This CVPR2015 paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

# Line-Sweep: Cross-Ratio for Wide-Baseline Matching and 3D Reconstruction

Srikumar Ramalingam[1] Michel Antunes[2] Daniel Snow[1] Gim Hee Lee[1] Sudeep Pillai[3]
[1]Mitsubishi Electric Research Laboratories (MERL), Cambridge, USA
[2]Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
[3]Massachussetts Institute of Technology (MIT), Cambridge, USA

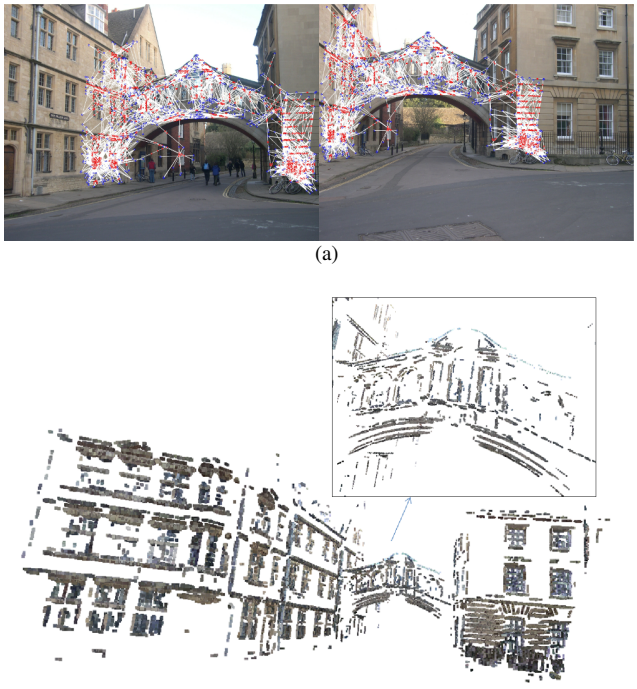
## Abstract

*We propose a simple and useful idea based on cross-ratio constraint for wide-baseline matching and 3D reconstruction. Most existing methods exploit feature points and planes from images. Lines have always been considered notorious for both matching and reconstruction due to the lack of good line descriptors. We propose a method to generate and match new points using virtual lines constructed using pairs of keypoints, which are obtained using standard feature point detectors. We use cross-ratio constraints to obtain an initial set of new point matches, which are subsequently used to obtain line correspondences. We develop a method that works for both calibrated and uncalibrated camera configurations. We show compelling line-matching and large-scale 3D reconstruction.*

## 1. Introduction

3D reconstruction from images using points is one of the success stories in computer vision [36, 40, 13, 10]. The success can be attributed to several well-researched and well-engineered tools: the availability of a variety of feature point descriptors, the key insight to use millions of uncalibrated images from wild, RANSAC, 5-point motion estimation [27] and optimization packages [1, 40]. As shown in Figure 1, lines are dominant in most urban scenes. However, lines are not used in 3D reconstruction algorithms as much as points and planes. Although numerous fundamental results have been derived on line reconstruction, these techniques are seldom applied in practice. The primary reason is the lack of good line descriptors and the noise in line detection algorithms. This makes us wonder if line-reconstruction can leverage from the existing machinery for points. We address this question by proposing a novel approach to transfer the point correspondences to line correspondences.

The basic idea proposed in this paper is simple. Consider Figure 1(a). We show two images of an urban scene with keypoint matches marked in blue obtained using standard feature detectors. By considering pairs of such keypoint matches, we can form virtual lines across the images. By looking at places where they intersect straight lines in images, and using cross-ratio constraints, we match pixels lying on one image line to another. By accumulating such correspondences (shown in red) we can match line segments. Note that many prior line detection algorithms only match lines from one image to another. There is no pixel-wise correspondence between lines. In our approach



(a) (b)

Figure 1. *The basic idea of line-sweeping. (a) We consider pairs of keypoint correspondences (shown in blue) obtained using standard feature detectors to form virtual lines (white) to identify several hundreds of additional point matches (shown in red) using cross-ratio constraints. (b) Using 32 images, we generate line-based 3D reconstruction of the bridge of Sighs in Oxford, where lines are represented as pencil of points.*

we compute pixel-wise correspondence between line segments. We use these pixel-wise correspondences to obtain a semantically more meaningful 3D reconstruction as shown in Figure 1(b).

## 2. Related Work

3D Reconstruction from lines is one of the oldest topics in computer vision [22, 37]. As with many vision problems, the earliest formulation of this problem was over-ambitious with the goal of obtaining line reconstruction from a single image. Several geometrical and constraint satisfaction algorithms were proposed to solve this problem and these techniques are mostly applied on synthetic line drawings. In the context of multi-view geometry, several algorithms were proposed for matching and reconstruction of lines using trifocal tensors [16]. While single-view line reconstruction is still a challenging theoretical problem, the case of multi-view is more-or-less solved in the geometrical sense. However, the challenges in real images are completely different. Both the classical and purely geometrical approaches relied on the fact that the lines are detected up to sub-pixel accuracy and matched without outliers. In contrast to points, the descriptors for lines mostly rely on nearby points and are not robust in matching lines across images. The issues in detection and matching of lines lead to severe degradation of the final line reconstruction. While 3D reconstruction from points can be done from random internet images with unknown camera parameters, line reconstruction still require careful calibration to obtain any useful results.

