Mapping of Sparse 3D Data using Alternating Projection

Siddhant Ranade*[1], Xin Yu*[1], Shantnu Kakkar[2], Pedro Miraldo[3], and Srikumar Ramalingam[1]

[1] University of Utah, Salt Lake City UT 84112, USA
{sidra,xiny,srikumar}@cs.utah.edu
[2] Trimble, Sunnyvale CA 94085, USA shantnu_kakkar@trimble.com
[3] Instituto Superior Técnico, University of Lisboa, Portugal
pedro.miraldo@tecnico.ulisboa.pt

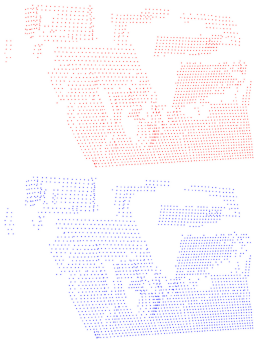
Abstract. We propose a novel technique to register sparse 3D scans in the absence of texture. While existing methods such as KinectFusion or Iterative Closest Points (ICP) heavily rely on dense point clouds, this task is particularly challenging under sparse conditions without RGB data. Sparse texture-less data does not come with high-quality boundary signal, and this prohibits the use of correspondences from corners, junctions, or boundary lines. Moreover, in the case of sparse data, it is incorrect to assume that the same point will be captured in two consecutive scans. We take a different approach and first re-parameterize the point-cloud using a large number of line segments. In this re-parameterized data, there exists a large number of line intersection (and not correspondence) constraints that allow us to solve the registration task. We propose the use of a two-step alternating projection algorithm by formulating the registration as the simultaneous satisfaction of intersection and rigidity constraints. The proposed approach outperforms other top-scoring algorithms on both Kinect and LiDAR datasets. In Kinect, we can use 100X downsampled sparse data and still outperform competing methods operating on full-resolution data.

Keywords: LiDAR, 3D registration, line intersection, generalized relative pose estimation

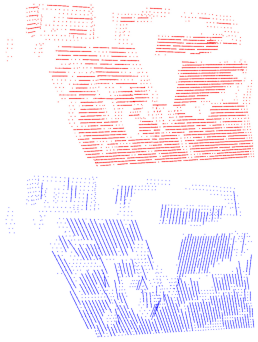
1 Introduction

The last few years have witnessed the rise of inexpensive 3D sensors for both indoor (e.g., Microsoft Kinect) and outdoor scenes (e.g., Velodyne's VLP-16 LITE LIDAR, ZED stereo camera). The algebraic centerpiece of all 3D problems is the 3D point cloud registration. This problem is well studied in the case of dense settings and scenarios with point correspondences extracted from RGB

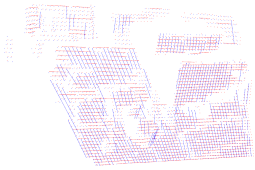
* indicates equal contribution.



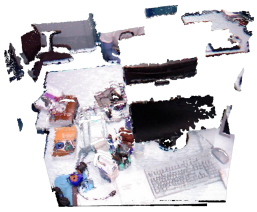
(a) *Original point clouds from two 3D scans, down-sampled by 100x*



(b) *3D straight lines fitted from the down-sampled frames (a).*



(c) *Aligned lines in (b).*



(d) *Aligned point-cloud.*

Fig. 1: *Illustration of the proposed method. (a) shows 100× down-sampled Kinect depth data (64×48). We fit 3D line segments to rows in the first and columns in the second frames as shown in (c). We identify line-to-line intersection correspondences and use alternating projection [1] to get the 3D scan alignment (c). (d) shows the registered point-cloud in original resolution with color. The average error is 0.53 degrees for rotations, and 7.58 mm for the translation, an improvement of 1.82x for the translation and 1.35x for the rotation, compared to FGR [2] which uses the full resolution Kinect data (640×480) for registration.*

components. Solutions include recent deep learning machinery [3], the celebrated iterative closest point (ICP) [4], and half a century old orthogonal Procrustes method [5] that is universally used till date. In this work, we study the problem of aligning highly sparse 3D scans without RGB information. In this setting, we lack boundary information, and without RGB components, it is hard to obtain any form of correspondences based on corners or line segments for registration.

We show the basic idea of our technique in Fig. 1. We fit lines to the data and represent the point-cloud using a large number of line segments. The original data hardly contains any correspondence constraints. On the contrary, in the re-parameterized line-based representation, there exists a rich set of line intersection constraints. Our strategy is to exploit these constraints to solve the registration problem. We consider 100x down-sampled Kinect data without RGB components. We show that the proposed algorithm can outperform competing Kinect registration algorithms that use full resolution (see Tab. 1).

There is an interesting connection between our approach and the relative pose estimation for generalized cameras. The problem of relative pose estimation involves finding rotation and translation between two cameras such that the projection rays associated with corresponding 2D points intersect with each other. In the case of perspective cameras, we can find the motion up to a scale, and the

associated projection rays are central [6]. In the case of generalized cameras [7], the projection rays are unconstrained, and we are looking at the alignment of two sets of line segments such that the corresponding line segments intersect with each other. In our problem, we compute line segments from the point cloud representing the 3D scene. In both settings, we try to satisfy line intersection constraints (which essentially amounts to placing four 3D points on a plane), where the four 3D points are the endpoints of the intersecting line segments. While minimal solvers typically produce robust solutions in general, the 6-point minimal solver for generalized relative pose [8] has degeneracy problems (i.e., the constraints are sometimes insufficient to produce a unique solution, and thereby lead to infinite number of poses). The algorithm is supposed to produce unique pose for truly unconstrained set of projection rays (two or more projection rays from the same camera can not be parallel or coplanar). It is hard to expect that the line segments re-parameterizing the point-cloud will satisfy them. Thereby, we solve the registration using an alternating projection [1] method (AP) with seven pairs of line intersection constraints (one additional constraint is used since six intersection constraints can have up to 64 different solutions [8]).

In addition to line intersection constraints, we also utilize constraints that corner points, occurring in the boundary regions, are incident with edge lines in the scene. The highlights of the paper are the following:

1. We propose an AP [1] algorithm for the problem of scan alignment by showing that it is equivalent to the generalized relative pose estimation, i.e., pose estimation by satisfying intersection constraints among six or more pairs of corresponding line segments.
2. We show a family of registration algorithms that, additionally, exploit corner points in sparse point clouds and utilize collinearity (corner points lying on corner lines) constraints.
3. We outperform other registration algorithms such as fast global registration (FGR) and Super4PCS, despite using 100X down-sampled point-cloud and not relying on the color information.
4. In 6 out of 11 KITTI LiDAR sequences, we outperform LOAM [9], a competing LiDAR registration algorithm that minimizes point-to-line and point-to-plane distances in a Levenberg Marquardt framework for registration.
5. On down-sampled KITTI data, we outperform LOAM in all 11 sequences.

