ELSR: Efficient Line Segment Reconstruction with Planes and Points Guidance

Dong Wei, Yi Wan†, Yongjun Zhang†, Xinyi Liu, Bin Zhang, Xiqi Wang
School of Remote Sensing and Information Engineering, Wuhan University
{weidong,yi.wan,zhangyj,liuxy,bin.zhang,wangxiqi}@whu.edu.cn

Abstract

Three-dimensional (3D) line segments are helpful for scene reconstruction. Most of the existing 3D-line-segment-reconstruction algorithms deal with two views or dozens of small-size images; while in practice there are usually hundreds or thousands of large-size images. In this paper, we propose an efficient line segment reconstruction method called ELSR [1]. ELSR exploits scene planes that are commonly seen in city scenes and sparse 3D points that can be acquired easily from the structure-from-motion (SfM) approach. For two views, ELSR efficiently finds the local scene plane to guide the line matching and exploits sparse 3D points to accelerate and constrain the matching. To reconstruct a 3D line segment with multiple views, ELSR utilizes an efficient abstraction approach that selects representative 3D lines based on their spatial consistence. Our experiments demonstrated that ELSR had a higher accuracy and efficiency than the existing methods. Moreover, our results showed that ELSR could reconstruct 3D lines efficiently for large and complex scenes that contain thousands of large-size images.

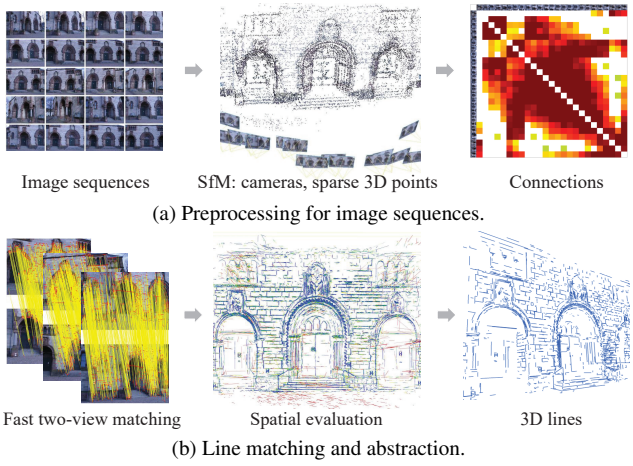
1. Introduction

1.1. Motivation and Objective

3D line segments are dominant in most urban scenes. It provides structure information that is very useful for 3D scene abstraction [12], dense matching [26], plane detection [15], and surface reconstruction [27]. But 3D line segment reconstruction is still a challenging task in computer vision.

The main drawback of the existing line reconstruction methods is that they lack an elegant solution for fast and robust line matching in two views. Line matching is more complex than point matching because line segments are indistinct in both texture and geometry in two views, and there are few deep-learning-based methods for line matching. Generally, the line needs to be grouped [7, 21] and the texture needs to be rectified [17, 23] for robust matching. However, these operations are computationally expensive for they lack geometry guidance.

In this paper, we focus on improving the 3D line segment reconstruction by geometry-guided matching for the



(a) Preprocessing for image sequences.

(b) Line matching and abstraction.

Figure 1. The pipeline of ELSR. ELSR acquires the camera matrix, the sparse 3D point, and the connection between the images from the SfM pipeline. ELSR first matches line segments for image pairs and finally abstracts the representative 3D lines by evaluating the matches in the image sequence.

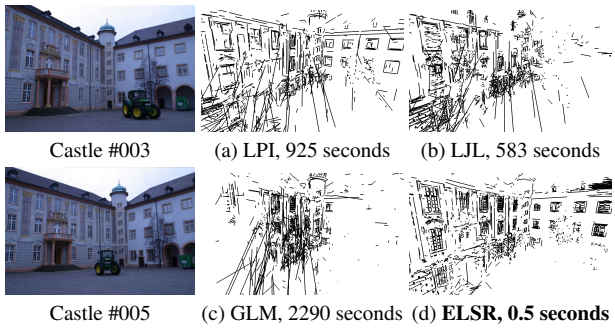


Figure 2. The 3D lines are reconstructed from two-view matching. ELSR is compared with LPI [7], LJL [17], and GLM [30].

†Corresponding author
[1]Available at https://skyearth.org/publication/project/ELSR

line segment. Our method is based on two observations: 1) there are many planes in the city scene, and the line segments tend to lie in the scene planes; and 2) it is a well-known fact in computer vision that exploiting the rough 3D model can greatly improve the efficiency and accuracy of 3D reconstruction. Thus, if we have found the scene planes, we can use the local homography to guide line matching and constrain the line candidate to a small region with sparse 3D points, which can be obtained with the efficient structure-from-motion (SfM) methods, such as VisualSfM [32], Bundler [24], and OpenMVG [20].

However, diving into the above two observations also brings about two challenges: 1) limited by the complexity of the city scene, finding all of the scene planes accurately and efficiently is challenging; and 2) the depth constraint is quite rough because the 3D points obtained with SfM are sparse, and the line segment is indistinct in geometry. Thus, we will propose an efficient method to determine the scene plane and calculate the homography; we will address the rough depth-constraint by combining sparse 3D points with lines and planes as a guidance but not a determination.

1.2. Method Overview

Fig. 1 shows the pipeline of ELSR. Given the image sequence, we first acquire the camera poses and the sparse 3D points with the SfM pipeline; then, we match lines for two views; finally, we abstract representative 3D lines from all of the matches in the image sequence. In detail, ELSR contains three components:

- **Homography estimation**. We use the scene plane geometry (Sec. 3.1) with two neighborhood lines to efficiently verify the valid homography, during which the rough point-depth is used for acceleration (Sec. 3.2).
- **Guided matching** (Sec. 3.3). We match the single line with the potential homographies, and the rough point-depth is used to constrain the matching.
- **Line abstraction** (Sec. 4). For multiple views, we first find the connections of the line matches among image pairs and score the spatial consistence; then, we select the representative matches as the final 3D lines.

As shown in Fig. 2, ELSR is more efficient and accurate than the existing algorithms in two-view matching because **the candidate in the second view is constrained to a very small region with planes and points guidance**.

