Template Assembly for Detailed Urban Reconstruction (Supplemental Material 1)

Liangliang Nan, Caigui Jiang, Bernard Ghanem, and Peter Wonka

KAUST, KSA

In this document, we give more detailed descriptions of our coarse model construction and 3D template construction.

1) Coarse Model Construction

The input of our framework is a set of ground level images. From the image sequence, we first extract a 3D point cloud using an efficient and popular version of Structure from Motion that uses Multi-View Stereo to produce a denser colored point cloud [1, 2]. In any SfM module, the input images are matched locally using low-level features (e.g. SIFT) and the camera's intrinsic (e.g. focal length and radial distortion) and extrinsic (e.g. camera-to-camera rotation and translation) parameters are estimated at each image so as to reinforce the local matches. These parameters are usually estimated in a nonlinear least squares manner. In this work, since we have access to the imaging device, we pre-calibrate the camera before using it in the field, i.e. the camera's intrinsic parameters are assumed to known beforehand. This calibration step leads to a much more accurate point cloud, since the SfM module will only need to estimate a much smaller number of variables (i.e. only the extrinsic parameters) in this case. Once we estimate the intrinsic and extrinsic camera parameters in the sequence, we employ a stage of multi-view stereo, which fills in gaps inherent to the preliminary sparse point cloud. Figure 1 shows the input images and the point cloud reconstructed for the teaser example of our paper.

Figure 1: From a set of ground level images (top), we reconstructed a point cloud (bottom) that is used in the teaser example of our paper.

Next, we detect dominant planar primitives within the dense point cloud. These primitives will usually define the boundaries of the coarse geometric model. Planar primitives are extracted by using an iterative and robust plane fitting method based on RANSAC, which finds plane segments [3]. Using these planar primitives as guidelines, the user then extrudes boxes in 3D to properly fit the extracted plane segments and the rest of the point cloud. These boxes form a coarse model that is used as the input to our geometry-appearance consistent optimization. Please also refer to the accompanying video.

2) 3D Template Construction

Template Initialization In this work, a 3D template is a single or compound group of basic primitives (e.g. boxes and prisms). 3D templates are constructed from the generated 3D point cloud and underlying image textures. To model different structures, we define two types of template primitives that trigger different operations on the faces of the coarse model.

- A *positive primitive* is a box or prism that is extruded outward from the facade plane (Figure 2 top).
- A *negative primitive* is a box or prism that is pushed inward from the facade plane, thus, triggering a *hole* operation on the facade plane (Figure 2 bottom).

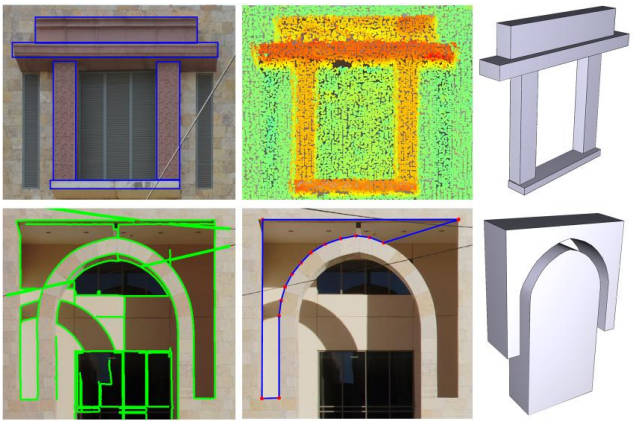

Figure 2: Constructing detailed 3D templates from images and corresponding points. Top shows the construction of a positive template (right) from user sketches (left) and depth information (middle) inferred from the 3D point cloud. Bottom shows a negative template (right) constructed from user sketched polylines (middle). The depth of the negative template is set manually due to missing depth information caused by occlusion.

The dense nature of 2D photographs allows us to efficiently extract 2D contours, thus, alleviating much tedious manual work that the user would have to do otherwise. First, contours are detected in the photographs. Then, 2D profiles of 3D primitives are constructed using our interactive tool and projected onto the underlying 3D plane of the facade using the estimated camera parameters. Next, the user sketches on photographs and extracts two types of primitives with minimal interaction.