2 Related Work

3D scan alignment is a classical problem in the computer vision and robotics communities, and there is a rich body of literature on this topic.

Iterative techniques: Iterative Closest Point (ICP) and its variants [4, 10–14] are the most applied scan alignment algorithm, and it works well for dense point-clouds with good initialization. ICP [4], the gold standard for dense point-to-point registration, is an alternating minimization algorithm, alternating between finding closest correspondences and computing the rigid transformation.

Alternating minimization and projection algorithms [1] have been used for human pose and other geometrical problems [15–19].

Techniques for finding globally optimal solutions combine local or probabilistic methods with graph optimization [20] and branch-and-bound [21, 22]. A closely related work is LOAM [9, 23], which is a top-ranking LiDAR alignment algorithm. We differ with LOAM as follows: 1) LOAM uses a distortion correction step for addressing the motion of the sensor at low frame-rate. This effect is similar to the rolling shutter effect in cameras. 2) LOAM uses the Levenberg Marquardt method for non-minimal registration, while we develop AP algorithm for near-minimal configurations and utilize the RANSAC framework. 3) LOAM additionally utilize IMU, and we rely only on sparse 3D points.

Several global methods have been proposed in the literature, such as [24, 25]. One of the problems with these methods is the high computation requirement for doing the branch and optimization. To overcome this, one line of research attempts to decompose the task of finding the relative transformation into first finding the rotation and then obtaining the translation given the optimal rotation [26, 27]. Further, [28] proposes Rotation Invariant Features (RIF) to ease the task of decoupling rotation solution from the translation. However, none of the methods have been tested on LiDAR data.

Minimal Solvers: Other popular scan alignment methods in the absence of good initialization include minimal 3-point solvers in a RANSAC framework. The most common approaches to remove outliers from the data are based on the use of RANSAC [29] plus some 3D point correspondences solver, such as [5, 30]. In addition to points, several registration algorithms have utilized other features on beam-based environment modeling [31], 3D planes [32–37], 3D line segments [33, 38], implicit surface representation [39], and edges [40]. A detailed survey on 3D SLAM methods can be found in [41, 42].

Our approach for the registration of 3D scans is tightly connected with minimal solvers for relative pose estimation. In particular, we have relative pose estimation algorithms for calibrated perspective cameras [6, 43], with known relative rotation angle [44], with known directions [45, 46], with unknown focal lengths [47, 48], solutions invariant to translation [49], and generalized relative pose [8, 50]. Recently, a minimal hybrid solver considers relative and absolute poses [51]. In the case of absolute poses, we have perspective [52–55], and multi-perspective systems [56, 57]. For 3D scans, we have the Procrustes solver for a pairwise alignement [5], and solutions to the mini-loop alignments in [30]. It was recently shown that the generalized relative pose estimation [8, 58–61] can be used in privacy-preserving pose estimation from 3D pointclouds [62].

Deep scan alignment: Deep neural networks (DNNs) for solving 3D registration problems have increased significantly in the last few years. Several methods have been developed to extract local 3D geometric structures, such as [63–68]. There have been a few recent algorithms for LiDAR registration [69, 70] using deep neural networks, but they are mostly applicable to dense point clouds.

A technique for registering 3D point clouds given a set of point correspondences is proposed in [3]. Floor plan reconstruction using deep networks has been shown in [36]. The PointNet [71] is used in [72], together with Lukas-Kanade algorithm in a single network. The proposed method works in an iterative fashion, similar to ICP. In [68, 73], a differential SVD layer is coupled with DNNs for generating matching. DNNs also prove to be helpful for solving other kind of problems such as multi-scan transformation averaging [74]. [75] utilizes deep neural networks to compute the weights for different pairwise relative pose estimates and [69] use two networks, for pose and scene estimation.

3 Problem Statement

Given two sparse 3D point sets $\{p_1^1, \ldots, p_{n_1}^1\}$ and $\{p_1^2, \ldots, p_{n_2}^2\}$, our goal is to compute $T = (R, \mathbf{t})$, such that $\{p_1^1, \ldots, p_{n_1}^1\}$ and the transformed point set $\{Rp_1^2 + \mathbf{t}, \ldots, Rp_{n_2}^2 + \mathbf{t}\}$ would model or represent the same 3D scene in the first coordinate frame.

We model the scene using a set of 3D line segments on planar surfaces and the detection of a few informative 3D corner points, and 3D edge lines. Concretely, the first point set $\{p_1^1, \ldots, p_{n_1}^1\}$ is re-parameterized as $\{l_1^1, \ldots, l_{m_l}^1, k_1^1, \ldots, k_{c_1}^1\}$, where l denotes *lines* and k are *corners* (obtained by line fitting and corner estimation respectively, see Sec. 5.1). The second point set $\{p_1^2, \ldots, p_{n_2}^2\}$ is re-parameterized as $\{g_1^2, \ldots, g_{m_2}^2, q_1^2, \ldots, q_{c_2}^2\}$, where g denotes *lines* and q are *edges* (obtained by line fitting and edge line estimation respectively, see Sec. 5.1). In the original point set representation, we can not identify any point correspondences due to sparsity. The re-parameterization allows us to obtain a rich set of line intersections (e.g., lines l_i^1 and g_i^2 intersect) and incidence relations (e.g., an edge q_j^2 passes through a corner point k_j^1).

Our implementation considers line intersections, and corner-edge, edge-corner incidences. For clarity of illustration, we will only look at lines and corners from the first frame, and lines and edges from the second.

