Supplementary Material

August 13, 2015

1 Template Deformation

For a given template T_j and an element s_i , our goal is to compute the deformation parameters, \mathbf{d}_j^i , that define the instance of T_j that match s_i as closely as possible. To achieve this goal, we first apply a similarity transformation to T_j that aligns the bounding boxes of T_j and s_i . We axis-align all templates in the pre-processing stage so that the *y*-axis corresponds to the up direction. We compute correspondences between the axes of the element and template bounding boxes by matching their up directions.

Once T_j and s_i are roughly aligned we setup an optimization to compute \mathbf{d}_j^i . The goal of this optimization is to compute the template parameters that minimize the distance between T_j and s_i . In our evaluations, we define this to be the point-to-point correspondence distance. Thus, we sample 3D points both on T_j and s_i to produce the sample point sets Q_j and P_i respectively. We establish 3D correspondences between these point sets by selecting the closest points and solve for \mathbf{d}_j^i by minimizing the following energy:

$$E_{fit}(T_j, \mathbf{d}_j^i, s_i) = c(\mathbf{d}_j^i) = \sum_{\mathbf{q}_j \in Q_j} \|\mathbf{q}_j - \mathbf{p}_i\|^2 + \sum_{\mathbf{p'}_i \in P_i} \|\mathbf{p'}_i - \mathbf{q'}_j\|^2.$$
(1)

The first term measures the distance from the template to the element where \mathbf{q}_j and \mathbf{p}_i are corresponding points in Q_j and P_i respectively. The second term measures the symmetric distance where $\mathbf{p}'_i \in P_i$ and $\mathbf{q}'_j \in Q_j$ denote the correspondences. We solve this optimization iteratively, updating Q_j and the correspondences at each iteration. We use the Ipopt package [WB06] to solve the non-linear optimization at each iteration.

i-Wires deformation model. When fitting a template T_j equipped with the i-Wires deformation model to an element s_i , our goal is to align the dominant linear features of T_j and s_i . Therefore, for the input MVS acquisition we perform standard image-space edge detection on each individual image and apply multiview stereo matching on the 2D edges [BSZF99] to generate a set of 3D line features. We then sample 3D points on the feature wires of T_j and the 3D line segments falling inside the bounding box of s_i to generate the point sets Q_j and P_i respectively. In order to preserve the structural relations detected among

the feature wires of T_j during deformation, we update Equation 1 by adding an additional term:

$$E_{fit}(T_j, \mathbf{d}_j^i, s_i) = c(\mathbf{d}_j^i) + c^W(\mathbf{d}_j^i).$$
(2)

 $c^{W}(\mathbf{d}_{j}^{i})$ is used to preserve the equal length, planarity, orthogonality, and symmetry relations between the wires of T_{j} and is defined as in [GSMCO09].

Once the parameters of the wires of T_j are computed, we adopt a 3D volumetric deformation approach to obtain the final geometry of the template. Specifically, we construct a regular 3D deformation grid, V_g , for the template model and sample corresponding points on the original and the updated wires of the model. Using these correspondences as handle constraints, we solve for the new width, height, and depth of each cell in V_g with respect to smoothness constraints on the size of the neighboring grid cells (see [KSSCO08] and [PWS12]). Once the updated grid is computed, the geometry of the deformed template is constructed by preserving the local coordinates of each vertex of the model with respect to its enclosing grid cell.

Parametric deformation model. The parametric model we use to generate curved columns acts an abstract template where each parametric column is an instance of this template. We begin our analysis by first generating a set of concrete column instances given the input elements. We cut each input element along its up direction and fit circles to the resulting outline curves. We generate helical structures in a RANSAC-like fashion by finding groups of circles that are related by the same pitch and thus are possibly swept along the same helix. We then evaluate different combinations of such helicals, i.e. different number and different CSG operations, to generate candidate columns fitting the element. Starting from this initial set of parametric columns, our coupled analysis identifies the best fitting column instance for each element. Column parameters are further optimized by coupling the parameters of the individual helical structures of the column instances detected as similar.

2 Subspace Analysis

After performing template deformations, we construct a multi-layer graph M where each individual graph layer $G_j = (S, E_j, W_j)$ encodes the deformation parameters of the instances of template T_j fitting each element. We adopt the subspace analysis approach of Dong et al. [DFVN14] to extract a set of consistent relations among the elements. This approach first computes a subspace representation U_j for each graph layer as the k- dimensional embedding of the graph, i.e. the smallest k eigenvectors of the corresponding normalized graph Laplacian L_j . In our experiments, we select k such that the gap between the consecutive eigenvalues $(e_{k+1} - e_k)$ is maximized. The individual subspaces U_j are then combined into a common representation U. This is achieved by solving a minimization problem where intuitively U is desired to (i) be as close as possible to each U_j and (ii) preserve the vertex connectivity at each graph layer. The solution U to this minimization problem is obtained as follows. First, a

common graph Laplacian $L = \sum_j L_j - \alpha \sum_j U_j U_j^T$ is computed where α (set to 0.5 in our experiments) balances the tradeoff between the desired properties of U. U is then defined as the smallest k eigenvectors of L.

3 Detailed Results

In the last part of the supplementary material, for each MVS dataset we provide a selection of the input images, 2D edges detected in the images, and views of the constructed 3D line features. For the scan dataset, we provide images of the scene together with the views of the input scan. For each dataset, we provide the element smoothness matrices computed in the first and final iterations of our algorithm together with the detected element similarities. We also include close-up views of the template instances matched to the elements. Finally, we provide the template model set and the grouping used in our evaluations.





































instance selection







input images



MVS reconstruction from the images & 3D scans of the columns overlaid



9 instances selected





We synthetically add noise to the input scan by uniformly disturbing each vertex in the range [-4d, 4d] where d is the average local sample spacing. Below we show the smoothness matrices computed in the first and final iterations of our algorithm for the same set of elements as on the previous page (the elements in the top floor).



7 instances selected



Template Set (Templates used for evaluation in Fig. 6b are shown indicated in orange.)

References

- [BSZF99] BAILLARD C., SCHMID C., ZISSERMAN A., FITZGIBBON A. W.: Automatic line matching and 3D reconstruction of buildings from multiple views. In ISPRS Conf. on Automatic Extraction of GIS Objects from Digital Imagery (1999), pp. 69–80.
- [DFVN14] DONG X., FROSSARD P., VANDERGHEYNST P., NEFEDOV N.: Clustering on multi-layer graphs via subspace analysis on grassmann manifolds. *IEEE Transactions on Signal Processing* (Feb. 2014).
- [GSMCO09] GAL R., SORKINE O., MITRA N. J., COHEN-OR D.: iWIRES: An analyze-and-edit approach to shape manipulation. In ACM TOG (Siggraph) (2009), vol. 28, pp. 33:1–33:10.
- [KSSCO08] KRAEVOY V., SHEFFER A., SHAMIR A., COHEN-OR D.: Nonhomogeneous resizing of complex models. In ACM TOG (Siggraph Asia) (2008), pp. 111:1–111:9.
- [PWS12] PANOZZO D., WEBER O., SORKINE O.: Robust image retargeting via axis-aligned deformation. In Com. Graph. Forum (Eurographics) (2012), vol. 31, pp. 229–236.
- [WB06] WAECHTER A., BIEGLER L. T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming 106*, 1 (2006), 25–57.