Robust Rank Deficient SLAM

Samer B. Nashed[1,2], Jong Jin Park[2], Roger Webster[2], and Joseph W. Durham[3]

***Abstract*— Autonomous mobile robots need maps for effective, safe navigation, and SLAM in general is still an unsolved problem. Nonetheless, certain combinations of environmental characteristics and sensors admit tractable solutions. In particular, detection and tracking of linear features such as line segments (2D) or planar facets (3D) has been proven robust in many man-made environments. However, these types of features produce rank-deficient constraints, which create challenges for graph-based SLAM optimizers. We present techniques for using rank-deficient features and constraints more robustly by analyzing the approximate null-space of the constraints for each node in the factor graph representing the trajectory. We also extend auxiliary methods for correspondence calculations and map update routines, the combination of which yields state-of-the-art performance for a rank-deficient SLAM system. We present results from quantitative experiments comparing memory use, compute load, accuracy, and robustness for several ablation tests on real and simulated data.**

I. Introduction

Simultaneous localization and mapping (SLAM) is a prerequisite for deploying autonomous mobile robots, and the performance of SLAM systems depends on the environment. SLAM systems that use depth sensors are the preeminent choice for indoor scenarios due to the accuracy of modern sensors and the desire of many practitioners to build dense geometric models of the environment. In many cases, indoor environments present linear features such as line segments (2D) or planar facets (3D), which can be detected robustly [20]. Such features have many benefits, including ease of detection and quality of outlier rejection. However, they also present challenges, including correspondence calculation, optimization robustness, and long-term map quality. This paper addresses these challenges individually and presents a state-of-the-art SLAM system for rank deficient constraints, which we call (RD-SLAM).

RD-SLAM is designed for environments that contain linear features and addresses two weaknesses inherent in dense iterative closest point (ICP) [3], [4] and correlative scan matching (CSM) [17]. First, ICP-based methods are not robust to outliers, while CSM cannot compute exact maximum likelihood transformations due to discretization. Second, neither algorithm is memory efficient online, typically requiring storage of raw point clouds or an occupancy grid. As robotic applications grow in scale and operate on ever lighter hardware, these representations become intractable.

[1] College of Information and Computer Sciences, University of Massachusetts, Amherst, MA 01003, USA. Email: snashed@umass.edu
[2] Amazon Lab126, 1100 Enterprise Way, Sunnyvale, CA 94089, USA. Email: {jongpark,rogerweb}@amazon.com
[3] Amazon Robotics, 300 Riverpark Dr, North Reading, MA 01864, USA. Email: josepdur@amazon.com

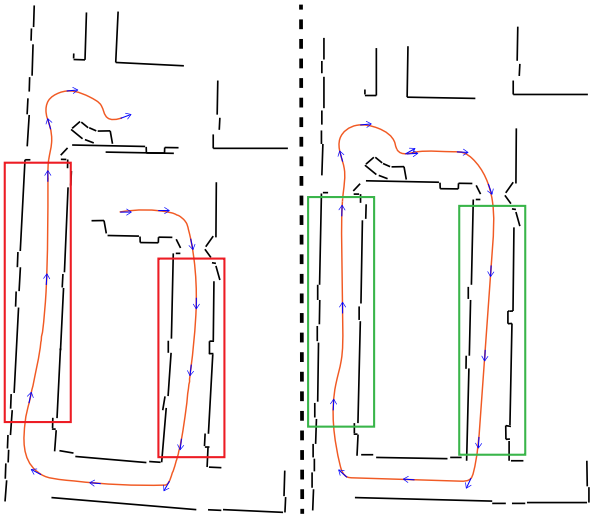

Fig. 1: RD-SLAM on 2D laser data with (right) and without (left) prevention of optimization along degenerate axes, which are unconstrained directions detected as the null space of the set of visual constraints. Long straight hallways produce degenerate axes for some poses. Robot trajectory is in orange, and features in black.

