality.com/

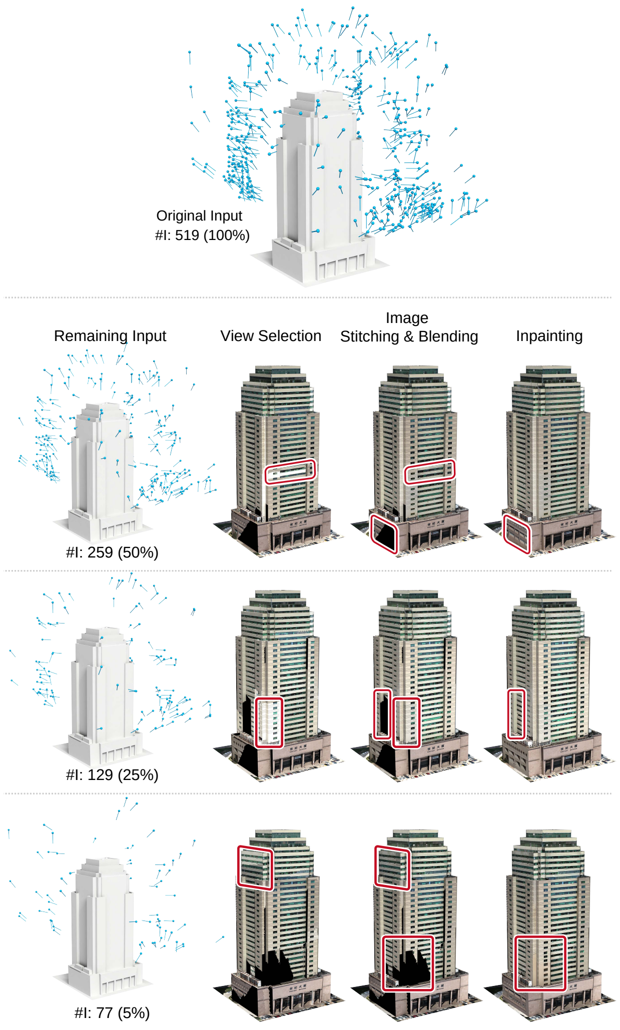


Fig. 10. Stress test of our methodology on Hall example via randomly removing a certain percentage of photos from original input. In each row we visualize: the camera distribution of the remaining input, the textured proxy based in remaining input after view selection, image stitching with blending, and inpainting. $\#I$ denotes the number of photos. The value in the bracket denotes the percentage of the remaining photos related to original input.

We randomly removed 50%, 75% and 95% photos from the original set and made use of the remaining views to texture this proxy model. The textured proxies after view selection are generated by simply overlapping the projection of selected images following the reverse order of selection. As the number of photos in the remaining input decreased, our selection algorithm can still select high-quality photos from the input to maximally cover the proxy, as illustrated in the second column of Fig. 10. From the third column, we can see the texture maps after performing our stitching step and illumination adjustment exhibit higher geometric and photometric consistency. However, there appears larger missing regions as the number of remaining views decrease. We only use 77 images to texture Hall example with 225 facades (each plane has about 6 photos to select

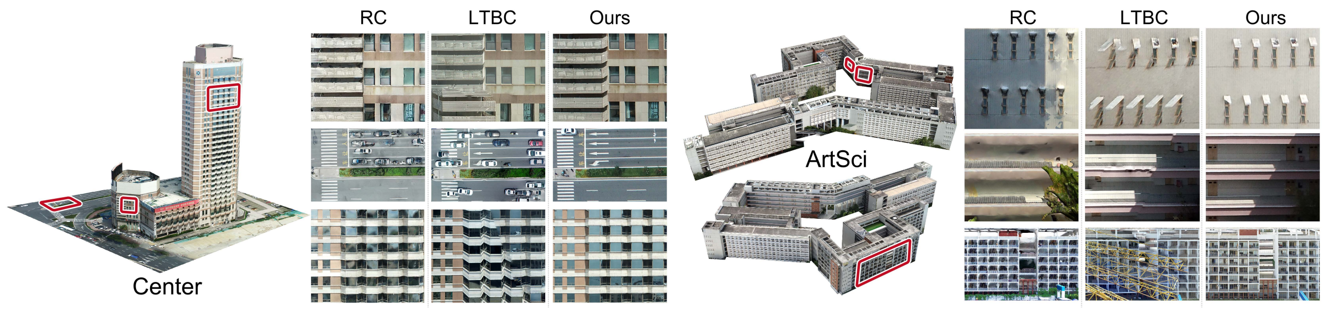


Fig. 11. Visual comparison against RC, LTBC and ours on two examples. The detailed comparisons are shown in the zoomed-in insets. Due to the page limit, we only show the overall views of our textured proxy models, which are shown on the left to the zoomed-in insets.

from). All the missing regions are visualized with black pixels, as marked by the red boxes in the figure. The final texture maps visualized in the last column of Fig. 10 demonstrate the ability of our inpainting framework for creating geometrically and semantically consistent contents under very limited resource.