- Simple boxes (Figure 2 top). Inspired by the idea of SmartBoxes[4], we allow the user to specify the extent of a rubber-band rectangle that loosely fits the region. The rectangle is then snapped to the nearest contours. We allow the user to modify the rectangle if some parts of the contours are missing.
- General prisms (Figure 2 bottom). The user clicks repeatedly on the image to delineate the polylines of the primitive. These clicks are automatically snapped to their nearest (within 10 pixels) contours. Our interactive tool tracks continuous curves until an end point or a branch is reached.

During the above sketching process, we compute depth information by collecting 3D points, whose 2D projections reside within the user-sketched region. For a positive primitive, the depth information is usually available from the point cloud. However, when it is unavailable due to

occlusions (especially for negative primitives, e.g. the template in Figure 2 bottom), we allow the user to manually set the depth for the primitives. Figure 3 shows a set of detailed 3D templates constructed using our interactive tool.

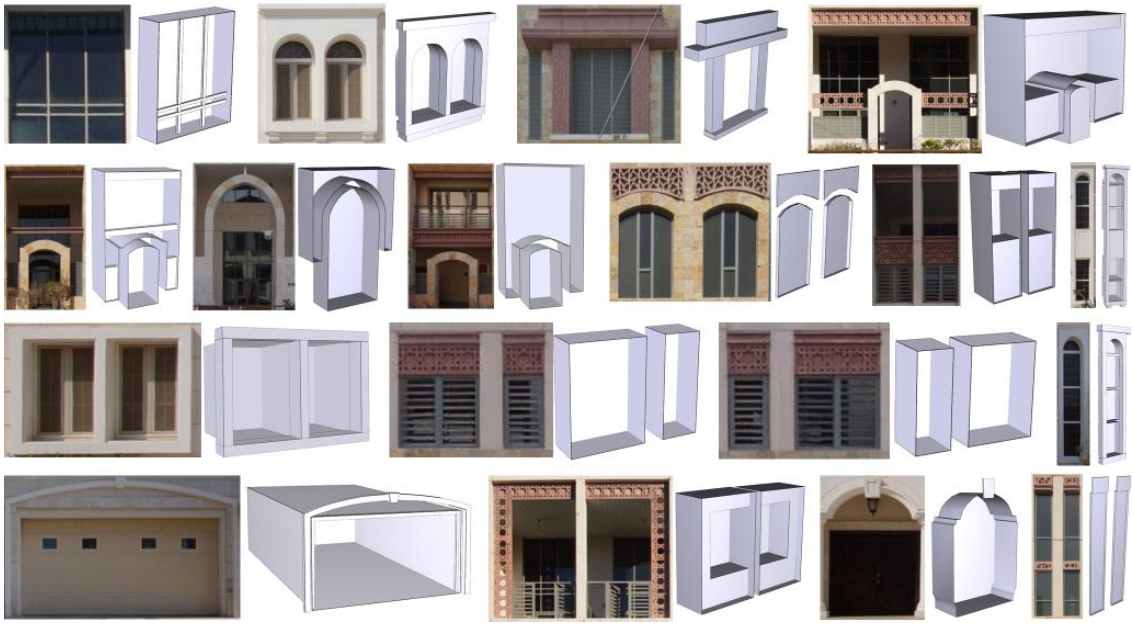

Figure 3: A set of detailed 3D templates created using our interactive template construction tool. The corresponding query key image is shown on the left of each 3D template.

Optimization Our template construction tool allows the user to loosely define geometric primitives; however, inconsistencies (e.g. intersections and gaps) may exist in the 3D template. To mitigate these inconsistencies, we refine the geometric primitives with an optimization similar to that described in Section 4 (Coarse model Optimization) for coarse models. The geometry-appearance consistency term *C(M)* is disabled during this optimization. Furthermore, since the template is not necessarily a union of boxes, the facet constraint term *F(M)* is only enforced on facets that do exhibit co-planarity or orthogonality (by simple thresholds).

Query Key Image We associate each template to the image region that was edited by the user during the template construction stage. This image region is considered a query key to be used during the subsequent template assembly step.

References

[1] Wu C.: VisualSFM: A Visual Structure from Motion System. http://ccwu.me/vsfm/.

[2] Wu C. *et al*. Multicore Bundle Adjustment. CVPR, 3057–3064, 2011.

[3] Schnabel *et al*. Efficient RANSAC for Point-Cloud Shape Detection. Computer Graphics Forum. 26(2), 214-226, 2007.

[4] Nan *et al.* Smartboxes for Interactive Urban Reconstruction. SIGGRAPH 2010, pp. 93:1–93:10.