To address these problems, RD-SLAM extracts line segments or planar facets and computes relative transformations by calculating correspondences on these larger features.

Given the lack of a distance metric between line segments or planar facets, we offer a set of algorithms and similarity functions for robust correspondence calculation. While linear features are easier to detect and track, using them in nonlinear least-squares optimization problems can cause instability since correspondences may not fully constrain the robot's motion. We propose an algorithm for adding regularization terms to the optimization problem based on the approximate null space of sets of rank deficient constraints, and its effect is shown in Fig 1. These terms also reduce the sensitivity of the optimizer to the uncertainty models of different sensors, and in contrast to hard constraints may allow the optimizer to escape local minima. We also extend an existing method used to construct and maintain highly compressed, maximum likelihood geometric maps to allow these maps to be updated online as the robot's trajectory estimate changes.

We evaluate RD-SLAM on real and simulated data, and present experiments examining the system's robustness to sensor noise, memory efficiency, compute load, and accuracy. We also perform ablation tests including other popular methods for each contribution. Moreover, we show that the combined effect of improvements to correspondence calculations and co-linear factor design can lead to reductions in compute and memory load as well as decrease the frequency of large localization errors.

II. Related Work

Metric SLAM is a well-studied topic of research and, broadly speaking, metric SLAM algorithms build either geometric reconstructions or maps of keypoints and landmarks. In this paper we restrict our attention to reconstructions of regular or man-made environments. Several existing SLAM algorithms have been designed for such environments. Many make strong assumptions, such as planar features appearing uniform [25], [8], completely rectilinear environments [7], or access to custom feature detectors [12]. Here, we make only the assumption that the environment contains line segments or planar facets that may be extracted from depth sensor data. In most deployment contexts this assumption is easily met.

Virtually all metric SLAM algorithms compute the affine transformation of the robot between successive frames, and many algorithms for computing these transformations are derivatives of the iterative closest point (ICP) method in that they compute correspondences and then minimize the distances between correspondences through optimization. ICP variants designed for specific environments or to address certain shortcomings of vanilla ICP include different methods of computing correspondences [5] or changing the minimization routine, such as by adding noise [18]. A survey of ICP algorithms is presented in [19]. Generalizations have also been established [13], [21], giving rise to algorithms using different physical primitives.

Both 2D and 3D RD-SLAM belong to a family of ICP-based methods which compute correspondences between geometric primitives such as line segments [1], [2], polylines [9], or planes [11]. One weakness of such approaches is their reliance on extraction of geometric objects and robust definitions of similarity or distance between objects in order to compute accurate correspondences. Fortunately, line segment extraction in 2D [16], and plane extraction in 3D [20], are mature areas of research. Various definitions for distance between line segments or between planar facets, which we extend here, have been explored in the context of line segment matching and are summarized nicely in [24].

Although many approaches use potentially rank deficient features, only a small number have investigated using co-linear constraints. Some approaches incorporate them into already constrained factor graphs [15], while most detect degeneracy online [10], [26], [6], [23]. RD-SLAM takes the latter approach, but differs in that it does not explicitly project inertial measurements along degenerate dimensions or do any hard switching between sensing modalities. Instead, we detect degeneracies and add regularization terms to the existing optimization problem. An additional challenge when constructing long term maps is how to combine primitives which describe the same physical object. RD-SLAM follows the approach of Long Term Vector Mapping [14], which uses shape covariance matrix decomposition. A similar trick was presented in [22] under the term recursive least-squares. This paper extends these methods to work online in the event that primitives may need to be transformed when the underlying pose estimates change during optimization.

III. Rank-deficient SLAM

RD-SLAM does not describe a single trick or insight. Rather, we describe a set of challenges and the corresponding types of approaches which result in functioning systems. These challenges include choosing the right level of abstraction for feature detection and calculating feature correspondences $x3{:}A$, using rank deficient constraints robustly $x3{:}B$, and maintaining a consistent, high-accuracy, low-memory map in the context of online trajectory optimization $x3{:}C$.