## 6.2 Comparison with Texturing Methods

In this subsection, we evaluate the quality of our generated texture maps by comparing them against RC and two competitive texturing generation frameworks, let there be color (LTBC) [Waechter et al. 2014] and patch based optimization (PBO) [Bi et al. 2017]. We adopt the evaluation scheme proposed by Waechter et al. [2017] to compare the rendered image with the corresponding real image following a specific view from the input cameras. We select the ground truth photos for evaluation from the input excluding the photos for texturing. Two visual similarity metrics, namely the structural similarity index measure (SSIM) and learned perceptual image patch similarity (LPIPS) [Kastryulin et al. 2022], are adopted for quantitative evaluation.

**Comparison with LTBC.** We first compare our method with RC and LTBC on two examples. Note that LTBC selects the most suitable view for each facet by solving a Markov Random Field formulation and adjusting the pixel values along the boundary edges of adjacent patches to alleviate the seams. This strategy fails to texture large facets when there is no camera view observing the whole region of the facet. Thus, we subdivide the simplified coarse model and perform LTBC to generate texture maps for the corresponding subdivided mesh for a reasonable comparison.

Fig. 11 shows the qualitative results on two abstracted models with sharp textures and structural features. First, we can observe that LTBC and RC are not able to produce satisfactory results for large-scale planar structures which are invisible for the input photos or with only low-quality pixels. In these cases, their texture maps show blurring and ghosting artifacts. By contrast, our method can fill large unobserved regions with geometrically and semantically consistent pixels.

Second, without the constraints on perspective consistency in view selection, LTBC failed in selecting photos with both consistent and front-parallel viewing directions, creating "stretching" alike artifact and perspective inconsistency in the generated textures. The "stretching" problem can be observed in the inset of the ArtSci building in Fig. 11 (first row, second column), which was caused by merging photos with large viewing angle variance. The perspective inconsistency can be observed in the insets of the Center example in Fig. 11 (first and third rows, second column). Unlike LTBC, our view selection strategy can pick out a very small set of perspective consistent and front-parallel photos as possible. Moreover, we introduced a smooth term considering the pixel-wise distance of each image pair. This helps select a set of photos with a higher level of photometric and content consistency, especially for filtering out

Table 1. Quantitative comparison on the texturing results in Fig. 11.

| Scene | Method | *SSIM* ↑ | *LPIPS* ↓ | Time (min) |
|---|---|---|---|---|
| Center | RC | 0.350 | 0.831 | 287.0 |
| | LTBC | 0.354 | 0.836 | 4.4 |
| | Ours | **0.358** | **0.831** | 167.4 |
| Artsci | RC | 0.346 | 0.883 | 320.9 |
| | LTBC | 0.331 | **0.873** | 6.0 |
| | Ours | **0.348** | 0.877 | 256.1 |

Table 2. Quantitative comparison on the performance of texturing results on the third and fourth examples in Fig. 12. Numbers in the bracket denote the quality value of the three zoomed-in views of each example from top to bottom. All the times are in minutes.

| Scene | Method | *SSIM* ↑ | *LPIPS* ↓ | Time |
|---|---|---|---|---|
| Library | RC | { 0.28, 0.28, 0.32 } | { 0.53, 0.60, 0.68 } | 383.6 |
| | LTBC | { 0.29, 0.29, 0.34 } | { 0.54, 0.68, 0.68 } | 4.8 |
| | PBO | { **0.63**, **0.45**, 0.38 } | { 0.63, 0.67, 0.69 } | 810.7 |
| | Ours | { 0.46, 0.38, **0.36** } | { **0.28**, **0.52**, **0.66** } | 200.4 |
| Bank | RC | { 0.25, 0.62, 0.26 } | { 0.43, 0.23, 0.54 } | 198.3 |
| | LTBC | { 0.20, 0.52, **0.33** } | { 0.51, 0.37, 0.53 } | 6.4 |
| | PBO | { 0.25, **0.70**, 0.27 } | { 0.43, 0.31, 0.57 } | 672.1 |
| | Ours | { **0.26**, 0.63, 0.28 } | { **0.27**, **0.18**, **0.52** } | 174.5 |

