Efficient 3D Room Shape Recovery from a Single Panorama: Supplementary Material

Hao Yang Hui Zhang
School of Software, Tsinghua University, Beijing, China
Tsinghua National Laboratory for Information Science and Technology (TNList)
yanghao14@mails.tsinghua.edu.cn, huizhang@tsinghua.edu.cn
http://cgcad.thss.tsinghua.edu.cn/yanghao/3droom

Abstract

*We explain more details of our work entitled **Efficient 3D Room Shape Recovery from a Single Panorama** in this supplementary material for completeness.*

1. Introduction

We provide details of several steps in our algorithm, which are not the core parts and are omitted in the main paper due to page limit. The following sections are organized as follows.

1. In Section 2, details of the over-segmentation algorithm mentioned in Section 2 of the main submission are explained.
2. In Section 3, a more detailed derivation for the Equation 1 of the main submission is presented.
3. In Section 4, the algorithm of finding the largest determinable subgraph, which is mentioned in Section 3.4 of the main submission, is provided.
4. In Section 5, the method of generating the parameter space for selecting the *cuboid of best fit* is given.

2. Panorama Over-segmentation

The over-segmentation algorithm we use for panorama is an extension of the algorithm for normal photographs proposed in [2] with three modifications: 1) the adjacency relationship between image pixels is extended; 2) the size of each image pixel is redefined; 3) a post-process is applied to remove thin superpixels. The algorithm can be described as follows.

First, a graph $G = \{V, E\}$ is constructed. V is the set of the panorama pixels and E is the set of the adjacency relations between the pixels. In addition to the traditional 8-neighborhood adjacency, as indicated by the black lines in Fig. 1, the adjacency relations between the leftmost and the rightmost pixels, between all pairs of the topmost pixels and between all pairs of the bottommost pixels are also considered in E for a 360° full view panorama. These new relations are shown as the red lines in Fig. 1.

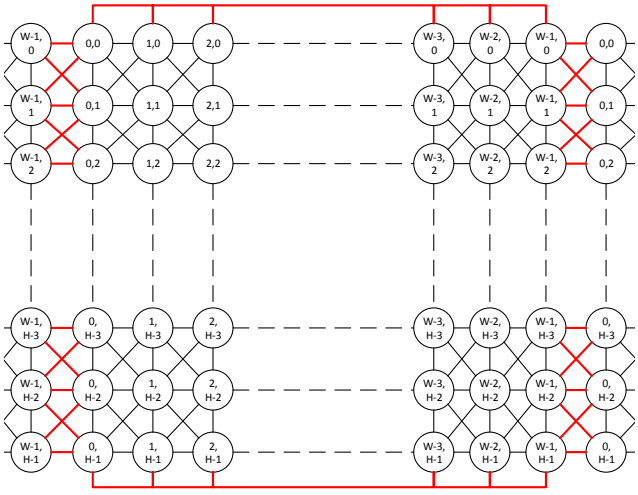

Figure 1: Adjacency relations between panorama pixels. Circles represent panorama pixels, the text in each circle gives its (x, y) position in image. Lines represent the adjacency relations between the pixels. Black lines indicate the traditional 8-neighborhood relations; red lines represent the new relations which are only considered for a panorama.

Different sizes are assigned to vertices (pixels) at different heights. These sizes are formulated as $size(v) = sin(\frac{y}{H}\pi)$, where y is the y-coordinate of the pixel, H is the height of the panorama. The equation is exactly the same with that for pixel weight $w(p)$ which is used by Equation 6 and defined in Section 4 in the main submission. Each edge $e \in E$ is assigned a weight $w(e)$; $w(e)$ follows the same definition proposed in [2]; it is measured by the color difference of two pixels.

A graph-cut is applied on G according to Algorithm 1 to perform the segmentation. The routine is generally the same with the algorithm proposed in [2].

Algorithm 1: Panorama over-segmentation

Input: Graph $G = (V, E)$, with n vertices and m edges; a parameter k.
Output: A segmentation of V into components $S = (C_1, \ldots, C_r)$.