2. Related work and our contributions

2.1. Two-view matching

As the vital step for line segment reconstruction, line segment matching in two views has been studied for decades. While a few past methods [29] relied only on textures for line matching, many other studies introduced the camera pose for robust matching. Schmid *et al*. [23] exploited the projective transformation induced by epipolar geometry to improve the gray correlation. OK *et al*. [21] improved the texture correlation by constructing the Daisy descriptor for image lines with epipolar constraint. Mohammed *et al*. [1] established the candidate based on the intersection of line-pairs and constructed an association graph for robust matching. Sun *et al*. [28] used the potential planes with sparse 3D points and the camera matrix to guide the line matching. Wei *et al*. [30] matched lines via the potential homographies with pairwise line segments under scene plane assumption.

Some methods attempt to be independent of the camera pose, which requires a well-designed descriptor or geometry constraint. Zhang *et al*. [33] created a scale invariant texture descriptor for an individual line; and an association graph was constructed for all matches to enhance the matching. Li *et al*. [17] grouped pairwise line segments and constructed an affine-invariant region descriptor. Because point matching is more mature and robust than line matching; some algorithms exploit the line-point geometry to confirm line matches. Fan *et al*. [7,8] used two points and one line to construct the affine invariant structure. Manolis *et al*. [18] built the projective invariant structure with two points and two lines. Ramalingam *et al*. [22] used projective geometry like [18] to match both points and lines. Wei *et al*. [31] used the point match to construct the association graph. However, most algorithms are computationally expensive.

2.2. Multiple-view reconstruction

After line segment matching, a clustering method merges the 3D lines related to the same line. Schmid *et al*. [23] and Wei *et al*. [31] merged the 3D line based on the projection in the images. Merging in the image space is intuitive, but it may fail when the line is near the epipolar plane. This failure will result in many incorrect 3D lines. Thus, Wei *et al*. [31] discarded the image line within a certain distance to the epipole. Li *et al*. [17] used the scene plane constraint to fix the degraded 3D line segment, but the time-consuming process in confirming scene planes may fail in complex scenes. Jain *et al*. [13] merged the 3D line located within an encircling cylinder; however, it is challenging to determine the radius when there is no scale information. Hofer *et al*. [12] calculated the clustering distance in object space, which was adaptive to the depth of the 3D lines. However, the false clustering could not be avoided and the authors thus used the graph optimization to acquire the result.

2.3. Our contributions

In summary, our contributions are:

- We propose an efficient method to match lines and reconstruct 3D lines from multiple images, which is easy to use and just requires the result of the SfM pipeline

as prior knowledge.

- We exploit the geometric relation between 2D lines and sparse 3D points to find the local homography efficiently. To our best knowledge, this is the first work that utilizes this simple yet efficient geometry for line segment matching.
- When evaluated on large image datasets, ELSR was over 1000 times faster than the existing algorithms in two-view matching; in multiple-view reconstruction, ELSR was 4 times faster than the state-of-the-art method and the quantity of 3D lines was raised by 360%.

3. Matching with two views

This section describes how to efficiently match line segments in two views with the guidance of scene planes and points. We confirm the neighbors of the image based on the number of the common world-points and connect one image to its t_{img} neighbors as the image pair for matching.

3.1. The homography with pairwise lines

Now we introduce the geometry constraint to obtain the homography for guiding the line matching. Denote the scene plane as $\boldsymbol{\pi} = (\mathbf{v}^\top, 1)^\top$, where $\mathbf{v} = (v_1, v_2, v_3)^\top$. With the fundamental matrix F, the scene plane theory [23] shows that the homography of the image plane related to $\boldsymbol{\pi}$ is calculated by

$$\mathrm{H} = \mathrm{A} - \mathbf{e}'\mathbf{v}^\top, \tag{1}$$

where $\mathbf{e}'$ is a homogeneous 3D vector that represents the epipole of the second view. A is a 3×3 matrix,

$$\mathrm{A} = [\mathbf{e}']\,\mathrm{F}. \tag{2}$$

where $[\mathbf{e}']$ is the skew 3×3 matrix.

Given two line matches denoted as $\mathbf{l}_1 \leftrightarrow \mathbf{l}'_1$ and $\mathbf{l}_2 \leftrightarrow \mathbf{l}'_2$, Wei *et al*. [30] solved for $\mathbf{v}$ by listing four equations with

$$\mathbf{l}'^\top \left(\mathrm{A} - \mathbf{e}'\mathbf{v}^\top\right)\mathbf{x} = 0, \tag{3}$$

to verify line matches, where the endpoint $\mathbf{x} = (x, y, 1)^\top$ is on the line $\mathbf{l} = (a, b, c)^\top$. Then, the two lines in the first image are mapped with H, and they are verified by the overlap and distance. In the rest of the paper, all lines and points are represented as the homogeneous form, and $'$ marks the second image.

In fact, because $\mathbf{v}$ is a 3D vector, it can be determined with just three equations:

$$\left[\mathbf{l}'_1\ \mathbf{l}'_1\ \mathbf{l}'_2\right]^\top \left[\mathrm{A} - \mathbf{e}'\mathbf{v}^\top\right] \left[\mathbf{x}_{1,1}\ \mathbf{x}_{1,2}\ \mathbf{x}_{2,1}\right] = 0, \tag{4}$$

where the subscript of $\mathbf{x}$ is in the form of "*line index, endpoint index*". Thus, as Fig. 3 shows, we have a unique endpoint $\mathbf{x}_{2,2}$ to verify H by the angular similarity:

$$S_{\text{ang}}(\mathrm{H}) = \angle\left(\mathbf{x}'_{2,2} - \mathbf{x}'_{2,1}, \mathrm{H}(\mathbf{x}_{2,2}) - \mathbf{x}'_{2,1}\right), \tag{5}$$

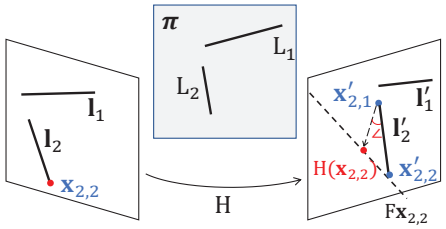


Figure 3. The geometry for pairwise lines. $\mathbf{x}_{2,2} \leftrightarrow \mathbf{l}'_2$ is not used in Eq. (4) to determine H. Thus, it just satisfies the epipolar constraint in the mapping with H.