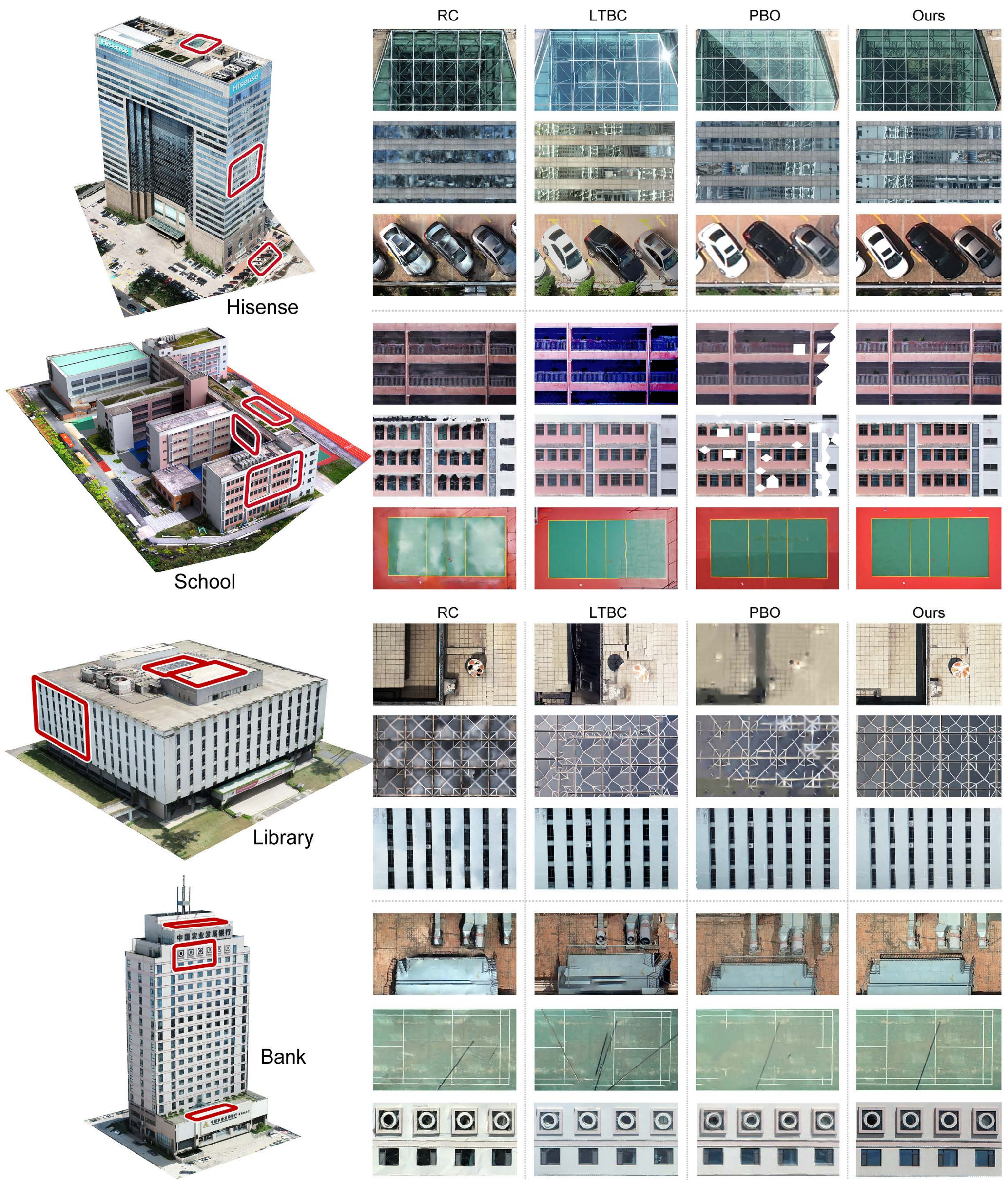


Fig. 12. Visual comparison against RC, LTBC, PBO and ours on four examples. The detailed comparisons are shown in the zoomed-in insets.

occluders or dynamic instances (e.g., cars). The significant improvement can be observed in the Center example in Fig. 11 (second row) and ArtSci example in Fig. 11 (third row).

Furthermore, due to the lack of explicit constraints on the linear features of buildings, the texture generated by LTBC destroys geometric structures, making them suffer from serious seams and distortion effects. Our image stitching mechanism succeeds in preserving the straightness and completeness of the line segments by constructing an adaptive mesh taking the original geometric primitives as constraints. Next, large and duplicate texture patterns are also well maintained in our results. This is mainly because we select a suitable small subset of camera views for each planar shape instead of each facet. This strategy guarantees that each selected view is used to texture the current plane as much as possible.