A. Feature Correspondences

Both dense reconstructions from point clouds and keypoint-based systems typically compute correspondences. These computations are costly, can require non-trivial data structures, and often lack robustness. False correspondences are common and in the absence of advanced non-linear optimization techniques may cause catastrophic failure. Geometric primitives such as line segments and planar facets inherently mitigate some of these challenges in several ways.

First, we have fast, accurate, robust algorithms for line segment and planar facet detection [16], [20]. Second, because of the relatively low number of features detected per frame due to their inherent size, even naive correspondence calculations can be done quickly. Third, large feature size creates natural robustness to false correspondences, since the relative motion of the robot between frames is typically small compared to the distance between distinct features. However, such features present a unique challenge in defining similarity functions or distance measures. Since an established distance metric over $SE(2)$ or $SE(3)$ does not exist, similarity functions are typically constructed heuristically, often by summing or otherwise combining proper distance metrics defined over subsets of $SE(2)$ or $SE(3)$. Below, we present pseudometrics for robustly computing correspondences between line segments and planar facets.

1) Line Segments in 2D: We derive a measure for line-segment similarity (LSS) under the following assumptions. First, corresponding line segments extracted from successive scans should have similar location and orientation. And second, corresponding line segments do not need to be co-located; they may be only co-linear. Not only does co-linearity provide sufficient information as long as there are at least 2 non-parallel segments, but it is also more robust than co-location in some scenarios where this condition is not met, such as when travelling down a straight corridor, or when only part of the feature can be detected by the robot due to occlusion or range and field of view limitations.

Many formulae for LSS have been proposed [24], but none meet all of the criteria which follow from the assumptions above. Thus, we present a definition of LSS which is sensitive to the relative positions of segments anisotropically. Given line segments a and b, we define $\mathrm{LSS}(a;b)$ as

$$\mathrm{LSS}(a;b) = \left((d_{\quad} = {}_{\quad})^2 + (d_{?} = {}_{?})^2 + (d_{k} = {}_{k})^2 \right)^{\frac{1}{2}} \qquad (1)$$

where d are different metrics over subspaces of $SE(2)$, and are scale factors based on sensor characteristics.

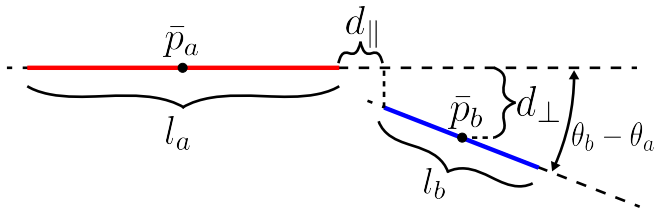

Fig. 2: Variables for computing LSS between line segments a (red) and b (blue).

Let line segments a and b have endpoints $p_1^i; p_2^i$, lengths l_i, centers of mass p_i, and orientations $_i$ for $i = a; b$, as in Figure 2. We define the d terms of LSS as follows.

$$d = j\sin(_b \quad _a)j \quad \text{and} \quad d_? = j(p_b \quad p_a) \quad \hat{n}_a j; \tag{2}$$

where $\hat{n}_a$ is the unit vector normal to line segment a. Defining segment a such that $l_a \quad l_b$ allows $d_?$ to be symmetric. Distance in the parallel direction is non-zero if the projection of both p_1^b and p_2^b onto the line defined by segment a fall outside the boundaries of segment a. That is,

$$d_k = \min(d_k^1; d_k^2); \tag{3}$$

where

$$d_k^j = \begin{cases} t_j \quad l_a & t_j > l_a \\ 0 & 0 \quad t_j \quad l_a \\ jt_j j & t_j < 0: \end{cases} \tag{4}$$