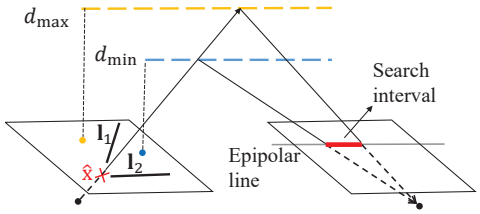


Figure 4. Point-guided searching. The depth range of the neighbor points constrains the intersection search to a small interval.

where $\angle$ is the clockwise angle of two vectors, and $\mathrm{H}(*)$ maps the feature with the homography H.

When $S_{\text{ang}}(\mathrm{H})$ is smaller than a given threshold t_{ang}, we say the scene plane $\boldsymbol{\pi}$ is true, and the induced H is valid. It is more accurate and efficient than [30]. In [30], the $\mathbf{v}$ is overfitted with four equations by the least square method, and the computation with distance and overlap is less efficient than the angular similarity of our method.

Note, the pairwise lines will be used only if their intersection is within t_{int} pixels to each line, because the neighborhood lines are more likely to be coplanar and the computation can be reduced. In addition, the line segment in ELSR is with a certain direction based on the gradient direction as in [9].

3.2. Point-guided searching

Algorithm 1 shows our method to find the homography. Searching the homography is conducted to find the pairwise line matches satisfying $S_{\text{ang}}(\mathrm{H}) < t_{\text{ang}}$. As illustrated in Fig. 4, the coplanar pairwise lines intersect at a definite point $\hat{\mathbf{x}}$. Thus, the intersection $\hat{\mathbf{x}}'$ in the second view is searched along the epipolar line. We will show the sparse 3D points obtained by SfM can reduce the search to a small interval.

Denote the first camera matrix as P, which is a 3×4 matrix. Let M and $\mathbf{c}^4$ be the left hand 3×3 submatrix and the last column of P, respectively. If the depth of $\hat{\mathbf{x}}$ ($\mathrm{dep}(\hat{\mathbf{x}})$) is available, calculating the position of $\hat{\mathbf{x}}$ in the object space is straight forward:

$$\mathrm{X} = \left[\left(-\mathrm{M}^{-1}\mathbf{c}^4 + (\mathrm{dep}(\hat{\mathbf{x}})/\cos(\mathbf{r}, \mathbf{p}))\,\mathbf{r}\right)^\top 1\right]^\top. \tag{6}$$

Algorithm 1 Point-guided searching

Input: ls, a pair of lines;
Ps, the neighborhood 3D points of ls's intersection;
M, the intersections'map for the second image;
t_{ang}, the angular threshold;
Output: H^*, the valid homography;

$\mathrm{H}^* \leftarrow null$
$[\hat{\mathbf{x}}'_{\min}, \hat{\mathbf{x}}'_{\max}] \leftarrow$ use Eq. (6) with Ps and ls
for $(x, y) \in [\hat{\mathbf{x}}'_{\min}, \hat{\mathbf{x}}'_{\max}]$ **do**
  if $\mathrm{M}(x, y) \neq null$ **then**
    $ls' \leftarrow \mathrm{M}(x, y)$
    $\mathrm{H} \leftarrow$ use Eq. (4) with ls and ls'
    if $S_{\text{ang}}(\mathrm{H}) < t_{\text{ang}}$ **then**
      $t_{\text{ang}} \leftarrow S_{\text{ang}}(\mathrm{H})$
      $\mathrm{H}^* \leftarrow \mathrm{H}$

Algorithm 2 Homography-guided matching

Input: $\mathbf{l}_i$, a single line, and its depth range is known;
Hs, the t_{hom} neighborhood homograpies of $\mathbf{l}_i$;
M, the line map for the second image;
Output: j^*, the index of the best line for $\mathbf{l}_i$;

$S_{\text{pos}}(i, :) \leftarrow 0$
$\text{Status}(:) \leftarrow 0$
for $Hs_k \in Hs$ **do**
  $\bar{\mathbf{l}}_i^k \leftarrow$ use Eq. (7) with $\mathbf{l}_i$ and Hs_k
  for $(x, y) \in \bar{\mathbf{l}}_i^k$ **do**
    $\mathbf{l}'_j \leftarrow M(x, y)$
    if $\mathbf{l}'_j \neq null$ and $\text{Status}(j) \neq 1$ **then**
      $\text{Status}(j) \leftarrow 1$
      if $\mathbf{l}'_j$ and $\bar{\mathbf{l}}_i^k$ satisfy Eqs. (8) and (9) **then**
        $S_{\text{pos}}(i, j) \leftarrow$ update with Eq. (11)
$j^* = \text{argmax}(S_{\text{pos}}(i, j))$

where $\mathbf{r}$ is the unit 3D-vector passing through the camera center ($-\mathrm{M}^{-1}\mathbf{c}^4$) and $\hat{\mathbf{x}}$, and $\mathbf{p}$ is the unit 3D-vector of the principal ray. Then, $\hat{\mathbf{x}}'$ in the second image can be obtained by projection with X.

The depth of $\hat{\mathbf{x}}$ is unfortunately unknown. However, the t_{pts} neighborhood points of $\hat{\mathbf{x}}$ can give a depth range, with which Eq. (6) can calculate the interval of $\hat{\mathbf{x}}$ as $\hat{\mathbf{x}}'_{\min}$ and $\hat{\mathbf{x}}'_{\max}$. $\hat{\mathbf{x}}$ may be beyond $\hat{\mathbf{x}}'_{\min}$ and $\hat{\mathbf{x}}'_{\max}$ when the depth is discontinuous. Thus, the search is extended along $\hat{\mathbf{x}}'_{\min}$ and $\hat{\mathbf{x}}'_{\max}$ for several pixels, *e.g.*10 pixels are sufficient.