$(o_1, \ldots, o_m) \leftarrow$ sort E by non-decreasing edge weight;
Start with a segmentation S^0, where each vertex v_i is in its own component ;
$\tau \leftarrow$ a map(dictionary) structure used to record the merging threshold of each component;
for $v_i \in V$ **do**
    $C_i^0 \leftarrow$ the component of S^0 containing v_i;
    $\tau[C_i^0] \leftarrow k/size(v_i)$;
for $q \leftarrow 1, \ldots, m$ **do**
    Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, *i.e.*, $o_q = (v_i, v_j)$;
    $C_i^{q-1} \leftarrow$ the component of S^{q-1} containing v_i;
    $C_j^{q-1} \leftarrow$ the component of S^{q-1} containing v_j;
    if $C_i^{q-1} \neq C_j^{q-1} \wedge w(o_q) < \min\{\tau[C_i^{q-1}], \tau[C_j^{q-1}]\}$ **then**
        $S^q \leftarrow S^{q-1}$ by merging C_i^{q-1} and C_j^{q-1};
        τ remains the same for the components in S^q which are not modified ;
        $C_{ij}^q \leftarrow$ the new component merged from C_i^{q-1} and C_j^{q-1};
        $\tau[C_{ij}^q] \leftarrow w(o_q) + k/\sum_{v \in C_{ij}^q} size(v)$;
    else
        $S^q \leftarrow S^{q-1}$;
$S \leftarrow S^m$;

Finally, a post-processing step is appended to remove thin regions. A size 2 dilation is applied on the boundary mask of the current segmentation. Regions that are completely covered by the dilated boundaries are marked as removed. Then, each pixel of the removed region is merged into its nearest region that is not marked as removed.

In Fig. 2 we compare the results of the original algorithm and those of the modified version by applying them to a same indoor panorama. Note that a similiar method is also utilized by [1] to over-segment a panorama, but it remains unknown whether we share the same underlying algorithm due to the lack of details.

(a) Full view (b) Bottom (c) Top

Figure 2: **(a)** The fist row shows the input panorama; the second row shows the segmentation results obtained by directly applying the original algorithm proposed in [2]; the third row shows the results of using the modified version proposed in our work. **(b)** and **(c)** are the bottom and top perspective conversions of (a). The original algorithm cannot reserve the coherence across image boundaries (take note of the boundary lines denoted by arrows shown in the second rows of (b) and (c) as an example). This issue is addressed by our modified algorithm, which can also generate superpixels that are spatially more uniform than those generated by the original algorithm.

3. More About Vertex Parameterization

The detailed derivation for Equation 1 of the main submission is provided here for clearness.

We set the origin of the coordinate system at the viewpoint. Therefore the equation $a_i x + b_i y + c_i z = 1$ can represent all the possible planes that do not pass through the viewpoint, and can thus represent all the visible surface planes of an indoor room. $\boldsymbol{\pi}_i = (a_i, b_i, c_i)^\intercal$ is used to represent the coefficients of the (supporting) plane $a_i x + b_i y + c_i z = 1$ of vertex i, as also denoted in the main submission.

(1) If $df_i = 1$, $\mathbf{n}_i$ is the unit normal vector of its plane $\boldsymbol{\pi}_i$, then a scalar k that satisfies $\boldsymbol{\pi}_i = k\mathbf{n}_i$ should exist. We denote d_i as the distance of the plane $\boldsymbol{\pi}_i$ to the viewpoint (*i.e.*, the origin $(0, 0, 0)$). Let $\mathbf{q} = (q_x, q_y, q_z)^\intercal$ be the point on plane $\boldsymbol{\pi}_i$ nearest to the viewpoint. Then we have $\mathbf{q} = d_i \mathbf{n}_i$, because the direction from $(0, 0, 0)$ to $\mathbf{q}$ must be orthogonal to the plane. We also have $\boldsymbol{\pi}_i^\intercal \mathbf{q} = a_i q_x + b_i q_y + c_i q_z = 1$ because $\mathbf{q}$ lies on the plane $\boldsymbol{\pi}_i$. Therefore, by substituting $\boldsymbol{\pi}_i$ and $\mathbf{q}$ in the equation $\boldsymbol{\pi}_i^\intercal \mathbf{q} = 1$, we obtain

$$(k\mathbf{n}_i)^\intercal (d_i \mathbf{n}_i) = 1 \Rightarrow k d_i (\mathbf{n}_i^\intercal \mathbf{n}_i) = 1 \Rightarrow k d_i = 1.$$

We denote $\mathbf{x}_i = (1/d_i)$ and $\mathbf{P}_i = [\mathbf{n}_i]$ as proposed in Table 1 of the main submission. Equation 1 of the main submission can be derived as:

$$\mathbf{P}_i \mathbf{x}_i = \frac{\mathbf{n}_i}{d_i} = k\mathbf{n}_i = \boldsymbol{\pi}_i.$$