Our work builds on top of many existing line matching and reconstruction algorithms in the literature. Hartley and many others studied the trifocal tensor constraints and degeneracies involved in the process of line reconstruction from three views [16]. Schmid and Zisserman showed a method for matching lines from two or three images using cross-correlation score from the neighborhood of a line [34]. There are several line matching algorithms in the literature and most or all of them use nearby points or color to match the lines [12]. A recent survey can provide more details about several line descriptors in the literature [38]. The cross-ratio constraint has been used before for matching points. For example, Duchenne et al [7] finds matches for given keypoints using cross-ratio constraint. In particular, the cross-ratio constraint is used to enforce projective invariance on three non-collinear points while matching. Our approach, on the other hand, finds new point matches using cross-ratio and all these points lie on a single line.

Bartoli et al. [2] proposed a method to solve the 3D reconstruction of lines by considering pencil of points (POPs) on lines. Many line matching and reconstruction algorithms match infinite lines and reconstruct them using intersection of planes. On the other hand, [2] use explicit pixel-wise correspondences for individual points on lines. Our work is closely related to this work and we also compute correspondences for points lying on lines. Our main contribution is an accurate and efficient method to compute correspondences among points on lines. There are a few single-view line reconstruction algorithms that use Manhattan or Atlanta world priors [33, 29, 30], but we do not use these priors. Jain et al. [20] showed that connectivity constraints can be very useful for obtaining accurate line reconstruction from multiple images. Hofer et al. [19] showed impressive 3D reconstruction of wiry objects using a semi-global line reconstruction technique. Many of these approaches solve an optimization problem for various locations of 3D line segments to best match its projections. Recently, Micusik and Wildenauer [25] showed large-scale line-based 3D reconstruction of indoor scenes. Lines have also been leveraged for image-based localization [28, 26].

There are many tracking based edge/line reconstruction algorithms [6, 3, 8, 9]. In particular LSD-SLAM [9] showed impressive monocular SLAM results by reconstructing image patches that correspond to edges. The main difference between our framework and LSD-SLAM is the same difference that exist between Bundler [36] and PTAM [23]. Our method is slower, but can handle wide-baseline images. LSD-SLAM is faster, but cannot handle wide-baseline images.

At a cursory glance, the use of cross-ratio constraint may appear very similar to many cross-correlation methods [34] used in line matching. Most prior methods match lines by looking at intensity and color profiles in the local neighborhood or in patches close to the line. Our method uses a very different approach by considering pairs of points that can be anywhere in the image. Despite the apparent simplicity, the proposed idea is very useful to transfer point correspondences to line correspondences. Computationally, our approach also provides a strong advantage. By using two anchor points from keypoint matches, we also significantly reduce the search space.

The virtual lines joining pairs of keypoints allow us to match points from one image to another. This reminds us of the popular plane-sweeping technique, where we use virtual planes to compute depth maps from multiple images [5]. We will refer to our approach as *line-sweeping*. In addition to plane-sweeping, there are many impressive dense reconstruction methods such as PMVS [13], SURE [31], and CMPMVS [21]. Most of these prior method require calibration. The use of cross-ratio, which is one of the projective invariants, allows us to obtain dense correspondences even in uncalibrated setups without any camera calibration and relative camera poses.

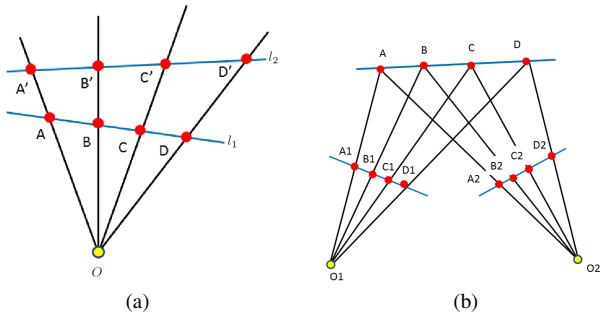
## 3. Algorithm

We briefly define cross-ratio below.

**Cross-Ratio:** In projective geometry, cross-ratio is one of the fundamental invariants [35, 17]. In Figure 2 we show a pencil of lines from center $O$ intersecting a line $l_1$ at four points $(A, B, C, D)$. The same pencil of lines also intersect another line $l_2$ at four other points $(A', B', C', D')$. The cross-ratio for the four collinear points on $l_1$ is defined as the following quantity:

$$\{A, B; C, D\} = \frac{|AC| \times |AD|}{|BC| \times |BD|}. \quad (1)$$

We can compute a cross-ratio $\{A', B'; C', D'\}$ using the four collinear points on line $l_2$. By using the property of invariance of cross-ratio, we have $\{A, B; C, D\} = \{A', B'; C', D'\}$. In the case of perspective projection, we have cross-ratio from four collinear points observed from different viewpoints as shown in Figure 2(b). Here we have $\{A_1, B_1; C_1, D_1\} = \{A_2, B_2; C_2, D_2\}$.