The quantitative comparison results are given in Table 1. In most of the cases, the rendered views of our textured model can deliver better SSIM and LPIPS scores. Please refer to the supplemental material for the overall views of the above two examples.

**Comparison with PBO.** Next, we perform comparison with PBO[3] and report SSIM and LPIPS as well on four more architectural scenes. We first tried to globally compare the similarity of

[3]https://github.com/yanqingan/EAGLE-TextureMapping

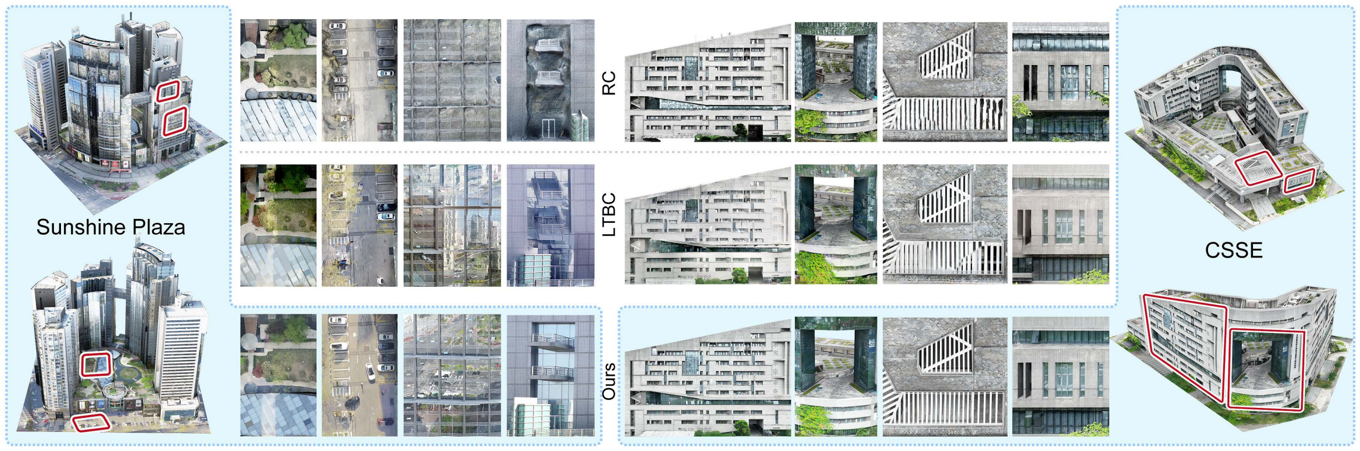


Fig. 13. Visual comparison of two challenging cases against RC, LTBC and ours. The detailed comparisons are shown as zoomed-in insets.

rendering views on the entire building with real photos. However, PBO requires an extremely large amount of time/storage given just several high-resolution images as input. Remember that our target is to retain the quality of the inputs, one way to address the issues is to decrease the number of input photos. This can produce other problems such as missing contents since no pixels can cover some of the regions. It is crucial to provide key views with high quality to the PBO in this scenario. Hence, we locally perform the comparison to PBO using three planes per example. To this end, we fed PBO with the set of views selected by our algorithm for the planar polygons involved in the comparison. Considering the nature of patch-based optimization, for a fair qualitative comparison, we use the subdivided mesh prepared for LTBC as the input to PBO as well. Fig. 12 shows the texturing results of RC, LTBC, PBO and ours. The comparison shows that given the same set of selected views, PBO can generate a texture with good quality in most cases which also proves the effectiveness of our view selection algorithm. However, the results of PBO have obvious blurring issue and line distortion artifact compared to the real photos. There are also some optimization failures of PBO which cause missing textures for the subdivided triangles. In comparison, our stitching and optimization frameworks can preserve clear line features and retain the original structure while we are able to inpaint missing regions.

Quantitative results are listed in Table 2. Since SSIM is by nature less sensitive to blurring issues and possibly gives higher scores to images with such artifacts [Zhang et al. 2018], the texture with the regional blurring problem could have higher scores. This phenomenon can be observed in some views per each example. Meanwhile, LPIPS is designed to match human perception and yield better scores for images with higher level of coherence. Note that the recorded time cost for PBO does not include view selection or is not for texturing the entire proxy although it is relatively long.