3.3. Homography-guided matching

Algorithm 2 shows the matching process. Let the k-th neighbourhood homography of line $\mathbf{l}_i$ be H_k, which can be queried efficiently by building a KD-tree for the intersections. Denote the mapping of $\mathbf{l}_i$ with H_k as

$$\bar{\mathbf{l}}_i^k = \mathrm{H}_k(\mathbf{l}_i). \tag{7}$$

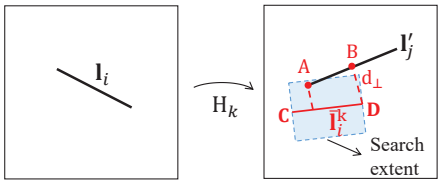


Figure 5. The homography-guided matching. $\mathbf{l}'_j$ is searched within the buffer of $\bar{\mathbf{l}}_i^k$. The maximum distance $|\mathrm{BD}|$ and the overlap rate $|\mathrm{AB}|/|\mathrm{CD}|$, are calculated for validation. The overlap rate is the length ratio between the overlap and the shorter of $\bar{\mathbf{l}}_i^k$ and $\mathbf{l}'_j$.

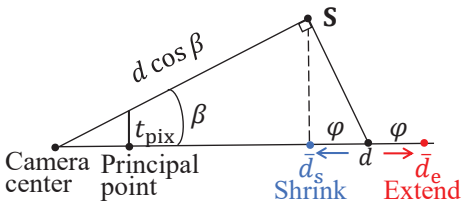


Figure 6. With φ obtained from pixel shift, $d_{\min}$ is shrank and $d_{\max}$ is extended. $\varphi = d - d\cos^2\beta$, $\bar{d}_{\min} = d_{\min} - \varphi$, and $\bar{d}_{\max} = d_{\max} + \varphi$.

As Fig. 5 shows, $\mathbf{l}_i \leftrightarrow \mathbf{l}'_j$ is considered as a potential match when

$$\mathrm{d}_\perp(\bar{\mathbf{l}}_i^k, \mathbf{l}'_j) < t_{\text{pos}} \text{ and } \mathrm{o}_\perp(\bar{\mathbf{l}}_i^k, \mathbf{l}'_j) > t_{\text{ove}}, \tag{8}$$

where $\mathrm{d}_\perp$ calculates the maximum distance between line segments, $\mathrm{o}_\perp$ gives the overlap rate, and t_{pos} and t_{ove} are the thresholds of the position and overlap rate, respectively.

Now we control false matches with the depth constraint. Let the endpoint of $\mathbf{l}$ be $\mathbf{x}$. Let the depth range of its t_{pts} neighbourhood 3D points be $d_{\min}$ and $d_{\max}$. The depth range can be exploited to constrain $\mathbf{x}$. But it need to be extended because $\mathbf{x}$ may be beyond the depth range, especially when the local scene is discontinuous. However, we may not know the unit of the object space. Thus, we exploit the connection between the pixel unit and the object distance to determine the extension. As in Fig. 6, we shift the principal point horizontally by t_{pix} pixels to obtain β. Then, we could calculate the depth shift φ corresponding to the pixel shift. Finally, $d_{\min}$ and $d_{\max}$ are shrank and extended with φ, respectively, to obtain the depth range.

$$\text{dep}(\mathbf{x}) \in \left[d_{\min}\cos^2\beta, d_{\max}\left(2 - \cos^2\beta\right)\right]. \tag{9}$$

The line's endpoint-depth is

$$\text{dep}(\mathbf{x}) = \mathbf{m}^3\tau(\mathbf{x}, \mathbf{F}\mathbf{x} \times \mathbf{l}', \mathrm{P}, \mathrm{P}'), \tag{10}$$

where $\mathbf{m}^3$ is the last row of P, τ is the triangulation function to obtain the 3D point, and $\mathbf{F}\mathbf{x} \times \mathbf{l}'$ calculates the point related to $\mathbf{x}$ on $\mathbf{l}'$.

There are generally several homographies to the same line, and some of them may be incorrect. Thus, we use the

t_{hom} neighbourhood homographies to guide $\mathbf{l}_i$ and score the match by the summary of the positional similarity:

$$S_{\text{pos}}(i,j) = \sum_{k=1}^{t_{\text{hom}}} \exp\left(-\mathrm{d}_{\perp}(\bar{\mathbf{l}}_i^k, \mathbf{l}'_j)/2t_{\text{pos}}\right). \quad (11)$$

$\mathbf{l}_i \leftrightarrow \mathbf{l}'_j$ that has the maximum $S_{\text{pos}}(i,j)$ is selected as the correct match.

4. Abstraction from multiple views

Each match in two views will reconstruct a 3D line segment; therefore, the 3D line segments related to the same line need to be merged as a cluster [12, 13, 31]. However, it is easy to fail for three reasons: 1) the fixed threshold can easily produce incorrect clusters; 2) the wrong match will result in bad reconstruction; and 3) there is currently no robust RANSAC method to confirm the inliers of the line cluster. Thus, ELSR abstracts the representative lines from the cluster instead of merging them, which could be more robust and efficient. The abstraction contains two steps: 1) calculate the spatial similarity between 3D lines and 2) abstract the representative 3D line across all views based on its spatial similarity with others.

4.1. The spatial similarity of 3D lines

Denote the 3D line triangulated from $\mathbf{l}_{a1}$ and $\mathbf{l}_{a2}$ as L_a. The similarity of L_a and L_b is scored by the angular similarity in object space and the positional similarity in image space.

$$S_{\text{ang}}(a,b) = \angle\left(\vec{\mathrm{L}}_a, \vec{\mathrm{L}}_b\right) \quad (12)$$

$$S_{\text{pos}}(a,b) = \max\left\{d_{\perp}(\bar{\mathbf{l}}_{a1}, \mathbf{l}_{b1}), d_{\perp}(\bar{\mathbf{l}}_{a2}, \mathbf{l}_{b2})\right\}, \quad (13)$$

where $\bar{\mathbf{l}}_{a1}$ and $\bar{\mathbf{l}}_{a2}$ are the projections of L_a to the two views of L_b. Note, $\mathrm{o}_{\perp}(\bar{\mathbf{l}}_{a1}, \mathbf{l}_{b1})$ and $\mathrm{o}_{\perp}(\bar{\mathbf{l}}_{a2}, \mathbf{l}_{b2})$ should be bigger than t_{ove}. The 3D line has a direction because the endpoint of the image line has been reordered based on the gradient orientation along the line sehment.