(a) (b)

Figure 2. *(a) Classical cross-ratio constraint in projective geometry. Sets of four points have same cross-ratios on the incident pencil of lines as shown. (b) In two different perspective projections observing the same set of four collinear points, the associated cross-ratios are same.*

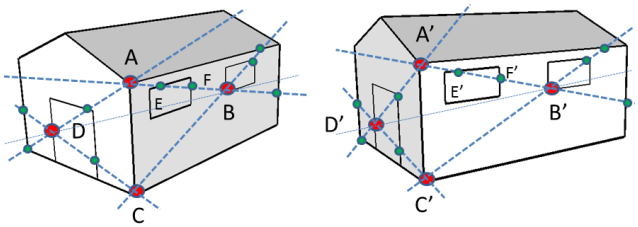



Figure 3. *We show two perspective images, taken from different viewpoints, with 4 initial keypoint matches (shown in red) obtained using standard feature detectors. By choosing pairs of keypoint matches, we can form virtual lines (dotted lines) where we can search for new point correspondences. By choosing several virtual lines, we generate 9 new point matches (shown in green).*

**Our main idea:** We refer to the lines obtained by joining pairs of keypoint matches as virtual lines. It is important to distinguish them from real lines in the image such as the ones in the rectangular windows in Figure 3. For example, the line joining A and B is referred to as a virtual line. Given a pair of keypoint matches, we consider a virtual line joining them to generate new point correspondences. The new points are first generated at regions where the virtual lines intersect the real lines. These newly generated points on the virtual lines are matched based on the cross-ratio constraint. In Figure 3, we show 4 initial point matches (marked in red) in two perspective images taken from different viewpoints. These 4 initial point matches can be obtained using feature descriptors. We consider several pairs of keypoint matches to generate newer ones. It can be observed that by using as few as 4 point matches, we are able to obtain 9 new matches (marked in green). In real images with numerous lines and keypoints, we typically have a combinatorial number of virtual lines and additional points.

We illustrate our algorithm for three different camera setups: uncalibrated, calibrated and rectified stereo configurations.

### 3.1. Uncalibrated cameras

Given keypoint matches between two images, we study the problem of generating new point correspondences, with no knowledge of the intrinsic and extrinsic camera parameters. Consider a pair of point matches $\{(A, A'), (B, B')\}$ as shown in Figure 3. Let the virtual line passing through $A$ and $B$ be denoted by $AB$. We refer to the point where a virtual line intersects a real line as a line-crossing. For example, points $E$ and $F$ are referred to as the line-crossings on $AB$.

We use a simple search strategy to identify point matches on pairs of virtual lines using cross-ratio property. Let us hypothesize one additional match $(E, E')$ using one line-crossing each in $AB$ and $A'B'$. Using the three point matches $\{(A, A'), (B, B'), (E, E')\}$, we can identify all other additional matches. In order to do this, we first compute the cross-ratio $\{A, B; E, F\}$ for a line-crossing $F$ lying on $AB$. Using this point $F$ and the computed cross-ratio $\{A, B; E, F\}$, we can find the corresponding point $F'$ on $A'B'$ by assuming that $\{A, B; E, F\} = \{A', B'; E', F'\}$. We consider the pair $(F, F')$ to be a match if $F'$ is a line-crossing on $A'B'$. In the same manner, we search for all additional matches that satisfy the cross-ratio constraint using the hypothesized match $(E, E')$. Our goal is to find one additional match $(E, E')$ that generates the maximum number of newer matches on the corresponding virtual lines $AB$ and $A'B'$. For identifying this one additional match, we can have $n^2$ possibilities where $n$ is the number of line-crossings on $AB$. While generating point matches on a pair of virtual lines, we also ensure that we identify a minimum number of point correspondences given by a search threshold $\mathcal{T}_{uc}$. We assume that the corresponding points on virtual lines are ordered and this reduces the number of possibilities significantly.

### 3.2. Calibrated cameras

In the presence of camera calibration and relative motion between the cameras, the task of generating new point correspondences becomes much simpler and computationally efficient. Let us consider a pair of feature point matches $\{(A, A'), (B, B')\}$ as shown in Figure 4. Since we can compute the depth using calibration information, we can also compute the 3D points $P(A)$ and $P(B)$. We compute the 3D point $P(C)$ for the line-crossing $C$ by assuming that $P(C)$ lies on the 3D line $P(A)P(B)$. Let $C'$ be projection of $P(C)$ on $A'B'$. We consider the pair $(C, C')$ to be a match if $C'$ is a line-crossing on $A'B'$. While generating point matches on a pair of virtual lines, we ensure that we identify a minimum number of matches given by a search threshold $\mathcal{T}_c$. The complexity is $O(n)$ on the number of line-crossings on the virtual line. This task is faster than the search in the uncalibrated case.