## 6.3 Analysis of Generalization Ability

In this subsection, we discuss the results of our method applied to more diverse scenes. First, we conduct experiments on several challenging urban scenes that contain curve surfaces, facades with

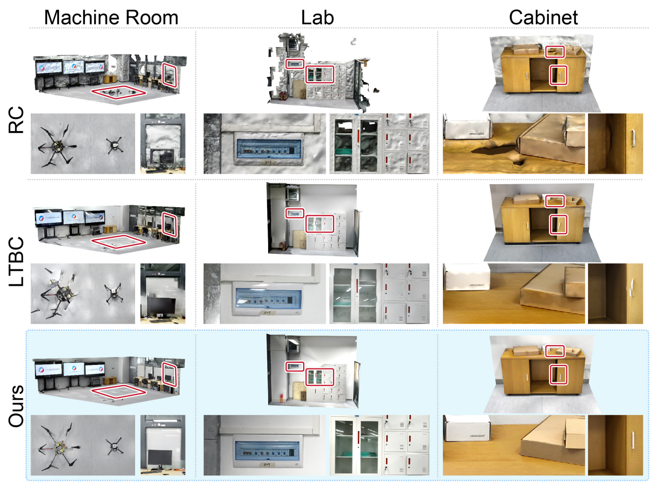


Fig. 14. Visual comparison against RC, LTBC and ours on two indoor scenes and a cabinet.

non-linear features, and facades with a lot of reflection, as shown in Fig. 13. Then we present the texturing results on two indoor scenes and one man-made object in Fig. 14, which are different scenarios from buildings. Note there exists only one fire extinguisher in the Lab scene. Although it has been moved during the photos collection process, our method still perfectly keeps its location where it has been placed in most photos.

To better understand the superiority of our TwinTex, we conduct comparisons with previous methods as well. From the figures we can see that our approach can still generate high-fidelity textures for the challenging buildings, and can be naturally extended to texture coarse piece-wise planar models in other scenarios. The visual comparisons further suggest that the textures generated by our approach have several significant advantages over previous methods: i) closer to real photos with perspective consistency and harmonic illumination, ii) contain rarely border misalignment, distortion, blurring and

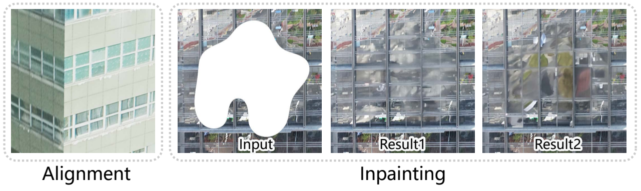


Fig. 15. Failure examples: inter-planar misalignment among two adjacent facades (left), and inpainting results of a reflective facade with large glass panels (right).

seaming artifacts resulting from inaccurate camera parameters, iii) contain rarely ghosting effects resulting from dynamic instances, iv) preserve geometric details better, and v) fill the missing regions with geometrically and semantically consistent contents.

## 7 CONCLUSION AND FUTURE WORK

In this work, we proposed a plane-based and geometry-aware texture generation method. Our target is to produce high-resolution texture maps for piece-wise planar architectural proxy models which were abstracted from dense 3D reconstructions. Our main technical contributions consist in: i) A greedy algorithm to select a set of views that are most suitable for texturing each planar shape; Both perspective and photometric qualities are considered in this algorithm. ii) An image stitching methodology that can stitch the selected views to an enlarged texture map, preserving the local and global linear geometric primitives based on an adaptive-mesh data structure; iii) An improved diffusion probabilistic model that fine-tuned with our created TwinTexSet to inpaint unobserved texture region with semantically and geometrically coherent pixels; iv) A customized texturing tool plugged in a widely used graphical engine. Experimental results show that our algorithm outperforms state-of-the-art facet-based texturing methods in the generation of realistic and high-resolution texture maps in a reasonable time.

**Limitations and future work.** Our system still has several limitations. First, although we significantly refined the extracted line segments, the line-guided scheme still relies on the quality of the line extraction algorithm. One possible solution is to consider camera information as well or make use of reconstructed 3D lines. Second, our plane-based texture generation method processes each planar shape independently, which does not take their mutual relationship into consideration. Fig. 15 (left) shows such an example where global alignment of linear features between adjacent planes is not achieved. Note we here choose a zoom-in view with the most obvious misalignment among given buildings to clearly show this issue. The other examples may also exhibit tiny misalignments. Third, we may fail to successfully perform inpainting for a reflected facade which lacks of coherent structures and clear content, making it even hard for a human to infer the missing region, see Fig. 15 (right). Finally, the processing time can be further improved with parallel processing. We will incorporate these issues in our future pipeline.

## ACKNOWLEDGMENTS

We thank the reviewers for their constructive comments. This work was supported in parts by NSFC (U21B2023, U2001206, U22B2034, 62172416, 62302313), DEGP Innovation Team (2022KCXTD025), and Shenzhen Science and Technology Program (KQTD20210811090044003, RCJC20200714114435012, JCYJ20210324120213036).