(2) If $df_i = 2$, $\mathbf{u}_i = (u_{ix}, u_{iy}, u_{iz})^\intercal$ is the unit direction vector that the surface plane must be parallel with. Hence, $\mathbf{u}_i$ should satisfy $\boldsymbol{\pi}_i^\intercal \mathbf{u}_i = a_i u_{ix} + b_i u_{iy} + c_i u_{iz} = 0$ because $\mathbf{u}_i$ is orthogonal to the normal of the plane. Then $c_i = -\frac{u_{ix}}{u_{iz}} a_i - \frac{u_{iy}}{u_{iz}} b_i$. Denote $\mathbf{x}_i = (a_i, b_i)^\intercal$ and $\mathbf{P}_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -\frac{u_{ix}}{u_{iz}} & -\frac{u_{iy}}{u_{iz}} \end{bmatrix}$ as proposed in Table 1 of the main submission. Then Equation 1 of the main submission can be derived as:

$$\mathbf{P}_i \mathbf{x}_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -\frac{u_{ix}}{u_{iz}} & -\frac{u_{iy}}{u_{iz}} \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix} = \begin{bmatrix} a_i \\ b_i \\ -\frac{u_{ix}}{u_{iz}} a_i - \frac{u_{iy}}{u_{iz}} b_i \end{bmatrix} = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} = \boldsymbol{\pi}_i.$$

The special case of $u_{iz} = 0$ can be addressed by swapping the components to make sure a nonzero component of $\mathbf{u}_i$ is set as the denominator. In particular, assume $u_{iy} \neq 0$ is the nonzero component, then denote $\mathbf{x}_i = (a_i, c_i)^\intercal$ and let $\mathbf{P}_i = \begin{bmatrix} 1 & 0 \\ -\frac{u_{ix}}{u_{iy}} & -\frac{u_{iz}}{u_{iy}} \\ 0 & 1 \end{bmatrix}$, and Equation 1 of the main submission can be derived in the same manner.

(3) If $df_i = 3$, then $\mathbf{x}_i = (a_i, b_i, c_i)^\intercal = \boldsymbol{\pi}_i$ and $\mathbf{P}_i = \mathbf{I}_{3\times 3}$ according to Table 1 of the main submission. It is straightforward that $\mathbf{P}_i \mathbf{x}_i = \mathbf{I}_{3\times 3} \boldsymbol{\pi}_i = \boldsymbol{\pi}_i$.

4. Subgraph Determination

After constructing the constraint graph $\mathcal{G}$, as described in Section 3.4 of the main submission, expansions(searches) are applied to retrieve a subgraph $\mathcal{G}^*$ from $\mathcal{G}$ to avoid any underdeterminacy within the system to solve.

The expansion starts from a vertex with DOF=1, its algorithm is presented in Algorithm 2. The subroutine *is-determinable*(S, i) used in Algorithm 2 judges whether the spatial configuration of vertex i can be determined given the depths of all the connection points in S. Its formal definition is described as follows.

Let S be the set of the connection points whose depths are all given, and let i be the vertex. If $df_i = 1$, then *is-determinable*(S, i)=$(\|S\| > 0)$. If $df_i = 2$, let $\mathbf{u}_i$ be the direction vector that the (supporting) plane of i must be parallel with. Then, *is-determinable*(S, i) is true iff at least two points P_1, P_2 exist in S and satisfy $(\overrightarrow{OP_1} \times \overrightarrow{OP_2}) \not\perp \mathbf{u}_i$, where O is the viewpoint and $\overrightarrow{OP_{\{1,2\}}}$ are the view directions of the points $P_{\{1,2\}}$, and $\times$ represents the cross product. Finally, if $df_i = 3$, then *is-determinable*(S, i) is true iff at least three points $P_{\{1,2,3\}}$ exist in S and are not visually collinear, which means they should satisfy $(\overrightarrow{OP_1} \times \overrightarrow{OP_2}) \nparallel (\overrightarrow{OP_2} \times \overrightarrow{OP_3})$. In implementation, the conditions are all checked according to a threshold.

Algorithm 2: Subgraph expansion from a root vertex

Input: Constraint graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
A root vertex $r \in \mathcal{V}$, $df_r = 1$.
Output: Subgraph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ without underdeterminacy.