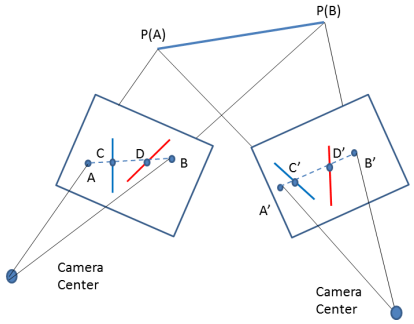



Figure 4. *We show the basic idea for line-sweep operation in a calibrated setup. We know the camera calibration and relative motion between the two cameras. This allows us to find points matches $(C, C')$ and $(D, D')$ very efficiently.*

### 3.3. Stereo cameras

Consider Figure 5. In the case of rectified stereo-pairs, line-sweep is a simple task. Let us consider two corresponding virtual lines $AB$ and $A'B'$. We consider the pair $(C, C')$ to be a match if it satisfies the following conditions:

- $C$ and $C'$ are line-crossings on $AB$ and $A'B'$ respectively.
- $C$ and $C'$ have the same $y$ coordinate.

### 3.4. Line matching

In Figure 6, we illustrate the individual steps in our line-matching algorithm. Given a pair of images, we obtain keypoint matches and detect lines. We generate and match new points on image lines using line-sweep. Let $l_1$ and $l_2$ be lines in images $I_1$ and $I_2$ respectively. We declare a pair of lines $(l_1, l_2)$ to be a match if they satisfy the following:


Figure 5. *The basic idea behind line-sweep in a stereo case. In the case of rectified stereo images, finding additional point correspondences is a simple look-up without requiring any additional operation.*

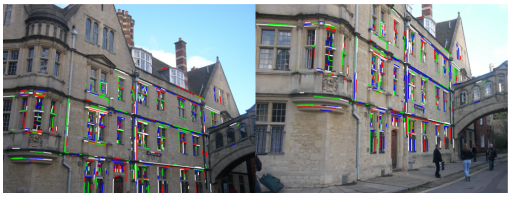
1. Among all the lines in the second image $I_2$, $l_2$ is the line that shares the maximum number of point matches with $l_1$.

2. Similarly, $l_1$ is the line in the first image $I_1$ that shares the maximum number of point matches with $l_2$.

3. The pair $(l_1, l_2)$ share at least a minimum number of point matches based on a threshold $\mathcal{T}_{lm}$.

(a)

(b)



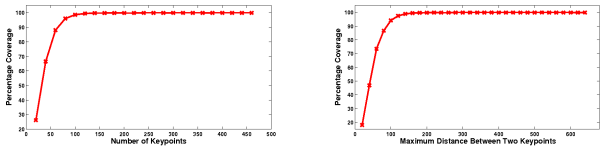
(c)

Figure 6. *We show the steps involved in obtaining line-matching. (a) We show a pair of wide-baseline images. (b) We show the new points (red) generated on the image lines using cross-ratio constraints along the virtual lines. (c) We show the line matches. For simplicity, we show the lines in four different colors (white, red, blue and green).*

# 4. Experiments

## 4.1. Scene coverage

Let us consider a pair of images having $n$ keypoint matches between them. Our approach using pairs of points would lead to $n^2$ possible virtual lines. Note that these lines extend till the boundary of the image to obtain new point correspondences as shown in Figure 3.




(a) (b)

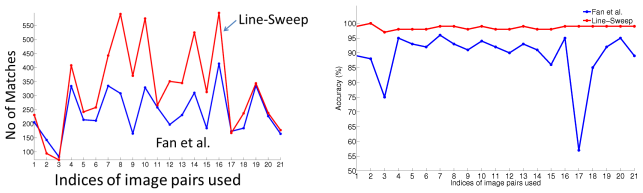
Figure 7. *Coverage Analysis: (a) Average percentage image coverage achieved by virtual lines passing through different number of image keypoints. (b) Average percentage image coverage with different maximum allowable distances (pixels) between two image keypoints.*

Using the first experiment, we show that we can adequately cover the whole image with virtual lines using very few keypoints. We select $n$ SIFT keypoints on an image and look at all possible virtual lines from pairs of keypoints. We assume that each virtual line is of width 1 pixel. We used images of resolution $2048 \times 768$. The total coverage is computed by summing the pixels in the image that was part of at least one virtual line. Figure 7(a) shows a plot of the average percentage coverage against the number of keypoints. We can see from the plot in Figure 7(a) that almost $99.9\%$ of an image is covered using only 150 image keypoints.