We use a regularization function to unify Eqs. (12) and (13):

$$S(a,b) = e^{-\max\{S_{\text{ang}}(a,b)/t_{\text{ang}}, S_{\text{pos}}(a,b)/t_{\text{pos}}\}/2}. \quad (14)$$

With Eq. (14), the spatial score of L_a is a summary of the spatial similarity with other 3D lines:

$$S^a = \ln(\theta^a/2t_{\text{epi}}) \sum_{b=1}^{n} S(a,b) \;\; s.t.\; S(a,b) > 0.5. \quad (15)$$

where θ^a is the minimum angle of the line segment and the epipolar line, i.e.

$$\theta^a = \min\left\{\angle\left(\mathrm{F}\mathbf{x}_{a1}^1, \mathbf{l}_{a2}\right), \angle\left(\mathrm{F}\mathbf{x}_{a1}^2, \mathbf{l}_{a2}\right)\right\}, \quad (16)$$

in which $\mathbf{x}_{a1}^1$ and $\mathbf{x}_{a1}^2$ are the two endpoints on $\mathbf{l}_{a1}$. We introduce θ^a for the accuracy of the 3D line is closely related to their distance to the epipolar plane [10]. t_{epi} is a regularization threshold for the epipolar angle.

4.2. Greedy assignment

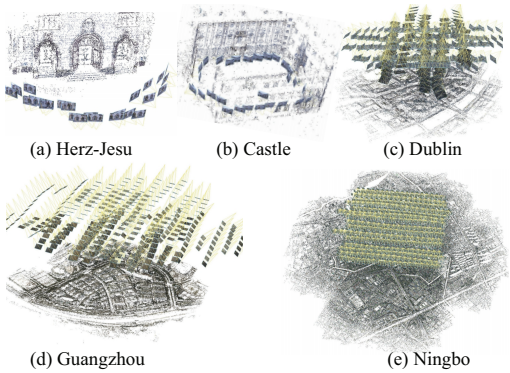
We obtain the vector $\mathbf{s} = [S^1, S^2, ..., S^n]$ after calculating the spatial score for each 3D line. Then, the representative 3D line is confirmed with a greedy strategy:

1) Find $a^* = \mathrm{argmax}_{a\in[1,n]}(\mathbf{s}(a))$. Stop the selection if $\mathbf{s}(a^*) \leqslant 0$. Otherwise, select L^{a^*} as a representative 3D line and set $\mathbf{s}(a^*) = 0$.
2) Set all 3D lines that are consistent with L^{a^*}, i.e., $\mathbf{s}(b^*) = 0$ when $S^{a^*,b^*} > 0.5$.
3) Go back to step 1).

For robustness, the representative 3D line will be used only if its spatial score is bigger than 1, which means it has a spatial similarity with other lines.

5. Experiments

We evaluated ELSR on five datasets and compared it with four existing algorithms. Tab. 1 describes the datasets, and Fig. 7 visualizes their cameras and sparse points acquired with VisualSfM [32]. The first three datasets are open-source; we captured the other two with unmanned aerial vehicles. We detected line segments with LSD [9] without setting a threshold. Sec. 5.3 discusses the hyperparameters in ELSR. Our test system was a desktop machine with an Intel(R) Core(TM) i7-10700KF CPU that supports 16 threads, 32 GB of main memory, and a Nvidia RTX 3070 GPU.



(a) Herz-Jesu (b) Castle (c) Dublin

(d) Guangzhou (e) Ningbo

Figure 7. The cameras and the sparse 3D points.

5.1. Matching with two views

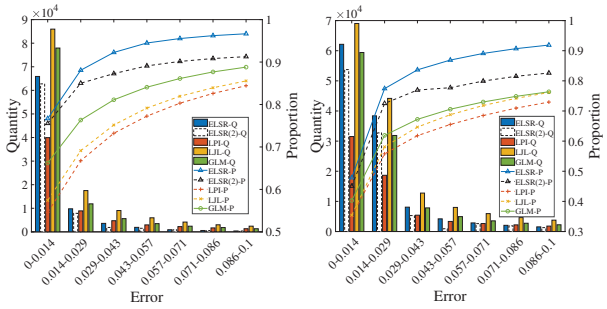
To evaluate the two-view matching, we made 165 image pairs from Herz-Jesu and Castle datasets by setting (t_{img}) as 3. We compared ELSR with LPI [6, 7], LJL [14, 17], and GLM(Graph-based Line Matching) [5, 30]. We did not

Dataset	Imgs	Size (px)	ANTP
Herz-Jesu [25]	25	3072×2048	3895
Castle [25]	30	3072×2048	3892
Dublin [16]	369	9000×6732	3478
Guangzhou	500	10336×7788	3139
Ningbo	1454	11608×8708	3197

Table 1. The image dataset for evaluation. "ANTP" is the average number of the tie-points obtained by VisualSfM.

Algorithm	Herz-Jesu		Castle	
	Total	Average	Total	Average
LPI	14.78 h	11.98 m	15.73 h	10.49 m
LJL	10.37 h	8.41 m	32.65 h	21.76 m
GLM	39.83 h	32.29 m	56.39 h	37.59 m
ELSR	**29.12** s	**0.39** s	**40.82** s	**0.45** s

Table 2. The run time of matching with the 165 pairs. "h,""m," and "s" denote "hour,""minute," and "second." The time does not include line detection while contains the run time of Visual SfM.



(a) Herz-Jesu (b) Castle

Figure 8. The evaluation of two-view matching with the 165 image pairs. "Q" and "P" denote "Quantity" and "Proportion". "ELSR(2)" runs without N-view SfM. Note, the proportion contains the summary before the error interval.

evaluate Line3D++ because it cannot match lines with two views. As in [12, 13], we evaluated the match in object space. We reconstructed the 3D line segments with the matching results and acquired the dense points with ContextCapture [4]. Then, we calculated the maximum distance from the 3D line segment to the mesh as the error. We present the error distribution in different intervals because it may be inaccurate to determine the correct/incorrect matches with a fixed threshold. Note, LPI, GLM, and ELSR required points or cameras as prior knowledge, which were acquired with VisualSfM in our experiment.