Here,

$$t_j = (p_j^b \quad p_1^a) \frac{(p_2^a \quad p_1^a)}{jj(p_2^a \quad p_1^a)jj} \quad \text{where} \quad j = 1; 2: \tag{5}$$

2) Planar Facets in 3D: Planar facet similarity (PFS) can be computed robustly in a similar manner. Given two planar facets a and b, PFS is composed of similar terms.

$$\text{PFS}(a; b) = \quad (d = \quad)^2 + (d_? = \quad _?)^2 + (d_k = \quad _k)^2 \quad ^{\frac{1}{2}}; \tag{6}$$

where d and $d_?$ become the angle between normal vectors and the point to plane distance, respectively. One key difference is the definition of d_k. Determining if a pair of planar facets overlap is expensive when considering the hull of each facet explicitly. Therefore, we represent each facet by an ellipse which we derive from the eigenvectors and eigenvalues of a shape covariance matrix constructed from the subset of the pointcloud corresponding to the planar facet, shown in Figure 3. Given ellipses a and b where $(a; b) <$, defined by centers p_a, p_b, eigenvectors e_1^a, e_2^a and e_1^b, e_2^b, and eigenvalues $_1^a$, $_2^a$ and $_1^b$, $_2^b$, we project b onto a, producing sets of eigenvectors which are co-planar. Let these new eigenvectors and eigenvalues be $e_1^{b'}$, $e_2^{b'}$ and $_1^{b'}$, $_2^{b'}$, respectively. We define d_k as

$$d_k = \max(0; (jjp_a \quad p_b jj \quad jj \quad _a jj \quad jj \quad _b jj)): \tag{7}$$

The equations presented below for assume ellipse has been transformed so p is at the origin with $e_1 \quad \hat{y} = 0$.

$$= ht \quad _1 \quad _2 \cos( ); t \quad _1 \quad _2 \sin( )i; \tag{8}$$

where

$$t = \quad \sqrt{(\quad _1)^2 \cos^2( ) + (\quad _2)^2 \sin^2( )} \quad ^1; \tag{9}$$

and

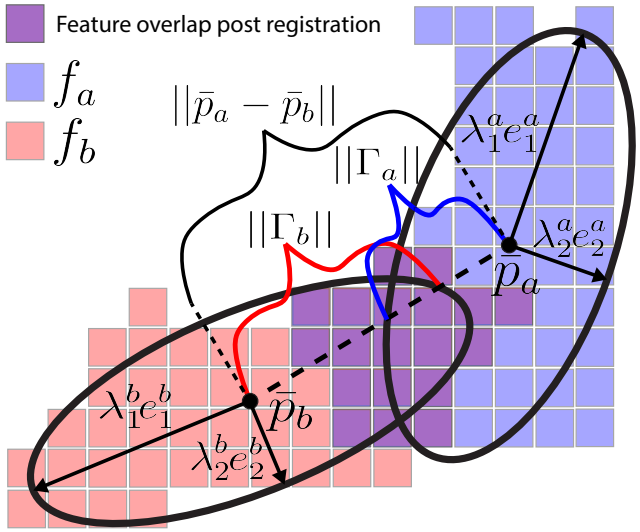

Fig. 3: Ellipses for features a (blue) and b (red) detect overlap using the sum of projections ( ) and the distance between feature centers. Purple patches show actual feature overlap.

$$= \arctan(p_y; p_x): \tag{10}$$

In addition to the LSS and PFS pseudometrics, we also find that discarding features based roughly on size adds additional robustness. We discard line segments shorter than 20cm and discard planar facets with area less than 0.16m^2.