In the second experiment, we would like to understand the best strategy to choose pairs of keypoints. We assume that nearby points are more likely to lie on a plane than points that are separated. We fixed the number of keypoints to be 200. We compute the percentage coverage for different maximum allowable distance between two image keypoints, i.e. we consider a virtual line only when the distance between the two image keypoints is less than the maximum allowable distance. Similar to the first experiment, the average percentage coverage is reported in Figure 7(b). We can see from the plot that almost $99.9\%$ of the image is covered when the maximum allowable distance is less than 200 pixels. As per this analysis, in all our real experiments, we only consider a pair of keypoints that are not separated by a large distance.

## 4.2. Line-matching

We use the recently proposed line matching algorithm of Fan et al. [11] as baseline. We use the publicly available code provided by the authors. The comparative results are shown in Figure 8. The line-sweep algorithm provides

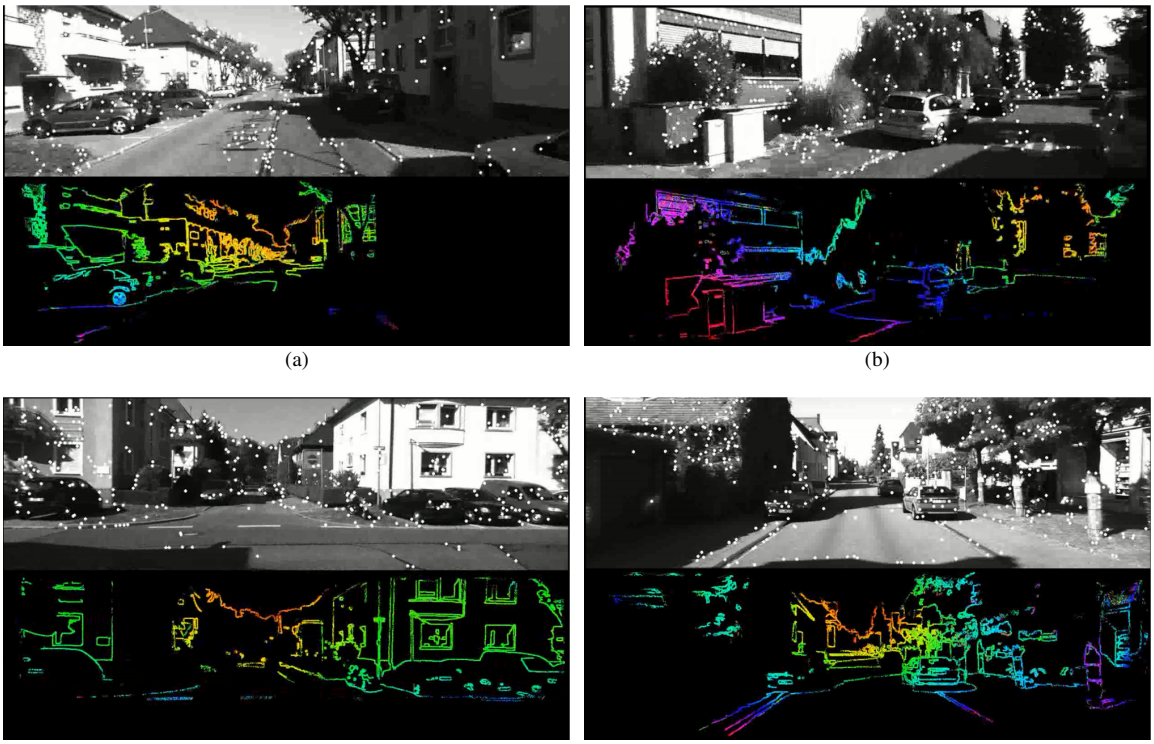



(a) Number of matched lines (b) Accuracy of line matching

Figure 8. *Comparison between Fan et al. [11] and line-sweep. (a) We show the number of line matches in different image pairs. (b) We show the accuracy in percentage for line matching in different image pairs.*

Figure 9. *We show examples of our line matching algorithm. The lines are colored red, blue and green.*

a higher number of matches in many examples and consistently outperforms Fan et al. [11] in terms of accuracy. Our method is computationally more efficient than Fan et al. [11]. Most of the images in our test set are high resolution images (at least $2048 \times 768$). Our calibrated setup takes about 1 to 2 seconds on most images when compared to around 2 minutes required by [11]. In our experiments, the pair of lines need to share at least $\mathcal{T} = 5$ point matches between them. In the case of uncalibrated setup, we took less than 10 seconds on most image pairs. We show a few examples of our line-matching results in Figure 9. As shown in the graph 8, our average matching accuracy is 98%, whereas [11] gets an accuracy of 90%. The calibrated line-matching is much faster than the uncalibrated one.



(a) (b) (c) (d)

Figure 10. *We use the stereo-based line-sweeping algorithm on KITTI sequences. On top, we show one of the images in the stereo pair with detected ORB features shown as white dots. We use the canny edge map and obtain semi-dense disparity map for these edge pixels using line-sweep. The disparity is shown in LSV color space.*

### 4.3. Semi-dense stereo reconstruction

We used KITTI stereo sequences [14] to test our stereo method explained in section 3.3. Instead of using line segments, we used canny edges. We show the results in Figure 10. As explained before, the stereo case is a special case and the method is computationally efficient.