Fig. 8 depicts the evaluation result. ELSR produced the best accuracy. As shown in Fig. 2, ELSR produced less disordered 3D lines. However, ELSR was not the best in line

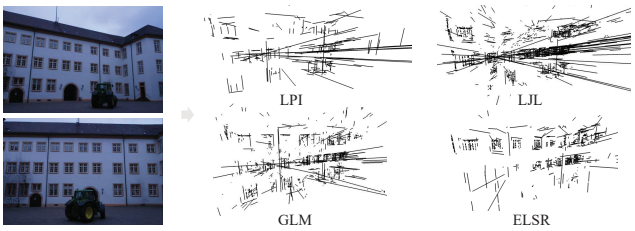


Figure 9. The failure case of ELSR: it may fail when the scene has only one plane and the lines in the plane have repeated patterns.

quantity because the depth constraint of the sparse tie-point may bring about the false negative. As for computation time (Tab. 2), ELSR was at least 1000 times faster than the other three algorithms for the two datasets. Fig. 9 shows one failure case of ELSR.

Running with the F matrix and point from N-view SfM is unfair for LJL, since it requires no prior cues. Thus, we also ran ELSR with SIFT [19]+GCRANSAC [2]+MAGSAC [3] to obtain accurate F matrix and points; the match amount was roughly the same, but the proportions in different error intervals decreased by 0.1%-3%.

5.2. Reconstruction from multiple views

We evaluated ELSR for 3D line segment reconstruction with all the datasets in Tab. 1. We also compared its results with that of Line3D++ [12] for which the code is available and is currently the most efficient algorithm for line reconstruction. Both ELSR and Line3D++ used the result (cameras and sparse 3D points) of VisualSfM as input. In Line3D++, we set the two parameters, the maximum number of line segments and the maximum image-width, to a significantly large number to avoid its controlling for image size and line-segment numbers. Line3D++ ran with both CPU and GPU, while ELSR only ran with a CPU.

As shown in Fig. 10, ELSR always obtained more 3D lines than Line3D++, especially for the aerial image dataset. See Fig. 11 for some of the details in the local areas. ELSR obtained more 3D lines about the details of the scenes, such as windows, roads, and playfields. Fig. 12 plots the profiles of the endpoints on the 3D lines, which demonstrates that the depth constraint of ELSR is effective in controlling the false positives.

For quantitative evaluation, we reconstructed the 3D mesh of the first three datasets with ContexCapture and calculated the maximum distance from the 3D line to the mesh. The results are shown in Fig. 13. ELSR reconstructed 3D lines with more errors; however, we thought this sacrifice was acceptable because ELSR produced much more correct lines than Line3D++.

For the run time, Line3D++ was about three times slower than ELSR in the first two datasets. In the aerial images that were large in size and quantity, Line3D++ was several hours

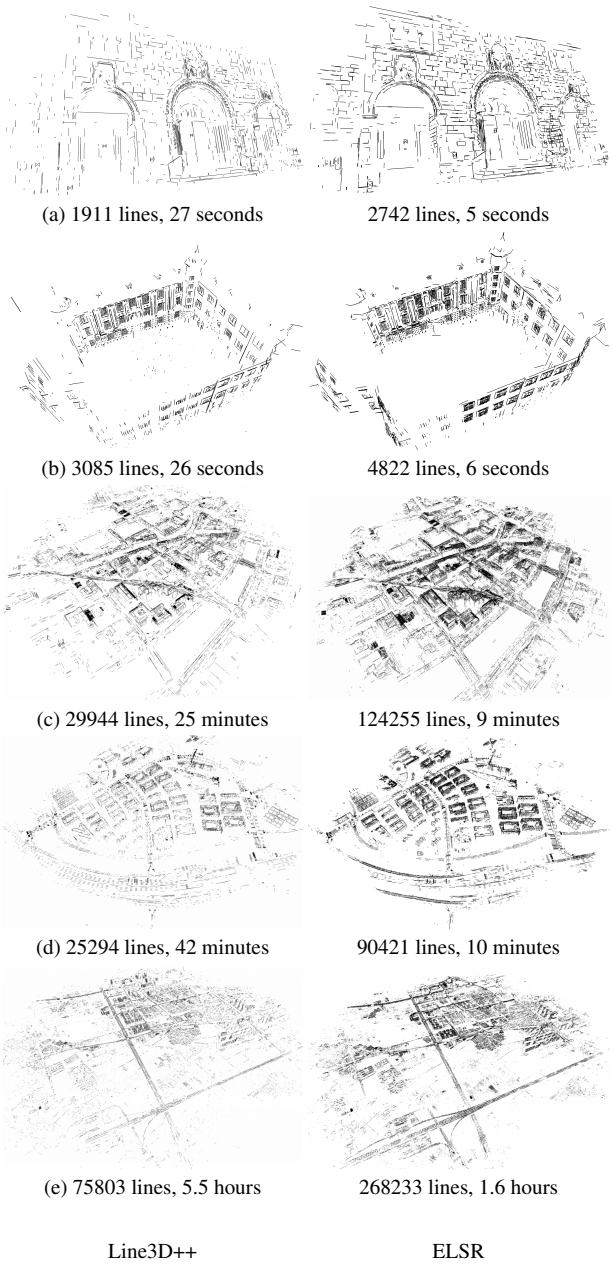


Figure 10. The quantity and time of the line reconstruction. Line3D++ used a GPU. The index is aligned with that in Fig. 7.