B. Dealing with Rank Deficient Constraints

Given a set of correspondences C, where $c_{a;b} \; 2 \; C$ relates feature a to feature b visible at times t_i and t_j, respectively, we can generally write the optimization problem for visual features using co-linear or co-planar constraints as

$$X = \underset{X}{\text{argmin}} \sum_{c_{a;b} 2 C} d(a; A_{ij} b) + d_?(a; A_{ij} b): \tag{11}$$

where X is the MLE trajectory and A_{ij} transforms features observed at pose x_j to the frame of pose x_i. The obvious limitation of such a constraint is the potential lack of information along one or more axes. Such situations are common when sensor sampling density, field of view, or range decrease, limiting the number and informatic diversity of observed features. Computation constraints that force smaller optimization windows have a similar effect, since they lower the probability of making informative data associations.

Consider pose x_t and the set of correspondences C_t, where $8c_{a;b} \; 2 \; C_t$, either a or b was observed at time t. Let F be the set of all features f such that $c_{f;} \; 2 \; C_t$. A scaled version of the second moment matrix is then

$$M = \sum_{f 2 F} \hat{n}_f \hat{n}_f^T; \tag{12}$$

where $\hat{n}_f$ is the unit normal of feature f. Analytically, degeneracy can be detected by performing Gaussian elimination on M. However, noise in depth data results in feature normal estimates that almost always produce matrices that are technically full rank. This can cause the optimizer to find global minima with respect to $d_?$ that are not accurate, due to low signal to noise ratios in the null directions.

Algorithm 1 NULL SPACE APPROXIMATION

Input: Set of features F, threshold
Output: Basis of null(M_t), N_t
N ;, M [0]
for f 2 F **do**
  M M + $\hat{n}_f \hat{n}_f^T$
V V^{-1} EIGENDECOMPOSITION(M)
for $_i$ 2 diag( ) where $_i$ ϵ $_{max}$ **do**
  $_{max}$ = $_i$
  if > **then**
    N_t N_t [$V_{;i}$
return N_t

To solve this, we approximate the null-space of M and apply constraints along all null directions. In contrast to other approaches, which use 'hard' constraints to prevent the optimizer from moving along null directions at all, we enforce these constraints as regularization terms, or 'soft' constraints, essentially constraining the optimizer to find a solution by moving only along well-conditioned directions to within some tolerance.

The method for approximating null(M) is shown in Algorithm 1. We analyze M's condition numbers, , which, because M is normal, are computed from its eigenvalues. Large condition numbers indicate degenerate dimensions, and in our experiments we used = 10:0. In practice, is easy to tune as most degenerate axes have condition numbers orders of magnitude higher than well-conditioned axes. For each pose within the optimization window, Algorithm 1 produces a set N_t that represents a basis of the null space of constraints on pose x_t. Regularization terms of the form

$$J(X) = \sum_{t=1}^{t} \sum_{\wedge 2 N_t} ((x_t \quad x_{t-1}) \quad (u_t \quad x_{t-1})) \wedge \tag{13}$$

are then added to the cost functions for each pose. Here, $\wedge$ are basis vectors describing the null space of visual constraints, u are the inertial measurements, and x are the pose variables. Combining equations 11 and 13 together, along with a cost function for the inertial measurements, we get an overall objective similar to

$$\begin{aligned} X \quad = \text{argmin} \quad & \sum_{t=1}^{t} jj(x_t \quad x_{t-1}) \quad (u_t \quad x_{t-1})jj \\ & + \sum_{c_{a;b} 2 C} d\ (a; A_{ij} b) + d_?(a; A_{ij} b) \\ & + \sum_{t=1}^{t} \sum_{\wedge 2 N_t} ((x_t \quad x_{t-1}) \quad (u_t \quad x_{t-1})) \wedge : \end{aligned} \tag{14}$$

C. Map Updates

To store long-term representations of the environment we adapt Long-term Vector Mapping (LTVM) [14] to work online. Online LTVM checks newly detected features registered in global frame against an existing map of features, which is empty when the robot is first deployed. Each frame, statistical tests are performed which estimate the likelihood that a given feature corresponds to a physical entity already represented in the map. If the new feature represents an unobserved object it is added to the map. If it represents an observation of an already mapped object, the map feature is updated as a weighted sum of the new feature and the map feature, where the weight is the number of raw observations supporting each feature. We find that different statistical tests work well for different features. For line segments we use chi-squared tests as in [14], and for planar facets we use conservative thresholds of projections based on the elliptical representation presented in x3:A:2.