## REFERENCES

Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. 2022. NeRF-Tex: Neural Reflectance Field Textures. *Comp. Graph. Forum* 41, 6 (2022), 287–301.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.* 28, 3 (2009), 24:1–24:11.

Jean-Philippe Bauchet and Florent Lafarge. 2020. Kinetic shape reconstruction. *ACM Trans. Graph.* 39, 5 (2020), 156:1–156:14.

Fausto Bernardini, Ioana M. Martin, and Holly Rushmeier. 2001. High-quality texture reconstruction from multiple scans. *IEEE Trans. on Vis. and Comp. Graph.* 7, 4 (2001), 318–332.

Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, Vol. 1611. 586–606.

Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2017. Patch-based optimization for image-based texture mapping. *ACM Trans. Graph. (SIGGRAPH)* 36, 4 (2017), 106:1–106:11.

Vasileios Bouzas, Hugo Ledoux, and Liangliang Nan. 2020. Structure-aware building mesh polygonization. *ISPRS Journal of Photogrammetry and Remote Sensing* 167 (2020), 432–442.

Yuri Boykov, Olga Veksler, and Ramin Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* 23, 11 (2001), 1222–1239.

Matthew Brown and David G Lowe. 2007. Automatic panoramic image stitching using invariant features. *Int. Journal of Computer Vision* 74, 1 (2007), 59–73.

Marco Callieri, Paolo Cignoni, Massimiliano Corsini, and Roberto Scopigno. 2008. Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3D models. *Computers & Graphics* 32, 4 (2008), 464–473.

Duygu Ceylan, Niloy J Mitra, Hao Li, Thibaut Weise, and Mark Pauly. 2012. Factored facade acquisition using symmetric line arrangements. *Comp. Graph. Forum (Proc. EUROGRAPHICS)* 31, 2pt3 (2012), 671–680.

Duygu Ceylan, Niloy J Mitra, Youyi Zheng, and Mark Pauly. 2014. Coupled structure-from-motion and 3D symmetry detection for urban facades. *ACM Trans. Graph.* 33, 1 (2014), 1–15.

Che-Han Chang, Yoichi Sato, and Yung-Yu Chuang. 2014. Shape-preserving half-projective warps for image stitching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 3254–3261.

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. In *Proc. ACM SIGGRAPH*. 905–914.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. 2023. Diffusion models in vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* (2023).

Paul Debevec, Yizhou Yu, and George Borshukov. 1998. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Workshop on Rendering Techniques*. 105–116.

Paul E Debevec, Camillo J Taylor, and Jitendra Malik. 1996. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proc. ACM SIGGRAPH*. 11–20.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Proc. International Conference on Neural Information Processing Systems* 34 (2021), 8780–8794.

Omar Elharrouss, Noor Almaadeed, Somaya Al-Maadeed, and Younes Akbari. 2020. Image Inpainting: A Review. *Neural Process. Letters* 51, 2 (2020), 2007–2028.

Hao Fang and Florent Lafarge. 2020. Connect-and-Slice: an hybrid approach for reconstructing 3D objects. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 13490–13498.

Yanping Fu, Qingan Yan, Jie Liao, and Chunxia Xiao. 2020. Joint texture and geometry optimization for RGB-D reconstruction. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 5950–5959.

Yanping Fu, Qingan Yan, Long Yang, Jie Liao, and Chunxia Xiao. 2018. Texture mapping for 3d reconstruction with RGB-D sensor. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 4645–4653.

Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. 2010. Seamless montage for texturing models. *Comp. Graph. Forum* 29, 2 (2010), 479–486.

Junhong Gao, Seon Joo Kim, and Michael S Brown. 2011. Constructing image panoramas using dual-homography warping. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 49–56.

Ignacio Garcia-Dorado, Ilke Demir, and Daniel G Aliaga. 2013. Automatic urban modeling using volumetric reconstruction with surface graph cuts. *Computers & Graphics* 37, 7 (2013), 896–910.

Michael Garland and Paul S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proc. ACM SIGGRAPH*. 209–216.

Yiangos Georgiou, Melinos Averkiou, Tom Kelly, and Evangelos Kalogerakis. 2021. Projective Urban Texturing. In *International Conference on 3D Vision (3DV)*. 1034–1043.

Christine Guillemot and Olivier Le Meur. 2014. Image Inpainting: Overview and Recent Advances. *IEEE Signal Processing Magazine* 31, 1 (2014), 127–144.