Our approach computes the disparity values of all edge pixels based on the disparities of nearby keypoints. In order to evaluate the proposed method, we compared the results with that of semi-global stereo (SGBM) results [18]. We found that a large subset (85% of edge pixels) had a disparity value that is within 2 pixels with respect to the SGBM. The average disparity difference for this subset of pixels is given by 0.28. We did two refinements to the disparity obtained using our line-sweep algorithm. First we did a simple SAD correlation to refine the disparity values based on an interval (about 5 pixels on both sides) centered on the disparity value estimated using line-sweep. Following this, we did a subpixel refinement based on quadratic bowl approximation. As a result of these refinements, we found that 95% of edge pixels were within 2 pixels with respect to the disparities given by SGBM. The average disparity difference for this subset of pixels is given by 0.10. We tested our C++ code on an Intel(R) Core(TM) i7-3920XM CPU @ 2.90GHz. We use ORB keypoint detector and descriptors [32] for speed and this takes around 50 milliseconds for obtaining 1000 keypoints. We take only 20 milliseconds for most of our code (line-sweep, disparity search using SAD, and quadratic approximation). Overall the disparity estimation code runs at 14 Hertz.

### 4.4. Large scale line-reconstruction

We report the results on four data sets. For keypoint detection and matching we used SIFT. For line detection, we used LSD [15]. Our matching algorithm was tested on images of resolution $2272 \times 1704$ in both calibrated and uncalibrated scenarios. The code is implemented in C++ and openCV libraries without any optimization for speed. In the calibrated case, our approach takes on an average less than 1 second for matching. In the uncalibrated case, our approach takes on an average less than 10 seconds for matching. We use a search threshold $\mathcal{T}_{uc} = 5$ for uncalibrated case and a search threshold $\mathcal{T}_c = 3$ for calibrated case. After computing the line correspondences, we continue to use pencil of points for bundle adjustment. The line matching is only used to filter out noisy points by keeping only the point matches that respect the 1-1 line correspondences. By formulating the problem this way, we are able to use the ex-

| Dataset | Imgs | Sparse Pts | Dense Pts | Lines |
|---|---|---|---|---|
| Oxford | 32 | 58K | 462K | 1245 |
| Barcelona | 28 | 33K | 154K | 2534 |
| Quadrangle | 109 | 16K | 575K | 9201 |
| CAB | 35 | 37K | 333K | 6193 |

Table 1. The consecutive columns denote the data set, the number of images, the number of sparse reconstructed points using standard approach, the number of points in our dense reconstruction, and finally the number of lines reconstructed, respectively.

isting point-based bundle adjustment tools for our line reconstruction. Furthermore, pencil of points approach can be more robust to line detection errors, without leading to noisy triangulation for the 3D lines.

We gathered two data sets (109 images of a Quadrangle, 32 images in Oxford) and we used two other data sets CAB and Barcelona from [4]. The Quadrangle dataset with 109 images comprises of a very large loop as shown in Figure 11 and we obtained loop closure using pose-graph optimization [24].

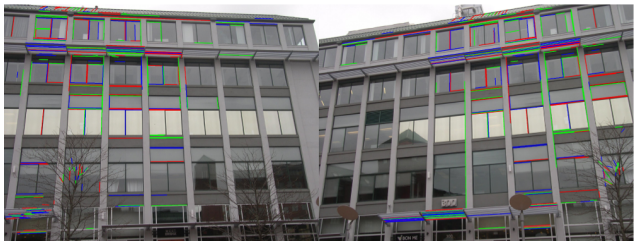
For testing our uncalibrated case, we first compute the dense matches from sparse SIFT correspondences. Then we perform bundle adjustment using multicore libraries [40]. For the calibrated case, we used *VisualSFM* [39, 40] to generate the point-based 3D models. For testing the calibrated case, we wrote a solver that can directly take the output files from VisualSFM and convert it into line-based models using our approach described in 3.2. In the Table 1, we report the statistics collected from our experiments. Qualitatively, we obtain a semantically more-meaningful and compact 3D models as shown in Figures 11 and 12. In the Supplementary videos, we show the 3D models reconstructed using our approach.

## 5. Discussion

We show a novel method to use cross-ratio constraints for mapping point matches to line correspondences. We show accurate line-matching performance as well as large-scale line reconstruction by leveraging on the existing tools for points. We show our results for line reconstruction as point clouds denoting pencil of points (POPs). We show a method to convert point-cloud from Bundler [36] to line-based models.

In several image pairs with repeated texture and numerous incorrect keypoint matches, our line matching still gives near-perfect matching as shown in Figure 9. In Figure 13, we observe a challenging image pair which suffers from both repeated patterns and reflections. Since our line-matching hinges on keypoint matches, it fails if there are too many incorrect keypoint matches. In future, we plan to explore this avenue to improve the robustness of our line-matching algorithm to handle such challenging cases.