Merging can also be performed between two features already in the map if their boundaries grow together. This process is expensive in the sense that it scales as $O(n^2)$, where n is the number of features in the map, since every mapped feature must be checked to see if it can merge with any other feature. However, in practice n is small since features are merged incrementally. Even for building-scale maps, n is typically in the hundreds or thousands. Moreover, space partitioning data structures such as kd-trees can eliminate most comparisons, providing further speedup.

One of the major strengths of LTVM is the maintenance of maximum likelihood feature location estimates along with a high compression ratio. This is possible via storing the shape covariance matrix representation of the supporting observations of each line segment or planar facet in a decoupled manner. However, features are extracted in robot frame, but the map updates are done in global frame. Thus, the decoupled representation must be rotated to global frame. We represent features using decoupled shape covariance matrices constructed from N depth observations p,

$$S = \sum_{i=1}^{N} p_i p_i^T \quad N p p^T = S_o \quad N S; \tag{15}$$

where S_o represents the orientation of the feature, and S is the outer product of the feature's centroid, p. Given an affine transformation defined by rotation R and translation T, we can compute the transformed matrices S^0 and S_o^0 as

$$S^0 = (Rp + T)(Rp + T)^T \tag{16}$$

and

$$S_o^0 = N(R\frac{1}{N} S_o R^T \quad RSR^T + S^0): \tag{17}$$

IV. RESULTS

We are mostly concerned with compute and memory efficiency and with accuracy and robustness given low quality sensing containing significant noise and visual artifacts. We hypothesize that under these constraints, algorithms from the RD-SLAM family, and specifically the improvements presented in this paper, offer advantageous tradeoffs compared to dense ICP methods as well as methods that deal with rank-deficiency by enforcing hard constraints on factors.

To test this hypothesis, we conduct several experiments using both simulated 2D lidar and 3D point cloud data and 2D and 3D data collected at the University of Massachusetts Amherst. We used a Hokuyo UST-10LX and an Asus Xtion PRO for lidar and point cloud data, respectively. Robots outfitted with these sensors were tele-operated around buildings

at the university which include a number of straight hallways with limited features as well as some open areas with widths that exceed the sensor range in some places. Importantly, these data sets represent canonical human environments with large, easily observable features that contain only partial information. Moreover, many points in the trajectory cannot be fully constrained based on visual features. The robot often receives information that constrains only one positional axis, and this condition can persist for periods longer than the sliding window used for optimization which is typically 1 to 2 seconds. For each sensor, data was collected from 5 deployments, each taking a different path through the same environment. All timing experiments were done on a 3.70GHz quad-core processor.

A. Compute and Memory Efficiency

To test compute efficiency, we compare the time required for both correspondence calculations and pose optimization for RD-SLAM against dense ICP. We include the time required to extract features into the timing results for RD-SLAM, and we use a dense ICP implementation with a kd-tree for efficiently pruning non-correspondences. We find that RD-SLAM takes on average 7ms to produce 3D correspondences, while the ICP implementation requires 22ms on average. Time saved during optimization is also substantial. With an optimization window of 20 poses, RD-SLAM uses an average of 61ms to converge while point-to-point correspondences take on average 177ms.