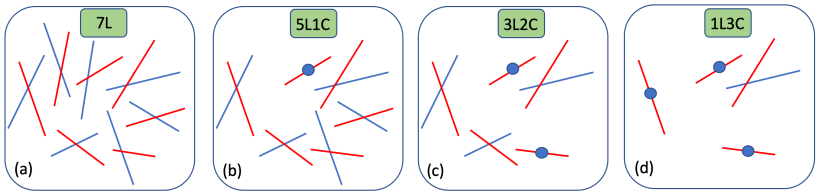


Fig. 2: *The four sets of constraints addressed in this paper are denoted by 7L, 5L1C, 3L2C, and 1L3C. For example, 3L2C in (c) denotes a total of five constraints (three line-to-line intersections, and two corner-to-line incidences). For simplicity, we consider the corners only from the first frame. The primitives from the first and the second frame are shown in blue and red, respectively.*

In Fig. 2 we show the different sets of constraints used in the paper. The line-to-line intersection constraint enforces that the four end-points of two intersecting line segments are coplanar, i.e., lie on the same plane. The corner-to-line incidence constraint enforces that a corner point lies on a line. It is always a useful exercise to check the degrees of freedom (DOF) of the unknowns and the number of available constraints. Our goal is to compute 6 DOF pose $(R, \mathbf{t})$ using line-to-line intersections and point-to-line incidences. Each constraint of line-to-line intersection and point-to-line incidences takes away 1 DOF (See [8]) and 2 DOFs (See [6, 43]), respectively. The algebraic methods that employ exact minimal solvers (where the DOFs from the constraints and the unknowns are equal) typically produce multiple solutions for poses corresponding to the multiple roots of the higher degree polynomial systems. Consequently, it needs a second step with additional correspondence constraints to pick the correct pose from the multiple solutions. In contrast to the algebraic solvers, we use the AP algorithm on near-minimal cases (i.e. each of the 4 sets of constraints shown in Fig. 2 has one DOF extra over the six unknowns), and the over-constrained setting leads to a single pose.

4 Alternating Projection (AP)

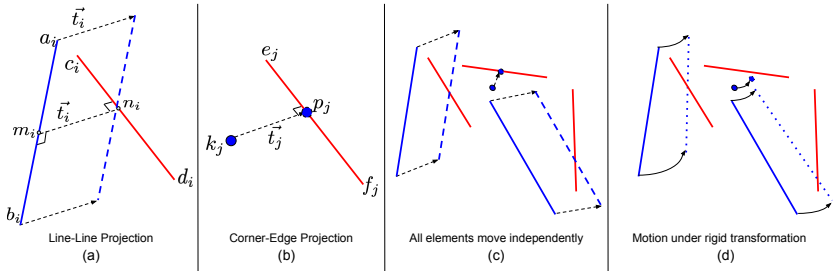


Fig. 3: *One* AP *iteration on a set of line-line and corer-edge pairs has two steps: 1) To satisfy their* **intersection** *constraints, it translates elements (lines, corners) from the first frame towards their counterparts (lines, edges) in the second frame, independent of the other pairs, as illustrated in (a), (b), and (c). 2) To preserve the* ***rigidity*** *for the first frame elements as a whole, it computes a single rigid transformation that takes all elements in this frame as close as possible to their position after the independent translation, as shown in (d).*

We briefly explain the general approach used in AP [1]. We compute a point (a higher dimensional vector denoting the unknowns) in the intersection of sets (denote the various constraint sets) by a sequence of projections onto the sets. Let us denote our unknown parameter vector as $x \in \mathbb{R}^n$, and the two sets as

C and D in $\mathbb{R}^n$, respectively. Let P_C and P_D be projection operators. Given a point $x \in \mathbb{R}^n$, the projection operator $P_C(x)$ produces a point $x' \in C$ such that $|x - x'|$ is minimum. The key elements of the AP are the constraints and the projection operations.

In our registration setting, given a set of candidate line intersections and corner-edge incidences, the goal of AP is to estimate a rigid transformation which satisfies the given intersection/incidence constraints (henceforth referred to simply as intersection constraints). Thus the solver needs to compute the unknowns $(R, \mathbf{t})$ to satisfy two constraint sets: **intersection**, and **rigidity**. The projection operators for intersection and rigidity constraints are shown in Fig. 3.

Algorithm 1 Alternating Projection

Require: $S^1 = \{l_1^1, .., l_M^1, k_1^1, .., k_N^1\}$, $S^2 = \{g_1^2, .., g_M^2, q_1^2, .., q_N^2\}$
Ensure: T
repeat
  $\overrightarrow{t_i}$ = LineLineProjection(l_i^1, g_i^2) for i $\in 1, ..., M$ ▷ as shown in Fig. 3(a)
  $\overrightarrow{t_j}$ = CornerEdgeProjection(k_j^1, q_j^2) for j $\in 1, ..., N$ ▷ as shown in Fig. 3(b)
  $l_i'^1 = l_i^1 + \overrightarrow{t_i}$ for i $\in 1, ..., M$ ▷ as shown in Fig. 3(c)
  $k_j'^1 = k_j^1 + \overrightarrow{t_j}$ for j $\in 1, ..., N$
  $S'^1 = \{l_1'^1, .., l_M'^1, k_1'^1, .., k_N'^1\}$
  $T =$`RigidAlign`(S^1, S'^1)
  Update $l_i^1 = T \cdot l_i^1$ for i $\in$ 1,...,M ▷ as shown in Fig. 3(d)
  Update $k_j^1 = T \cdot k_j^1$ for j $\in$ 1,...,N
  $\delta = \max_{i,j}(dist(l_i^1, g_i^2), dist(k_j^1, q_j^2))$
until $\delta \leq \epsilon$ or max iterations reached ▷ ϵ is a distance threshold
return T

Formally, let us denote the lines from the first frame with $l_i^1 = (a_i, b_i)$ (where a_i and b_i are the end-points), and those from the second frame by $g_i^2 = (c_i, d_i)$, for $i \in \{1, \ldots, M\}$, i.e. each (l_i^1, g_i^2) is a candidate pair of lines, where M is the number of line intersections used by that particular solver. For simplicity, let's consider only corners from the first frame k_j^1 and edges $q_j^2 = (e_j, f_j)$ from the second, for $j \in \{1, \ldots N\}$, such that each (k_j^1, q_i^2) is a candidate corner-edge pair, where N is the number of corner-edge incidence constraints used by the solver. In this paper, we solve the cases $(M, N) \in \{(7, 0), (5, 1), (3, 2), (1, 3)\}$, corresponding to our solvers 7L, 5L1C, 3L2C, and 1L3C respectively. Please note that edges are also line segments, but they are extracted in a different manner compared to regular line segments (see Sec. 5).