Jianwei Guo, Yanchao Liu, Xin Song, Haoyu Liu, Xiaopeng Zhang, and Zhanglin Cheng. 2022. Line-Based 3D Building Abstraction and Polygonal Surface Reconstruction From Images. *IEEE Trans. on Vis. and Comp. Graph.* (2022), 1–15.

Jingwei Huang, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 2017. 3Dlite: towards commodity 3D scanning for content creation. *ACM Trans. Graph. (SIGGRAPH Asia)* 36, 6 (2017), 203:1–203:14.

Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. 2014. Image completion using planar structure guidance. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014), 129:1–129:10.

Jireh Jam, Connah Kendrick, Kevin Walker, Vincent Drouard, Jison Gee-Sern Hsu, and Moi Hoon Yap. 2021. A comprehensive review of past and present image inpainting methods. *Computer vision and image understanding* 203 (2021), 103147.

Qi Jia, ZhengJun Li, Xin Fan, Haotian Zhao, Shiyu Teng, Xinchen Ye, and Longin Jan Latecki. 2021. Leveraging line-point consistence to preserve structures for wide parallax image stitching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 12186–12195.

Sergey Kastryulin, Jamil Zakirov, Denis Prokopenko, and Dmitry V Dylov. 2022. PyTorch Image Quality: Metrics for Image Quality Assessment. *arXiv preprint arXiv:2208.14818* (2022).

Tom Kelly, Paul Guerrero, Anthony Steed, Peter Wonka, and Niloy J. Mitra. 2018. FrankenGAN: Guided Detail Synthesis for Building Mass Models Using Style-Synchonized GANs. *ACM Trans. Graph. (SIGGRAPH Asia)* 37, 6 (2018), 216:1–216:14.

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph.* 22, 3 (2003), 277–286.

Kyu-Yul Lee and Jae-Young Sim. 2020. Warping residual based image stitching for large parallax. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 8198–8206.

Victor Lempitsky and Denis Ivanov. 2007. Seamless mosaicing of image-based texture maps. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 1–6.

Nan Li, Yifang Xu, and Chao Wang. 2017. Quasi-homography warps in image stitching. *IEEE Trans. Multimedia* 20, 6 (2017), 1365–1375.

Shiwei Li, Lu Yuan, Jian Sun, and Long Quan. 2015. Dual-feature warping-based motion model estimation. In *IEEE International Conference on Computer Vision (ICCV)*. 4283–4291.

Tianli Liao and Nan Li. 2019. Single-perspective warps in natural image stitching. *IEEE Trans. Image Process.* 29 (2019), 724–735.

Chung-Ching Lin, Sharathchandra U Pankanti, Karthikeyan Natesan Ramamurthy, and Aleksandr Y Aravkin. 2015. Adaptive as-natural-as-possible image stitching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 1155–1163.

Liqiang Lin, Yilin Liu, Yue Hu, Xingguang Yan, Ke Xie, and Hui Huang. 2022. Capturing, Reconstructing, and Simulating: the UrbanScene3D Dataset. In *European Conference on Computer Vision (ECCV)*. 93–109.

Wen-Yan Lin, Siying Liu, Yasuyuki Matsushita, Tian-Tsong Ng, and Loong-Fah Cheong. 2011. Smoothly varying affine stitching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 345–352.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. 2022. Repaint: Inpainting using denoising diffusion probabilistic models. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 11461–11471.

Robert Maier, Kihwan Kim, Daniel Cremers, Jan Kautz, and Matthias Nießner. 2017. Intrinsic3D: High-quality 3D reconstruction by joint appearance and geometry optimization with spatially-varying lighting. In *IEEE International Conference on Computer Vision (ICCV)*. 3114–3122.

Jalpa D Mehta and SG Bhirud. 2011. Image stitching techniques. In *Thinkquest˜2010: Proceedings of the First International Conference on Contours of Computing Technology*. 74–80.

Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. 2022. Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures. *arXiv preprint arXiv:2211.07600* (2022).

Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. 2015. RAPter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 103:1–103:12.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph. (SIGGRAPH)* 41, 4 (2022), 102:1–102:15.

Przemyslaw Musialski, Christian Luksch, Michael Schwärzler, Matthias Buchetics, Stefan Maierhofer, and Werner Purgathofer. 2010. Interactive Multi-View Facade Image Editing.. In *Vision, Modeling, and Visualization*. 131–138.

Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, Luc Van Gool, and Werner Purgathofer. 2013. A survey of urban reconstruction. *Comp. Graph. Forum* 32, 6 (2013), 146–177.

Liangliang Nan and Peter Wonka. 2017. PolyFit: Polygonal surface reconstruction from point clouds. In *IEEE International Conference on Computer Vision (ICCV)*. 2353–2361.