**Acknowledgments:** We thank J. Thornton, Y. Taguchi, M. Brand, M. Liu, ACs and anonymous reviewers for useful discussions and constructive feedback.



(a)

Figure 13. *Failure case: A challenging image pair with too many reflections and repeated patterns. The keypoint matching fails and this leads to failure in line-matching.*

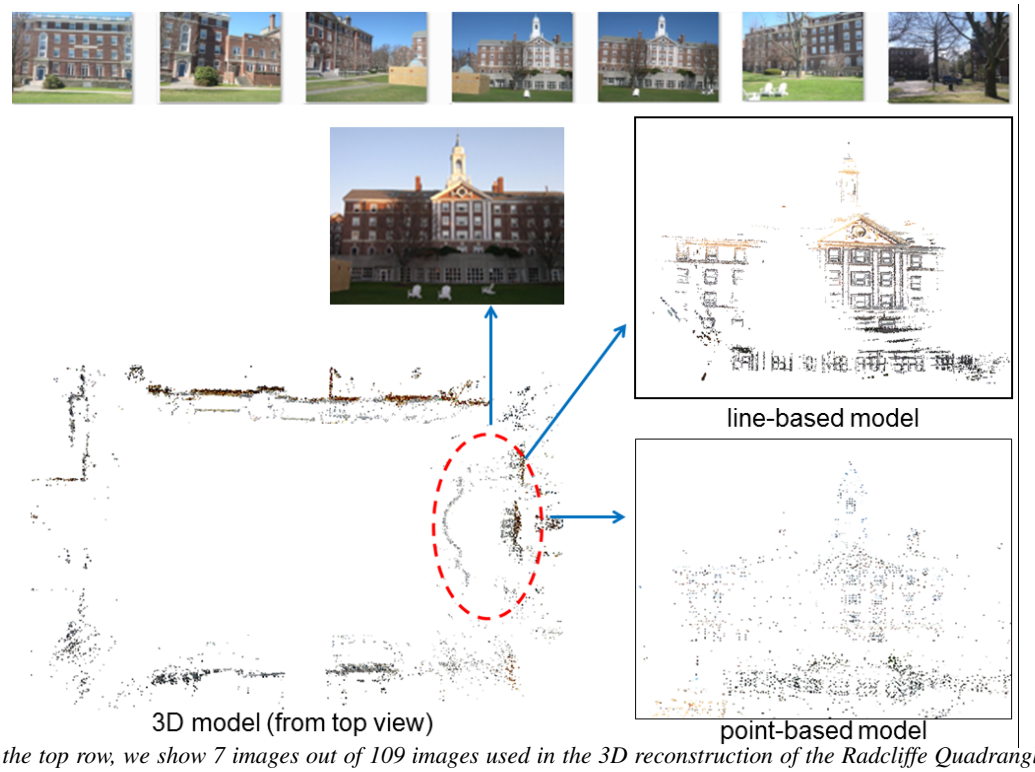



Figure 11. *In the top row, we show 7 images out of 109 images used in the 3D reconstruction of the Radcliffe Quadrangle in Cambridge (USA). In the bottom row, we show an aerial view of the 3D model of a Quadrangle. For a small region denoted by the red ellipse, we show the corresponding image, and 3D models associated with both line-based and point-based ones. As we can see, the line-reconstruction looks semantically more meaningful compared to the point-based one.*