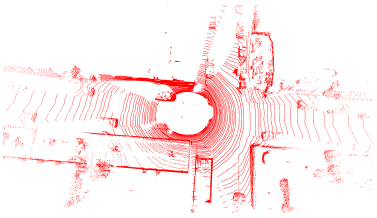
We illustrate our AP solver in Fig. 3 with a simple scenario that considers only two intersection constraints (one line-to-line and one corner-to-edge). In 3(a), the closest points on the line segments are denoted by m_i and n_i respectively. The line segment from the first frame (blue) is moved by the smallest distance so that the two line segments intersect. In 3(b), p_j denotes the point on the edge-line q_j closest to the corner point k_j, i.e. p_j is the projection of point k_j

on the line q_j. After applying this projection operation to frame 1 (blue) in (c), we get their projected position. For the rigid alignment between frame 1 (lines, corners) and their projected positions, we estimate translation by aligning their centroids and then estimate rotation using Orthogonal Procrustes, as shown in 3(d). We repeat these two steps (intersections followed by rigid transform estimation) until the update is lower than a threshold. The complete algorithm is illustrated in Algorithm 1. In our implementation, we move both elements of each pair towards each other, rather than moving only elements from the first frame towards those from the second.

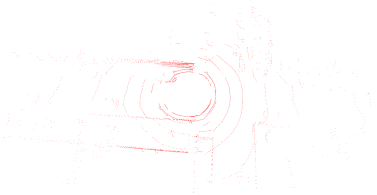
5 Implementation Details

5.1 Pre-processing

Organized point clouds: Our algorithms require line fitting outputs, which we find is easiest done on organized point clouds. In the case of Kinect data, the input is already organized in the form of a depth image, and no further action is necessary. In the case of the KITTI data, we use the sensor calibration parameters to organize the raw input points into a grid-like structure by azimuth and elevation. In the experiments involving downsampling of this data, we select points at appropriate indices from these organized point clouds. For instance, Fig. 4 shows a point-cloud down-sampled by a factor of 6 along both horizontal and vertical directions, to retain roughly $1/36^{\text{th}}$ the points.



(a) original scan



(b) 1/36 downsampling

Fig. 4: *Single frames from KITTI at (a) original resolution and (b) 1/36 down-sampled.*

Line Fitting: Given a set of 3D points in a regular grid (with some points missing due to sensor noise), we consider horizontal and vertical scan-lines in this organized point cloud and use RANSAC to fit lines to every scan-line. We call lines that come from horizontal scan-lines "H-lines", and the ones coming from vertical scan lines are called "V-lines".

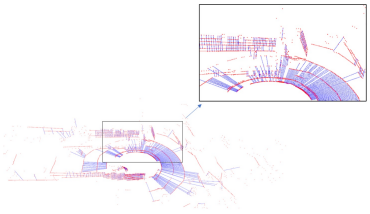


Fig. 5: *Line fitting on one KITTI frame. Red lines come from horizontal scan-lines(H-lines) and blue from vertical(V-lines).*

For instance, in Fig. 5, H-lines are represented by the color red, and V-lines by blue. H and V do not refer to the orientations of the lines in 3D, only the scan-line they come from; the blue lines on the ground plane in this figure are actually V-lines.

Corner points and edge lines: We use a similar formulation to [9] for estimating corner points. Let $\{x_i | i \in W\}$ denote the coordinates of the (ordered) points in a single scan-line in an organized point cloud in the local coordinate frame, where W is the total number of points in the scan-line. We define a local smoothness term for each point i based on K neighbors on either side of i as

$$c_i = \frac{1}{2K\|x_i\|} \| \sum_{j=-K, j\neq 0}^{K} (x_i - x_j) \|, \tag{1}$$

where a higher c value indicates lower smoothness. We divide each scan-line into 4 zones, and select two points with the highest c as corner points. We fit lines to corner points from successive scanes in the same frame to get the edge lines.

5.2 Full Meta-Algorithm with RANSAC

At each step, we estimate the frame-to-frame transformations between successive frames. We designate the local frame of the first scan to be the world coordinate frame, and compose (multiply) these transformations to obtain transformation matrices for all frames with respect to the first frame.

The proposed solvers (7L, 5L1C, 3L2C, 1L3C) produce a transformation matrix given a small number of inputs. We apply these solvers in a RANSAC framework to obtain the best possible transformation matrices. At each RANSAC iteration, we pick a solver at random (all 4 solvers are selected with the same probability, i.e., 0.25), and select candidates for that solver as appropriate.

Candidates for RANSAC: We identify candidates for (a) line intersection constraints based on the distance between line segments in the first and second scans, and (b) corner point-edge line incidence constraints based on the distances between corner points from one scan and edge lines from the other. We select from among these candidates at random in every RANSAC iteration. This simple approach assumes that the relative transformation is close to identity, but we observe that this works well in practice.

Inlier counting: From the set of candidates pairs of lines, we count the number of pairs that have a distance less than another threshold (2 cm on KITTI dataset and 5mm on TUM here) after applying the computed transformation.

In addition, we find that we get better results by running the full algorithm described above thrice, each time using the best transformation from the previous step as the initial guess for candidate estimation. With improved initialization we obtain better candidates for line intersection constraints. Both ICP and our algorithm iteratively update the relative transformation between two frames, but there are some important differences: (a) our algorithm needs only 3 iterations, while ICP typically needs many more, (b) within each iteration, our algorithm uses RANSAC, the AP solver works with near-minimal sets of constraints, while ICP uses correspondence constraints involving all primitives.

6 Experiments

We evaluated the proposed algorithms on two datasets: TUM [42] and KITTI [76].
Relative Pose Error (RPE): As proposed in [42], we compute the error in the relative pose between successive frames, and report the translation error in meters and rotation error in degrees respectively.

Translation and Rotation Error along the trajectory: This metric is used in [76] and on their online leaderboard. For all sub-sequences of length 100m, 200m, ..., 800m, we compute the translation and rotation errors per unit length of the trajectory. Translation error is reported as a percentage value, and rotation error in degrees per meter.