To test memory efficiency, we compare the space required to store several different possible map representations using a simulated environment. Storing raw point clouds in 3D requires about 10MB per second. This grows unbounded over the deployment and is clearly infeasible. Creating a 3D occupancy grid with a resolution of 2cm requires nearly 100MB to explore our roughly 10m by 10m environment. Storing the map in an oct-tree reduces memory, but is still more expensive than planar representations. Storing all planar facets extracted during the deployment also grows without bound, but in our simulation it requires only roughly 1MB for every 10,000 poses. Lastly, merging planar facets incrementally allows the robot to store the entire map in about 10KB. Compute and memory savings are more pronounced for 3D data, but are still significant for 2D data.

B. Accuracy

Figures 5, and 6 show the effects of different optimization routines on accuracy using simulated 2D data. We use 2D simulations instead of 3D simulations because it is easier to construct more complex and realistic noise models. The main hypothesis is that soft constraints allow the solver more flexibility, which is beneficial in some scenarios, and the histograms illustrate how soft and hard constraints perform when optimizing co-linear constraints. The key takeaway is that although hard constraints are slightly more likely to produce very low error, they are also more likely to produce higher error, and in this respect soft constraints seem to increase the probability that a given location estimate will have error less than some , for sufficiently large .

Figure 4 shows a qualitative comparison between no constraints, hard constraints, and soft constraints on a real laser data set. We believe the increase in accuracy compared to the naive method (no constraints) is due primarily to the elimination of catastrophic localization failures, where the optimizer finds minima far from the ground truth. This behavior may be a natural consequence of optimization along directions with no real information or where the signal to noise ratio is very small, corresponding to the rank-deficient axes. We believe the decreased upper bound on error relative to optimizers using hard constraints is due to an entirely different phenomenon. In this case, soft constraints may open paths to minima with lower absolute values that are not accessible when using hard constraints, in some sense increasing the basin of convergence for some minima. This may be most beneficial when dealing with exceptionally noisy data that does not contain a strong signal.

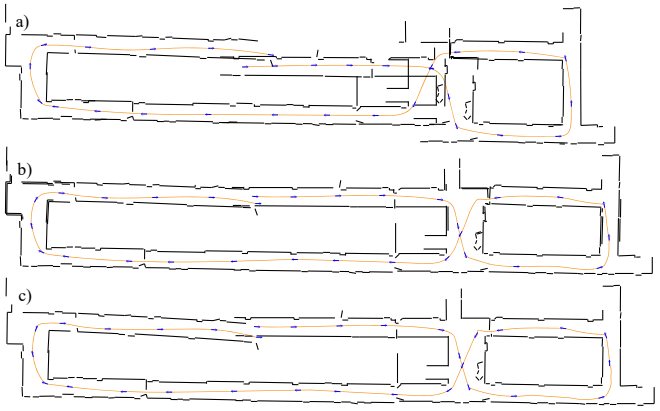

Fig. 4: Maps produced using optimization over co-linear visual constraints. In a), no additional terms are added. In b), optimization along degenerate axes is prohibited. In c), regularization terms are added to discourage, but not prevent changes along degenerate axes.

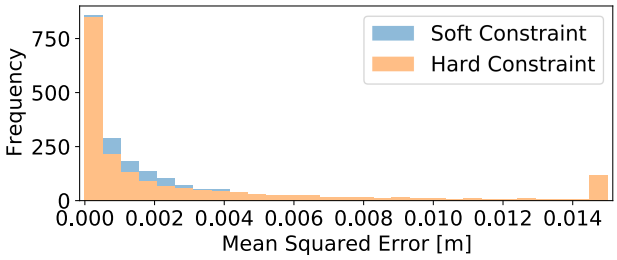

Fig. 5: Histogram of translation MSE w.r.t. constraint type.

C. Robustness

Figures 7 and 8 compare the accuracy of dense ICP and the proposed methods for computing correspondences between line segments under different levels of simulated noise. As expected, dense methods lack robustness to outliers in point-to-point correspondence calculations, and we see l_2 filtering increases robustness substantially. This is due to a small number of features during each deployment that erroneously pass each filter individually, but are not in reality reliable