$\mathcal{V}^* \leftarrow \varnothing$; $\mathcal{U} \leftarrow \{r\}$;
$S \leftarrow$ an array of sets of connection points;
for $i \in \mathcal{V}$ **do** $S[i] \leftarrow \varnothing$;
repeat
  $\mathcal{T} \leftarrow \varnothing$;
  for $i \in \mathcal{U}$ **do**
    for $c \in$ *related-constraints*(i) **do**
      $j \leftarrow$ the vertex of c other than i;
      if $j \in \mathcal{V}^* \vee j \in \mathcal{U}$ **then** continue;
      else if $c \in \mathcal{E}_{cop}$ **then** $\mathcal{T} \leftarrow \mathcal{T} \cup \{j\}$;
      else if $c \in \mathcal{E}_{con}$ **then**
        $S[j] \leftarrow S[j] \cup$ connection points of c;
        if *is-deteminable*$(S[j], j)$ **then**
          $\mathcal{T} \leftarrow \mathcal{T} \cup \{j\}$
  $\mathcal{V}^* \leftarrow \mathcal{V}^* \cup \mathcal{U}$;
  $\mathcal{U} \leftarrow \mathcal{T}$;
until $\mathcal{U} = \varnothing$;
$\mathcal{E}^* \leftarrow$ all the edges connecting vertices $\in \mathcal{V}^*$;
$\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$;

The expansion is performed once for each vertex with DOF=1 to retrieve multiple subgraphs. The subgraph with the most vertices is selected as the final result. The size of the resulted subgraph determines the coverage of reconstruction, which is reported in Section 4 of the main submission.

5. Parameter Space for COBF Selection

The *cuboid of best fit* (COBF) is selected by enumerating cuboid parameters in a discretized parameter space of vanishing direction aligned cuboids. The 3D position of an axis-aligned cuboid can be parametrized by 6 parameters: $(x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max})$, which are the minimal and maximal coordinates of all 8 cuboid vertices on the vanishing directions as shown in Fig. 3.

We build the discretized parameter space in two steps. First, the corners of the cuboids are restricted to be on integral grid points satisfying $x_{min}, y_{min}, z_{min} \in \{-1, -2, \ldots, -N\}$ and $x_{max}, y_{max}, z_{max} \in \{1, 2, \ldots, N\}$ with N being a positive integer. Enumerating all combinations of these six parameters generates N^6 cuboids, some of them are same up to scales. Therefore, each cuboid is rescaled by normalizing the length of its diagonal; then any

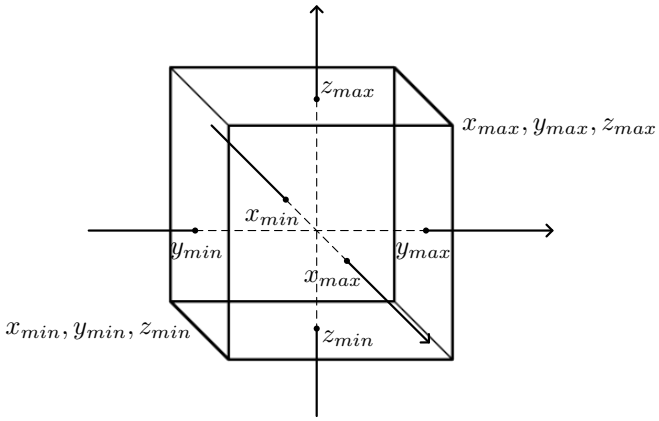

Figure 3: Parameters of an aligned cuboid.

two cuboids whose parameters satisfy $\|P^1_{min} - P^2_{min}\| < t$ and $\|P^1_{max} - P^2_{max}\| < t$ are merged, where $P^{\{1,2\}}_{min} = (x^{\{1,2\}}_{min}, y^{\{1,2\}}_{min}, z^{\{1,2\}}_{min})$ and $P^{\{1,2\}}_{max} = (x^{\{1,2\}}_{max}, y^{\{1,2\}}_{max}, z^{\{1,2\}}_{max})$ are the two corners of the $\{$first, second$\}$ cuboid. $15,559$ cuboids are generated by setting $N = 5$ and $t = 0.01$. Finally, all these cuboids are rotated to world coordinates according to the vanishing directions of each scene.

References

[1] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014. 2

[2] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 1, 2