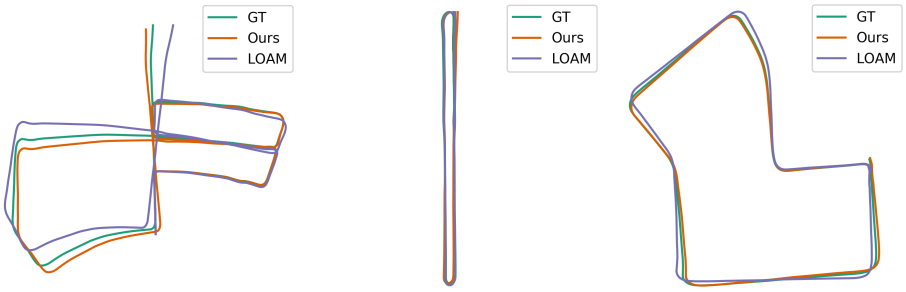
Kinect Data: TUM Dataset [42] consists of sequences of Kinect RGBD data captured in an indoor environment. The sensor resolution is 640x480, at 30 fps. The sequences come with a ground truth trajectory of the sensor, obtained from a high-accuracy motion-capture system. We test our algorithm on 7 sequences from the TUM dataset, down-sampled by a factor of 10 in both dimensions (i.e., $1/100^{\text{th}}$ the points). The error values for all three sequences are presented in Tab. 1 (on page 10). We use two baseline methods: Super4PCS [77] + ICP and fast global registration (FGR) [2] methods. Super4PCS works with point clouds without any point correspondences, and FGR uses point correspondences from the depth maps of the Kinect data. Both these methods are tested on full resolution Kinect data, whereas our approach only takes the sparsified input (down-sampled by a factor of 100). As shown in Tab. 1, we outperform the baselines in 6 out of the 7 sequences, despite using down-sampled data.

Sequence	**Proposed**		**Super4PCS+ICP**		**FGR**	
	tra. [mm]	rot. [°]	tra. [mm]	rot. [°]	tra. [mm]	rot. [°]
`fr1/xyz`	**3.86**	**0.46**	12.99	0.49	9.51	0.70
`fr1/360`	17.67	0.82	22.89	0.79	**14.76**	**0.76**
`fr3/sitting_xyz`	**6.09**	0.41	9.47	**0.39**	9.09	0.46
`fr1/room`	**7.25**	**0.52**	15.29	0.73	11.82	0.70
`fr1/plant`	**6.47**	**0.61**	14.23	0.74	11.86	0.68
`fr3/cabinet`	**7.25**	**0.55**	16.56	0.64	16.63	0.84
`fr3/structure_nn`	**4.49**	**0.35**	8.28	0.53	9.77	0.67

Table 1: *Results on 7 sequences of the TUM RGBD dataset. Our method uses only the depth-maps, down-sampled by a factor of 100. Super4PCS + ICP and FGR use full resolution depth maps. Mean rotation and translation error between successive frames is reported.*

LiDAR Data: KITTI dataset [76] consists of LiDAR point-clouds collected from the top of a moving vehicle. The LiDAR sensor captures roughly 10 fps (frames per second), with about 100k points per frame.

We compare our results against the LOAM Algorithm [9], which is currently highly ranked on the online leaderboard. The computed trajectories on these sequences are shown in Fig. 6, the relative pose error between consecutive frames,



(a) Sequence 5 (b) Sequence 6 (c) Sequence 7

Fig. 6: *Computed trajectories on KITTI Sequences at original resolution. The trajectory from the proposed method is closer to the ground truth than LOAM.*

and the error along the trajectory is reported in Tab. 2 (on page 12). Selected full point-clouds after registration are presented in Fig. 7 (on page 14).

The results of LOAM here differ from those on the leaderboard because we run LOAM by ourselves, without using IMU data, for a fair comparison. As the official version is unavailable, we use the open-source version of LOAM [4].

Comparison of Our Algorithms: We test all our algorithms separately on sequence 03 of the KITTI dataset. The combined approach produces the best translation and 5L1C produces the best rotation (Tab. 4). All of these approaches have their own strengths and weaknesses. For example, line intersection constraints handle planar regions better, and the methods utilizing corner points are supposed to handle scenes containing vegetation and other non-planar regions.

Algorithm	tra. err. [%]	rot. err. [deg/m]
7L	2.628	0.021
5L1C	2.452	**0.017**
3L2C	2.589	0.025
1L3C	2.794	0.021
Combined	**2.291**	0.023

Table 4: *Comparison of our algorithms*

Under extreme sparsity: We test our approach on LiDAR data down-sampled by a factor of 6 in both dimensions, $1/36^{\text{th}}$ of all points (see Fig. 4, page 8) and compare our results with LOAM on the KITTI dataset. We show a trajectory in Fig. 8, and the errors are reported in Tab. 3 (on page 13). While both our algorithm and LOAM deteriorate as we increase the sparsity, our algorithm significantly outperforms LOAM.

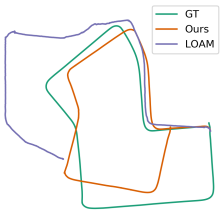


Fig. 8: *1/36 Down-sampled KITTI sequence 7*

Performance and Speed: Our code is implemented in C++. The average computation time on KITTI dataset at full resolution for various operations

[4] `https://github.com/laboshinl/loam_velodyne`

Seq.	Proposed				LOAM			
	Mean errors		KITTI metrics		Mean errors		KITTI metrics	
	tra. [m]	rot. [°]	tra. [%]	rot. [°/m]	tra. [m]	rot. [°]	tra. [%]	rot. [°/m]
00	**0.024**	**0.105**	**1.612**	**0.007**	0.061	0.457	2.865	0.015
01	2.125	2.432	91.139	0.327	**0.651**	**0.245**	**25.711**	**0.022**
02	0.198	**0.325**	17.361	0.057	**0.165**	0.396	**8.567**	**0.033**
03	**0.035**	**0.106**	**2.291**	0.023	0.103	0.361	9.861	**0.021**
04	0.698	0.247	57.950	0.091	**0.138**	**0.213**	**1.927**	**0.009**
05	**0.018**	**0.096**	**1.407**	**0.009**	0.048	0.312	1.721	0.011
06	**0.023**	**0.079**	**0.818**	**0.006**	0.063	0.211	1.884	0.010
07	**0.018**	**0.084**	**0.999**	**0.006**	0.042	0.321	2.080	0.014
08	**0.044**	**0.105**	3.064	**0.011**	0.064	0.348	**2.705**	0.013
09	0.249	0.366	18.654	0.054	**0.109**	**0.328**	**5.196**	**0.031**
10	**0.074**	**0.289**	10.558	0.047	0.093	0.432	**9.727**	**0.038**

Table 2: *Mean translation and rotation errors between successive frames, and translation, rotation errors over the trajectory for KITTI sequences. Proposed method has a lower error than LOAM in many sequences.*

(running on 1 core of Intel i7-8700K) is as follows: 22ms for the AP solver; 2ms for the inlier counting; and 3s for the line fitting. The time taken by inlier counting varies with the number of detected lines; these are typically observed values. The total runtime of the entire algorithm depends on the number of RANSAC iterations and can be brought down to 30s by paralleling on 10 threads.