Shanshan Pan, Jiahui Lv, Hao Fang, and Hui Huang. 2022. Efficient and robust 3D structure-aware reconstruction. *Journal of Image and Graphics* 27, 2 (2022), 421–434.

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. In *Proc. ACM SIGGRAPH*. 313–318.

David Josué Barrientos Rojas, Bruno José Torres Fernandes, and Sergio Murilo Maciel Fernandes. 2020. A Review on Image Inpainting Techniques and Datasets. In *SIBGRAPI Conference on Graphics, Patterns and Images*. 240–247.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *Int. Journal of Computer Vision* 115, 3 (2015), 211–252.

David Salinas, Florent Lafarge, and Pierre Alliez. 2015. Structure-aware mesh decimation. *Comp. Graph. Forum* 34, 6 (2015), 211–227.

Sudipta N Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. 2008. Interactive 3D architectural modeling from unordered photo collections. *ACM Trans. Graph.* 27, 5 (2008), 159:1–159:10.

Neil Smith, Nils Moehrle, Michael Goesele, and Wolfgang Heidrich. 2018. Aerial Path Planning for Urban Scene Reconstruction: A Continuous Optimization Method and Benchmark. *ACM Trans. Graph. (SIGGRAPH Asia)* 37, 6 (2018), 183:1–183:15.

Iago Suárez, José M Buenaposada, and Luis Baumela. 2022. ELSED: Enhanced line SEgment drawing. *Pattern Recognition* 127 (2022), 108619.

Richard Szeliski et al. 2007. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision* 2, 1 (2007), 1–104.

Yannick Verdie, Florent Lafarge, and Pierre Alliez. 2015. LOD generation for urban scenes. *ACM Trans. Graph.* 34, 3 (2015), 30:1–30:14.

Michael Waechter, Mate Beljan, Simon Fuhrmann, Nils Moehrle, Johannes Kopf, and Michael Goesele. 2017. Virtual Rephotography: Novel View Prediction Error for 3D Reconstruction. *ACM Trans. Graph.* 36, 1, Article 8 (2017), 11 pages.

Michael Waechter, Nils Moehrle, and Michael Goesele. 2014. Let there be color! Large-scale texturing of 3D reconstructions. In *European Conference on Computer Vision (ECCV)*. 836–850.

Bin Wang, Pan Pan, Qinjie Xiao, Likang Luo, Xiaofeng Ren, Rong Jin, and Xiaogang Jin. 2018. Seamless Color Mapping for 3D Reconstruction with Consumer-Grade Scanning Devices. In *European Conference on Computer Vision (ECCV) Workshops*. 633–648.

Chao Wang and Xiaohu Guo. 2018. Plane-based optimization of geometry and texture for RGB-D reconstruction of indoor scenes. In *International Conference on 3D Vision (3DV)*. 533–541.

Tian-Zhu Xiang, Gui-Song Xia, Xiang Bai, and Liangpei Zhang. 2018. Image stitching by line-guided local warping with global similarity constraint. *Pattern recognition* 83 (2018), 481–497.

Cem Yuksel, Sylvain Lefebvre, and Marco Tarini. 2019. Rethinking texture mapping. *Comp. Graph. Forum* 38, 2 (2019), 535–551.

Julio Zaragoza, Tat-Jun Chin, Michael S Brown, and David Suter. 2013. As-projective-as-possible image stitching with moving DLT. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 2339–2346.

Guofeng Zhang, Yi He, Weifeng Chen, Jiaya Jia, and Hujun Bao. 2016. Multi-viewpoint panorama construction with wide-baseline images. *IEEE Trans. Image Process.* 25, 7 (2016), 3099–3111.

Han Zhang, Yucong Yao, Ke Xie, Chi-Wing Fu, Hao Zhang, and Hui Huang. 2021. Continuous Aerial Path Planning for 3D Urban Scene Reconstruction. *ACM Trans. Graph. (SIGGRAPH Asia)* 40, 6 (2021), 225:1–225:15.

Lilian Zhang and Reinhard Koch. 2013. An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation* 24, 7 (2013), 794–805.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 586–595.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 2881–2890.

Qian-Yi Zhou and Vladlen Koltun. 2014. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014), 155:1–155:10.

Xiaohui Zhou, Ke Xie, Kai Huang, Yilin Liu, Yang Zhou, Minglun Gong, and Hui Huang. 2020. Offsite Aerial Path Planning for Efficient Urban Scene Reconstruction. *ACM Trans. Graph. (SIGGRAPH Asia)* 39, 6 (2020), 192:1–192:16.