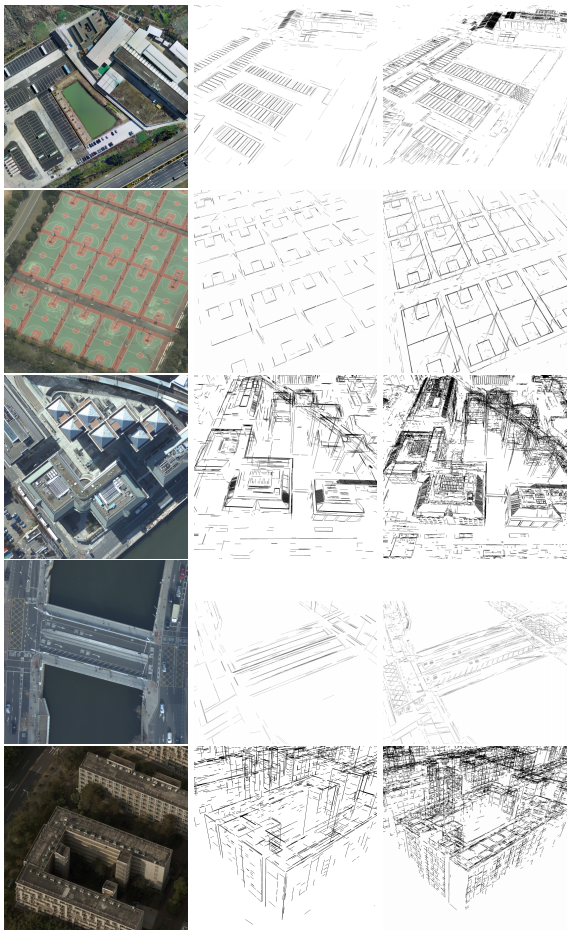


Figure 11. The 3D line segments in local areas.

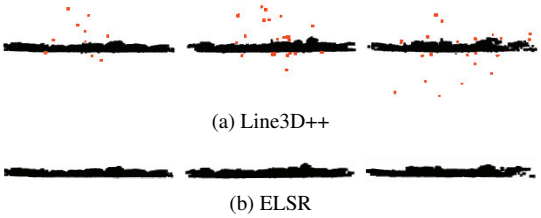
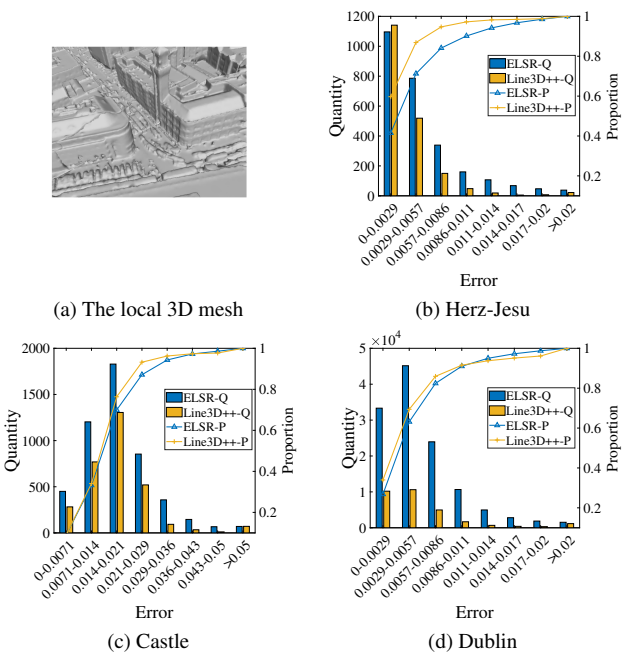


Figure 12. The profile of the 3D lines in Dublin, Guangzhou, and Ningbo. The outliers are drawn in red.

slower than ELSR. These were the expected results for the following reasons. First, Sec. 5.1 showed the line matching in ELSR to be very efficient. Second, many functions in ELSR are independent and processed in multithread. Simultaneously, Line3D++ only uses an epipolar constraint to reduce the candidates and requires at least three views to establish 3D line candidates.

5.3. Influence of the hyper-parameters

ELSR has some intuitive hyper-parameters. To explore their impacts and limitations, we altered the hyper-parameters to evaluate their influences on ELSR using the Castle dataset. The values of the positional similarity ($t_{\text{pos}} = 2$), the intersection distance ($t_{\text{int}} = 10$), and the overlap rate ($t_{\text{ove}} = 0.5$) are common parameters, and they have been studied in several past works [30] for two-view matching. Thus, we set them as the common value and ex-



(a) The local 3D mesh (b) Herz-Jesu (c) Castle (d) Dublin

Figure 13. The maximum distance from the line to the 3D mesh was computed as the error.

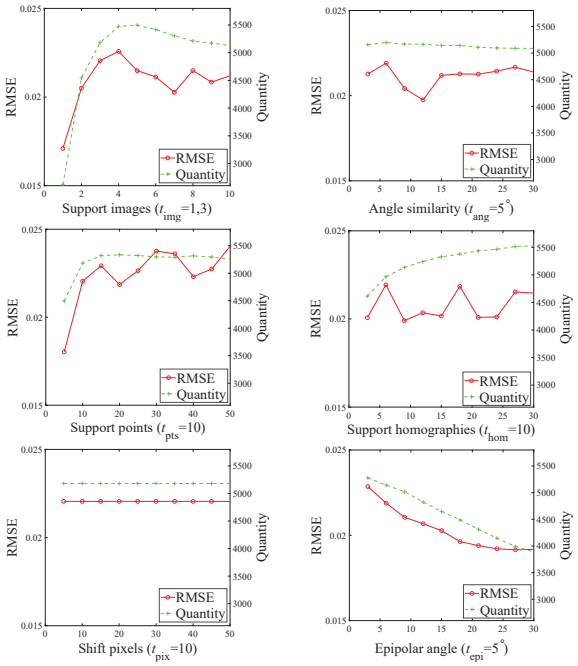


Figure 14. The evaluation with the Castle dataset by altering the hyper-parameters.

plore the influence of the other six parameters.

Fig. 14 shows the RMSE and quantity for different thresholds. The shift of pixels (t_{pix}) had a little influence on the results. Because the range of the point depth covered the line depth in most cases. Increasing the support homographies (t_{hom}) raised the quantity since it increased the possibility of finding the correct match. A loose threshold for the support points(t_{pts}) and epipolar angle (t_{epi}) brought about more 3D line segments while the accuracy generally decreased. However, this result did not hold for the angular similarity (t_{ang}). Because a loose threshold causes more false matches, and many of them will obtain a low spatial score.

Among these parameters, the number of support images (t_{img}) in building the image pair most influences the results because more image pairs produce more matches. Thus, we relied on our experience to set t_{img} as 1 and 3 for large and small image datasets, respectively.