We use a threshold (2cm on KITTI, 5mm on TUM) as well as max number of iterations (30K iterations) for terminating the AP. In the future, we will explore certain extrapolated projection techniques for further speedup [78].

6.1 Sensitivity analysis

We conducted some experiments to study the robustness of our algorithm to large motion and sparse settings.

Convergence under large motion: We generate a pair of frames by taking the first frame of KITTI's sequence 7 and apply large translation and rotation to it. Then we use the same fitting strategy to get pairs of lines and edge/corner candidates from the two frames. We repeat the process 100 times and applied the AP solver for registration. We count a success when the registration has rotation error smaller than 0.5°and translation error within 0.1 meter. The success rate is the number of success out out of 100 trials. As shown in Tab. 5, the success rate is monotonic and RANSAC only needs some

tra. [m]	succ. rate	rot. [deg]	succ. rate
0.5	94%	0.5	97%
2	86%	2	91%
8	79%	8	83%
32	65%	32	78%
128	60%	128	49%

Table 5: *Robustness to large translations and rotation by augmenting them in noisy LiDAR data.*

Seq.	**Proposed**				**LOAM**			
	Mean errors		KITTI metrics		Mean errors		KITTI metrics	
	tra. [m]	rot. [°]	tra. [%]	rot. [°/m]	tra. [m]	rot. [°]	tra. [%]	rot. [°/m]
00	**0.202**	**0.191**	**14.937**	**0.033**	0.446	1.343	38.939	0.176
01	**2.196**	**0.936**	**94.428**	**0.134**	2.205	1.602	95.274	0.212
02	**0.544**	**0.396**	**36.003**	**0.065**	0.833	1.488	62.246	0.194
03	**0.277**	**0.300**	**33.545**	**0.064**	0.583	1.573	89.949	0.491
04	**1.329**	**0.400**	**92.768**	**0.074**	1.386	1.762	97.335	0.352
05	**0.172**	**0.180**	**14.139**	**0.038**	0.527	2.041	54.738	0.280
06	**0.615**	**0.466**	**37.981**	**0.094**	0.956	2.813	60.321	0.370
07	**0.123**	**0.123**	**14.522**	**0.038**	0.334	1.491	36.841	0.209
08	**0.290**	**0.346**	**26.796**	**0.101**	0.479	2.081	58.518	0.272
09	**0.551**	**0.541**	**44.765**	**0.134**	0.781	1.921	66.405	0.255
10	**0.188**	**0.311**	**19.770**	**0.071**	0.384	1.444	36.346	0.189

Table 3: **Under extreme sparsity:** *Mean translation and rotation errors between successive frames, and translation, rotation errors over the trajectory for KITTI sequences at low resolution (*$1/36^{th}$ *the points). Proposed method outperforms LOAM in all sequences.*

good hypotheses. For comparison, the maximum translation, rotation between two frames in KITTI is 1.5 m and 5°respectively.

Robustness of intersection constraints under sparsity: Under sparsity, line intersection constraints continue to exist while point correspondences diminish significantly. We take two consecutive frames of KITTI's sequence 7 and align them using the ground truth. For each line/point in the first frame, we find the closest line/point in the second frame. We fixed a threshold for the line/point distance for inlier counting and vary the down-sampling factor (sparsity). As observed in Fig. 9, the percentage of inliers using line intersections (line2line in Fig. 9) is not affected by sparsity. On the other hand, the percentage of inliers using point correspondences (point2point in Fig. 9) decreases considerably. This means that point-based methods like ICP are more sensitive to the inlier threshold.

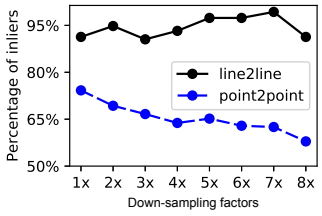


Fig. 9: *Inlier ratio of point correspondence vs. intersected lines as sparsity increases.*

7 Discussion

The proposed algorithm applies to sensors on a moving platform and we make smoothness assumption for obtaining line intersection constraints, although our algorithm is robust to outliers due to the use of near-minimal solvers in a RANSAC framework. While we outperform LOAM in 6 out of 11 sequences,

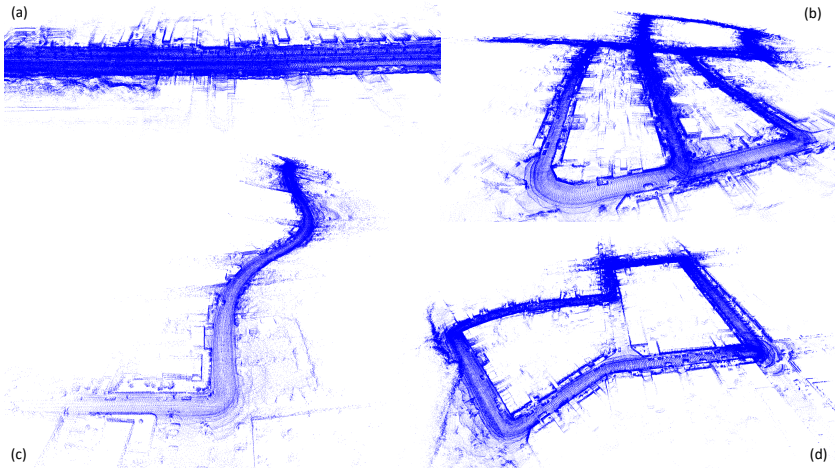


Fig. 7: *Visualization of the fully registered point-clouds from KITTI sequences (a) 06, (b) 05, (c) 03, and (d) 07.*

our method can further be improved by following LOAM and correcting for distortions obtained from moving platforms [9]. We also show that we outperform LOAM in all the KITTI sequences under extreme sparsity. Note that the proposed method is not customized to a specific sensor. After handling the challenges in LiDAR, we ported to Kinect datasets with almost zero-development cost and outperformed all baselines in 6 out of 7 sequences using 1/100th of the data, without texture.

At the heart of our technique, we hinge on two elements to achieve the superior performance. First, we exploit interior regional information (by fitting line segments on the point-cloud and generating rich line intersection constraints) in addition to the boundary information (through the extraction of corners and edges) typically used by classical methods. Every line intersection adds a planarity constraint implicitly on the four end points. Compared to plane detection, intersected lines is easier to find in sparse point cloud. Plane detection in sparse settings with sensor motion distortion can be brittle, and finding enough planes in near-degenerate situations will be challenging (e.g., many road scenes with buildings on both sides or highways may not provide enough planes to lock the pose from sliding). Second, despite the simplicity, AP should not be treated as a trivial endeavour. In addition to several vision related problems [15–19], variants of AP have also been used to solve computationally hard problems such protein folding, sphere packing, and Sudoku [79, 80].