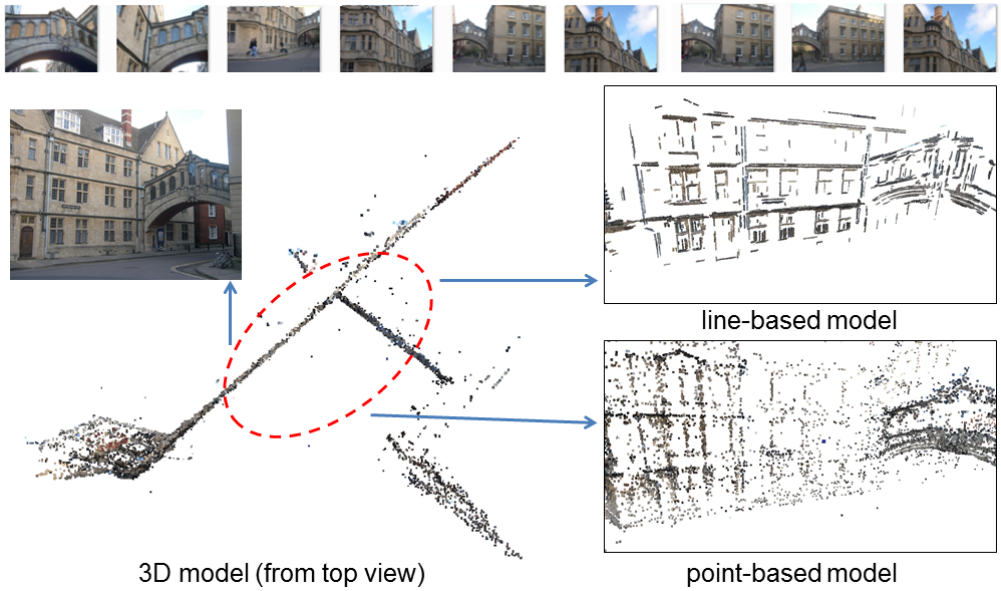



Figure 12. *In the top row, we show 9 images out of 32 images used in the 3D reconstruction of the Bridge of Sighs in Oxford (UK). In the bottom row, we show an aerial view of the 3D model of a Quadrangle. For a small region denoted by the red ellipse, we show the corresponding image, and 3D models associated with both line-based and point-based ones. As we can see, the line-reconstruction looks semantically more meaningful compared to the point-based one.*

## References


[1] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org. 1

[2] A. Bartoli, M. Coquerelle, and P. Sturm. A framework for pencil-of-points structure-from-motion. In *ECCV*, 2004. 2

[3] M. Chandraker, J. Lim, and D. Kriegman. Moving in stereo: Efficient structure and motion using lines. In *ICCV*, 2009. 2

[4] A. Cohen, C. Zach, S. N. Sinha, and M. Pollefeys. Discovering and exploiting 3d symmetries in structure from motion. In *CVPR*, 2012. 7

[5] R. Collins. A space-sweep approach to true multi-image matching. In *CVPR*, 1996. 2

[6] J. Crowley, P. Stelmaszyk, T. Skordas, and P. Puget. Measurement and integration of 3-d structures by tracking edge lines, 1992. 2

[7] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *PAMI*, 2011. 2

[8] E. Eade and T. Drummond. Edge landmarks in monocular slam. In *BMVC*, 2006. 2

[9] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, 2014. 2

[10] J.-M. F. et al. Building rome on a cloudless day. In *ECCV*, 2010. 1

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

[12] V. Ferrari, T. Tuytelaars, and L. V. Gool. Wide-baseline multiple-view correspondences. In *CVPR*, 2003. 2

[13] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *PAMI*, 2010. 1, 2

[14] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 6

[15] R. Gioi, J. Jakubowicz, J. Morel, and G. Randall. Lsd: a line segment detector. *IPOL*, 2012. 6

[16] R. Hartley. Lines and points in three views and the trifocal tensor. In *IJCV*, 1997. 2

[17] R. Hartley and A. Zisserman. *Multiview geometry in computer vision*. Cambridge University Press, 2004. 3

[18] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR*, 2005. 6

[19] M. Hofer, M. Donoser, and H. Bischof. Semi-global 3d line modeling for incremental structure-from-motion. In *BMVC*, 2014. 2

[20] A. Jain, C. Kurz, T. Thormahlen, and H. Seidel. Exploiting global connectivity constraints for reconstruction of 3d line segment from images. In *CVPR*, 2010. 2

[21] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*, 2011. 2

[22] T. Kanade. A theory of origami world. *AI*, 1980. 2

[23] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, 2007. 2

[24] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Robust pose-graph loop-closures with expectation-maximization. In *IROS*, 2013. 7

[25] B. Micusik and H. Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. In *3DV*, 2014. 2

[26] B. Micusik and H. Wildenauer. Descriptor free visual indoor localization with line segments. In *CVPR*, 2015. 2

[27] D. Nister. An efficient solution to the five-point relative pose. In *PAMI*, 2004. 1

[28] S. Ramalingam, S. Bouaziz, and P. Sturm. Pose estimation using both points and lines for geo-localization. In *ICRA*, 2011. 2

[29] S. Ramalingam and M. Brand. Lifting 3d manhattan lines from a single image. In *ICCV*, 2013. 2

[30] S. Ramalingam, J. Pillai, A. Jain, and Y. Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *CVPR*, 2013. 2

[31] M. Rothermel, K. Wenzel, D. Fritsch, and N. Haala. Sure: Photogrammetric surface reconstruction from imagery. In *LC3D Workshop*, 2012. 2

[32] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efcient alternative to sift or surf. In *ICCV*, 2011. 6

[33] G. Schindler and F. Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *CVPR*, pages 203–209, 2004. 2

[34] C. Schmid and A. Zisserman. Automatic line matching across views. In *CVPR*, 1997. 2

[35] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Oxford University Press, New York, 1998. 3

[36] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *SIGGRAPH*, 2006. 1, 2, 7

[37] K. Sugihara. *Machine Interpretation of Line Drawings*. MIT Press, 1986. 2

[38] B. Verhagen, R. Timofte, and L. V. Gool. Scale-invariant line descriptors for wide baseline matching. In *WACV 2014*, 2014. 2

[39] C. Wu. Towards linear-time incremental structure from motion. In *3DV*, 2013. 7

[40] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *CVPR*, 2011. 1, 7