6. Limitations

As shown in the above section, ELSR is not a parameter-free algorithm. Because it has several hyper-parameters and requires parameter-tuning for the best performance. The number of support images can significantly influence the quantity of 3D lines, and turning up this parameter will obtain more 3D lines at the cost of run time.

Also, ELSR selects representative 3D lines for better speed and robustness. Since the single 3D line is only reconstructed with two views, the positional accuracy may not be as accurate as merging the cluster. Thus, exploiting the multiple-view optimization, such as the bundle adjustment, would be a future work to improve the accuracy.

Finally, ELSR requires running point-based SfM in advance, which makes it not applicable when the point is difficult to detect.

7. Conclusions

In this paper, we proposed a novel algorithm called ELSR for efficient line segment reconstruction. ELSR can efficiently match line segments by utilizing scene planes and sparse points, and ELSR abstracts representative 3D lines that is robust to false matches. Our results show that our method was more efficient and accurate than other methods in two-view matching. Compared with the most efficient existing algorithm in line segment reconstruction (Line3D++), ELSR also showed the superiority in the speed of reconstruction and the quantity of 3D lines. In the future, we will first improve the reconstruction accuracy with multiple-view optimization, and then, exploit the accurate 3D lines to give a better building reconstruction.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 42001406 and 42030102.

References

[1] Mohammed Al-Shahri and Alper Yilmaz. Line matching in wide-baseline stereo: A top-down approach. *IEEE Transactions on Image Processing*, 23(9):4199–4210, 2014. 2

[2] Daniel Barath and Jiri Matas. Graph-cut ransac. In *CVPR*, pages 6733–6741, 2018. 6

[3] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiří Matas. Magsac++, a fast, reliable and accurate robust estimator. In *CVPR*, pages 1301–1309, 2020. 6

[4] Bentley. Contextcapture. https://www.bentley.com/en/products/brands/contextcapture. 6

[5] Wei Dong. Glm code. https://github.com/weidong-whu/line-match-RRW. 5

[6] Bin Fan. Lpi code. https://kailigo.github.io/projects/LineBenchmark/codes/LPI.zip. 5

[7] B. Fan, F. Wu, and Z. Hu. Line matching leveraged by point correspondences. In *CVPR*, 2010. 1, 2, 5

[8] Bin Fan, Fuchao Wu, and Zhanyi Hu. Robust line matching through line–point invariants. *Pattern Recognition*, 45(2):794–805, 2012. 2

[9] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010. 3, 5

[10] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. 2003. 5

[11] Manuel Hofer. Line3d++ code. https://github.com/manhofer/Line3Dpp.

[12] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding*, 157:167–178, 2017. 1, 2, 5, 6

[13] Arjun Jain, Christian Kurz, Thorsten Thormählen, and Hans-Peter Seidel. Exploiting global connectivity constraints for reconstruction of 3d line segments from images. In *CVPR*, pages 1586–1593, 2010. 2, 5, 6

[14] Li KAI. Ljl code. https://github.com/kailigo/LineSegmentMatching. 5

[15] C. Kim and R. Manduchi. Planar structures from line correspondences in a manhattan world. In *ACCV*, 2014. 1

[16] Laefer. 2015 aerial laser and photogrammetry survey of dublin city collection record. https://geo.nyu.edu/catalog/nyu_2451_38684. 6

[17] Kai Li and Jian Yao. Line segment matching and reconstruction via exploiting coplanar cues. *ISPRS Journal of Photogrammetry and Remote Sensing*, 125:33–49, 2017. 1, 2, 5

[18] M.I.A Lourakis, S.T Halkidis, and S.C Orphanoudakis. Matching disparate views of planar surfaces using projective invariants. *Image and Vision Computing*, 18(9):673–683, 2000. 2

[19] DG Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, NOV 2004. 6

[20] Pierre Moulon, Pascal Monasse, Renaud Marlet, et al. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *ICCV*, 2013. 2

[21] Ali Ozgun Ok, Jan Dirk Wegner, Christian Heipke, Franz Rottensteiner, Uwe Soergel, and Vedat Toprak. Matching of straight line segments from aerial stereo images of urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 74:133–152, 2012. 1, 2

[22] Srikumar Ramalingam, Michel Antunes, Daniel Snow, Gim Hee Lee, and Sudeep Pillai. Line-sweep: Cross-ratio for wide-baseline matching and 3d reconstruction. In *CVPR*, pages 1238–1246, 2015. 2

[23] C. Schmid and A. Zisserman. Automatic line matching across views. In *CVPR*, pages 666–671, 1997. 1, 2, 3

[24] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008. 2

[25] Strecha. Strecha dense mvs. https://documents.epfl.ch/groups/c/cv/cvlab-unit/www/data/multiview/denseMVS.html.. 6

[26] Takayuki Sugiura, Akihiko Torii, and Masatoshi Okutomi. 3d surface reconstruction from point-and-line cloud. In *International Conference on 3D Vision*, pages 264–272, 2015. 1

[27] T. Sugiura, A. Torii, and M. Okutomi. 3d surface reconstruction from point-and-line cloud. In *International Conference on 3D Vision*, 2015. 1

[28] Yanbiao Sun, Liang Zhao, Shoudong Huang, Lei Yan, and Gamini Dissanayake. Line matching based on planar homography for stereo aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:1–17, 2015. 2

[29] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. Msld: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, 2009. 2

[30] Dong Wei, Yongjun Zhang, and Chang Li. Robust line segment matching via reweighted random walks on the homography graph. *Pattern Recognition*, 111:107693, 2021. 1, 2, 3, 5, 7

[31] Dong Wei, Yongjun Zhang, Xinyi Liu, Chang Li, and Zhuofan Li. Robust line segment matching across views via ranking the line-point graph. *ISPRS Journal of Photogrammetry and Remote Sensing*, 171:49–62, 2021. 2, 5

[32] Changchang Wu. Towards linear-time incremental structure from motion. In *International Conference on 3D Vision*, pages 127–134, 2013. 2, 5

[33] Lilian Zhang and Reinhard Koch. 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):794–805, 2013. 2