In our work, we observed that AP can solve near-minimal problems that are known to be hard for algebraic solvers. From the formulations used in this paper, it is not difficult to see that the proposed approach can be easily extended to other geometric vision problems as well. We show a video demonstration of our algorithm in the Supplementary Materials.

References

1. Bauschke, H.H., Borwein, J.M.: On projection algorithms for solving convex feasibility problems. SIAM Review **38** (1996) 367–426
2. Zhou, Q.Y., Park, J., Koltun, V.: Fast global registration. In: European Conf. Computer Vision (ECCV). (2016) 766–782
3. Pais, G.D., Miraldo, P., Ramalingam, S., Govindu, V.M., Nascimento, J.C., Chellappa, R.: 3DRegNet: A deep neural network for 3d point registration. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2020)
4. Besl, P.J., McKay, N.: A method for registration of 3-D shapes. IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI) **14** (1992) 239–256
5. Schönemann, P.H.: A generalized solution of the orthogonal procrustes problem. Psychometrika **31** (1966) 1–10
6. Nister, D.: An efficient solution to the five-point relative pose problem. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2003)
7. Grossberg, M.D., Nayar, S.K.: A general imaging model and a method for finding its parameters. In: IEEE Int'l Conf. Computer Vision (ICCV). (2001)
8. Stewenius, H., Oskarsson, M., Astrom, K., Nister, D.: Solutions to minimal generalized relative pose problems. In: Workshop on Omnidirectional Vision (OMNIVIS). (2005)
9. Zhang, J., Singh, S.: LOAM: Lidar odometry and mapping in real-time. In: Robotics: Science and Systems (RSS). (2014)
10. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-d point sets. IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI) **9** (1987) 698–700
11. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society of America A **4** (1987) 629–642
12. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI) **13** (1991) 376–380
13. Penney, G.P., Edwards, P.J., King, A.P., Blackall, J.M., Batchelor, P.G., Hawkes, D.J.: A stochastic iterative closest point algorithm (stochastICP). In: Medical Image Computing and Computer-Assisted Intervention (MICCAI). (2001) 762–769
14. Colas, F., Pomerleau, F., Siegwart, R.: A review of point cloud registration algorithms for mobile robotics. Foundations and Trends in Robotics **4** (2015) 1–104
15. Zhou, X., Leonardos, S., Hu, X., Daniilidis, K.: 3d shape reconstruction from 2d landmarks: A convex formulation. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2015)
16. Zhou, X., Zhu, M., Daniilidis, K.: Multi-image matching via fast alternating minimization. In: IEEE Int'l Conf. Computer Vision (ICCV). (2015)
17. Yan, J., Wang, J., Zha, H., Yang, X., Chu, S.M.: Multi-view point registration via alternating optimization. In: AAAI Conference on Artificial Intelligence. (2015)
18. Schops, T., Sattler, T., Pollefeys, M.: Bad slam: Bundle adjusted direct rgb-d slam. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2019)
19. Campos, J., Cardoso, J., Miraldo, P.: POSEAMM: A unified framework for solving pose problems using an alternating minimization method. In: IEEE Int'l Conf. Robotics and Automation (ICRA). (2019)
20. Theiler, P.W., Wegner, J.D., Schindler, K.: Globally consistent registration of terrestrial laser scans via graph optimization. ISPRS Journal of Photogrammetry and Remote Sensing **109** (2015) 126–138

21. Campbell, D., Petersson, L.: GOGMA: Globally-optimal gaussian mixture alignment. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2016) 5685–5694
22. Li, H., Hartley, R.: The 3D-3D registration problem revisited. In: IEEE Int'l Conf. Computer Vision (ICCV). (2017) 1–8
23. Zhang, J., Singh, S.: Low-drift and real-time lidar odometry and mapping. Autonomous Robots **41** (2017) 401–416
24. Yang, J., Li, H., Jia, Y.: Go-ICP: Solving 3d registration efficiently and globally optimally. In: IEEE Int'l Conf. Computer Vision (ICCV). (2013) 1457–1464
25. Yang, J., Li, H., Campbell, D., Jia, Y.: Go-ICP: A globally optimal solution to 3D ICP point-set registration. IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI) **38** (2016) 2241–2254
26. Makadia, A., Patterson, A., Daniilidis, K.: Fully automatic registration of 3d point clouds. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). Volume 1. (2006) 1297–1304
27. Straub, J., Campbell, T., How, J.P., III, J.W.F.: Efficient global point cloud alignment using bayesian nonparametric mixtures. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2017) 2403–2412
28. Liu, Y., Wang, C., Song, Z., Wang, M.: Efficient global point cloud registration by matching rotation invariant features through translation search. In: European Conf. Computer Vision (ECCV). (2018) 460–474
29. Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24** (1981) 381–395
30. Miraldo, P., Saha, S., Ramalingam, S.: Minimal solvers for mini-loop closures in 3d multi-scan alignment. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2019)
31. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-D mapping with an RGB-D camera. IEEE Trans. Robotics (T-RO) **30** (2014) 177–187
32. Raposo, C., Loureno, M., Barreto, J.P., Antunes, M.: Plane-based odometry using an rgb-d camera. In: British Machine Vision Conference (BMVC). (2013)
33. Zhou, L., Li, Z., Kaess, M.: Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In: IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS). (2018)
34. Ma, L., Kerl, C., Stuckler, J., Cremers, D.: CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM. In: IEEE Int'l Conf. Robotics and Automation (ICRA). (2016) 1285–1291
35. Bhattacharya, U., Veerawal, S., Govindu, V.M.: Fast multiview 3D scan registration using planar structures. In: Int'l Conf. 3D Vision (3DV). (2017) 548–556
36. Liu, C., Wu, J., Furukawa, Y.: FloorNet: A unified framework for floorplan reconstruction from 3d scans. In: European Conf. Computer Vision (ECCV). (2018)
37. Grant, W.S., Voorhies, R.C., Itti, L.: Efficient velodyne SLAM with point and plane features. Autonomous Robots (2018)
38. Lu, Y., Song, D.: Robust RGB-D odometry using point and line features. In: IEEE Int'l Conf. Computer Vision (ICCV). (2015) 3934–3942
39. Deschaud, J.E.: IMLS-SLAM: Scan-to-model matching based on 3D data. In: IEEE Int'l Conf. Robotics and Automation (ICRA). (2018) 2480–2485
40. Choi, C., Trevor, A.J.B., Christensen, H.I.: Rgb-d edge detection and edge-based registration. In: IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS). (2013) 1568–1575

41. Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An evaluation of the rgb-d slam system. In: IEEE Int'l Conf. Robotics and Automation (ICRA). (2012) 1691–1696
42. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS). (2012) 573–580
43. Li, H., Hartley, R.: Five-point motion estimation made easy. In: Int'l Conf. Pattern Recognition (ICPR). Volume 1. (2006) 630–633
44. Li, B., Heng, L., Lee, G.H., Pollefeys, M.: A 4-point algorithm for relative pose estimation of a calibrated camera with a known relative rotation angle. In: IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS). (2013) 1595–1601
45. Fraundorfer, F., Tanskanen, P., Pollefeys, M.: A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. In: European Conf. Computer Vision (ECCV). (2010) 269–282
46. Saurer, O., Vasseur, P., Demonceaux, C., Fraundorfer, F.: A homography formulation to the 3pt plus a common direction relative pose problem. In: Asian Conf. Computer Vision (ACCV). (2015) 288–301
47. Stewenius, H., Nister, D., Kahl, F., Schaffalitzky, F.: A minimal solution for relative pose with unknown focal length. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). Volume 2. (2005) 789–794
48. Li, H.: A simple solution to the six-point two-view focal-length problem. In: European Conf. Computer Vision (ECCV). (2006) 200–213
49. Kneip, L., Siegwart, R., Pollefeys, M.: Finding the exact rotation between two images independently of the translation. In: European Conf. Computer Vision (ECCV). (2012) 696–709
50. Ventura, J., Arth, C., Lepetit, V.: An efficient minimal solution for multi-camera motion. In: IEEE Int'l Conf. Computer Vision (ICCV). (2015) 747–755
51. Camposeco, F., Cohen, A., Pollefeys, M., Sattler, T.: Hybrid camera pose estimation. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2018) 136–144
52. Kneip, L., Scaramuzza, D., Siegwart, R.: A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2011) 2969–2976
53. Ke, T., Roumeliotis, S.I.: An efcient algebraic solution to the perspective-three-point problem. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2017) 4618–4626
54. Wang, P., Xu, G., Wang, Z., Cheng, Y.: An efficient solution to the perspective-three-point pose problem. Computer Vision and Image Understanding (CVIU) **166** (2018) 81–87
55. Persson, M., Nordberg, K.: Lambda twist: An accurate fast robust perspective three point (P3P) solver. In: European Conf. Computer Vision (ECCV). (2018) 334–349
56. Ventura, J., Arth, C., Reitmayr, G., Schmalstieg, D.: A minimal solution to the generalized pose-and-scale problem. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2014) 422–429
57. Camposeco, F., Sattler, T., Pollefeys, M.: Minimal solvers for generalized pose and scale estimation from two rays and one point. In: European Conf. Computer Vision (ECCV). (2016) 202–218
58. Pless, R.: Using many cameras as one. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). Volume 2. (2003) 587

59. Sturm, P.: Multi-view geometry for general camera models. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2005)
60. Li, H., Hartley, R., hak Kim, J.: A linear approach to motion estimation using generalized camera models. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2008)
61. Kneip, L., Li, H.: Efficient computation of relative pose for multi-camera systems. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. (2014)
62. Pittaluga, F., Koppal, S.J., Kang, S.B., Sinha, S.N.: Revealing scenes by inverting structure from motion reconstructions. In: CVPR. (2019)
63. Elbaz, G., Avraham, T., Fischer, A.: 3D point cloud registration for localization using a deep neural network auto-encoder. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2017) 2472–2481
64. Khoury, M., Zhou, Q.Y., Koltun, V.: Learning compact geometric features. In: IEEE Int'l Conf. Computer Vision (ICCV). (2017) 153–161
65. Zhou, L., Zhu, S., Luo, Z., Shen, T., Zhang, R., Zhen, M., Fang, T., Quan, L.: Learning and matching multi-view descriptors for registration of point clouds. In: European Conf. Computer Vision (ECCV). (2018) 505–522
66. Deng, H., Birdal, T., Ilic, S.: Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In: European Conf. Computer Vision (ECCV). (2018) 602–618
67. Deng, H., Birdal, T., Ilic, S.: 3d local features for direct pairwise registration. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2019)
68. Lu, W., Wan, G., Zhou, Y., Fu, X., Yuan, P., Song, S.: Deepvcp: An end-to-end deep neural network for point cloud registration. In: IEEE Int'l Conf. Computer Vision (ICCV). (2019)
69. Ding, L., Feng, C.: DeepMapping: Unsupervised map estimation from multiple point clouds. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2019)
70. Elbaz, G., Avraham, T., Fischer, A.: 3d point cloud registration for localization using a deep neural network auto-encoder. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2017) 2472 – 2481
71. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2017) 652–660
72. Aoki, Y., Goforth, H., Srivatsan, R.A., Lucey, S.: Pointnetlk: Robust & efficient point cloud registration using pointnet. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2019) 7163–7172
73. Wang, Y., Solomon, J.M.: Deep closest point: Learning representations for point cloud registration. In: IEEE Int'l Conf. Computer Vision (ICCV). (2019)
74. Chatterjee, A., Govindu, V.M.: Robust relative rotation averaging. IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI) **40** (2018) 958–972
75. Huang, X., Liang, Z., Zhou, X., Xie, Y., Guibas, L., Huang, Q.: Learning transformation synchronization. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2019)
76. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (2012)
77. Mellado, N., Mitra, N., Aiger, D.: SUPER 4PCS: Fast global pointcloud registration via smart indexing. Computer Graphics Forum (Proc. EUROGRAPHICS) **33** (2014) 205–215

78. Censor, Y., Chen, W., Combettes, P.L., Davidi, R., Herman, G.T.: On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints. Computational Optimization and Applications (2012)
79. Gravel, S., Elser, V.: Divide and concur: A general approach to constraint satisfaction. Physical Review E **78** (2008)
80. Elser, V., Rankenburg, I., Thibault, P.: Searching with iterated maps. In: Proc. National Academy of Sciences of the United States of America (PNAS). Volume 104. (2007) 418–23