2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

On Degeneracy of Optimization-based State Estimation Problems

Ji Zhang, Michael Kaess, and Sanjiv Singh

Abstract— **Positioning and mapping can be conducted accurately by state-of-the-art state estimation methods. However, reliability of these methods is largely based on avoiding degeneracy that can arise from cases such as scarcity of texture features for vision sensors and lack of geometrical structures for range sensors. Since the problems are inevitably solved in uncontrived environments where sensors cannot function with their highest quality, it is important for the estimation methods to be robust to degeneracy. This paper proposes an online method to mitigate for degeneracy in optimization-based problems, through analysis of geometric structure of the problem constraints. The method determines and separates degenerate directions in the state space, and only partially solves the problem in well-conditioned directions. We demonstrate utility of this method with data from a camera and lidar sensor pack to estimate 6-DOF ego-motion. Experimental results show that the system is able to improve estimation in environmentally degenerate cases, resulting in enhanced robustness for online positioning and mapping.**

I. Introduction

Recent developments on state estimation methods have shown promising results in positioning and mapping. It is now common to use vision and/or range sensing to recover sensor motion with low drift over long distances. However, the problem of degeneracy remains less studied and prevents navigation systems from reliable functioning, e.g. a camera in a feature-poor scene or a lidar in a planar environment, causing estimation failure. Common ways to deal with degeneracy are 1) switching to a different method, and 2) adding in artificial constraints such as from a constant velocity model during the course of state estimation. Neither approach is satisfactory as the first approach requires a spare method being available, and the second approach brings in unnecessary errors even when the problem itself is well-conditioned and solvable.

Our approach is motivated by the observation that even if a set of data used by an estimation method is locally degenerate, very often, some of the constraints provided can be used to solve in a subspace of the original problem. In other words, additional constraints or assumptions do not need to apply in well-conditioned directions, but degenerate directions only, to maximally reduce their negative effect. The above discussion leads to two extreme cases, 1) the problem is well-conditioned entirely and can be solved as is, and 2) the problem is degenerate completely and needs help in all DOF. Our method automatically separates degenerate directions from well-conditioned directions. When degeneracy is determined, a simple technique called solution remapping linearly combines the solution with a best guess. The problem is only solved in well-conditioned directions, and the best guess is used in degenerate directions.

J. Zhang, M. Kaess, and S. Singh are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213. Email: zhangji@cmu.edu, kaess@cmu.edu, and ssingh@cmu.edu.

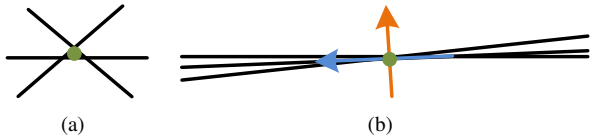


Fig. 1. Intuition of the proposed method in determining state estimation degeneracy. The black lines represent constraints in a state estimation problem. (a) illustrates a well-conditioned problem. The solution (green dot) is constrained in different directions. (b) gives an example of degeneracy. The constraints are mostly parallel such that the problem is degenerate along the blue arrow. Our method evaluates the constraints online to determine degeneracy. A simple technique called solution remapping automatically separates degenerate directions (blue arrow) from well-conditioned directions (orange arrow), and only solves the problem in the well-conditioned directions. This prevents faulty solutions caused by degeneracy.

The proposed method online evaluates degeneracy for optimization-based methods, through analysis of geometric structure of the problem constraints. In a linearized system, intuitively, a constraint is a (hyper-)plane in the state space. A set of well-conditioned constraints should distribute toward different directions, constraining the solution from different angles (an example is shown in Fig. 1(a)). On the other hand, the case that all planes are mostly parallel corresponds to degeneracy – the solution is poorly constrained in directions parallel to the planes (as shown in Fig. 1(b)). We mathematically define a *degeneracy factor* as stiffness of the solution w.r.t. disturbances to the constraints. By formulating and solving an optimization problem, we derive a closed form expression of the *degeneracy factor* containing nothing but eigenvalues and eigenvectors of the system – without reinventing the wheel, our findings enrich existing eigenvalues and eigenvectors with new geometrical meanings.

Our method functions as a plug-in step and can be adapted to common linear and nonlinear solvers with negligible add-in computational complexity. We evaluate the method with a custom-built state estimation system combing both vision and lidar sensors. Experimental results show that the system is able to improve estimation in environmentally degenerate cases and robustly conduct online positioning and mapping.

II. Related Work

The topic of this paper is most relevant to sensitivity and robustness analysis. Sensitivity has been extensively studied in the robotics community. Typically, a covariance matrix is used as a representation of accuracy. For state estimation problems, if a Kalman filter [1] is used, the filter itself maintains a covariance matrix of the state estimate.

Other work has been devoted to derive an upper bound of the covariance [2]. With a particle filter [1], distributions of the particles and the corresponding importance weights represent the uncertainty. A Jacobian matrix is commonly used to propagate errors from sensor noise to uncertainty of state estimates assuming local linearity. For instance, Eudes and Lhuillier [3] analyze accuracy of structure from motion by deriving a sequence of Jacobian matrices connecting errors in camera pixels and 3D reconstructed points, in a local bundle adjustment problem. Also, Censi uses the error model of ranger finders to derive a lower bound of the state estimation accuracy (known as achievable accuracy) in both pose tracking [4] and localization [5] problems. The bound is derived through usage of Fisher information matrices.

Robustness is mostly studied in the control domain. Robustness refers to the capability of a system to converge to a planned state or a sequence of states under disturbances [6]. A well-known class of methods is robust control [7]. The methods study convergence characteristics of a closed loop system with variations of system parameters, based on the Lyapunov theory [8]. The methods are widely used on ground [9] and aerial [10], [11] vehicles for controller design. Also, from linear control theory [12], one is able to determine controllable/observable subspaces. It is similar to our method which finds degenerate directions. Using controllability/observability gramian [13], one can analyze directions of the controllable/observable subspaces through its eigenvalues and eigenvectors. In comparison, our method is designed for analysis of degeneracy. After separation of degenerate directions from well-conditioned directions, the method linearly projects the solution and combines it with the best guess to prevent estimation failure.

The study of robustness is rather sparse in positioning and mapping problems. The robustness of state estimates is mostly handled heuristically. For example, with an RGB-D camera, Hu et al.'s [14] motion estimation method switches between two estimation algorithms to handle failure of an individual algorithm. Also, robustness can refer to capability of an optimization method in finding global optima instead of local optima. For instance, the Levenberg-Marquardt method is known to be more robust than the Gauss-Newton method [15]. In comparison, our problem is essentially different. The global optimum itself can be a faulty solution. Our method is devoted to determining degenerate directions and preventing the optimization from finding the global optimum in those directions – the solution is only updated in well-conditioned directions and a best guess is used in the degenerate directions.

The contributions of this paper are, 1) defining and deriving a closed form expression of a *degeneracy factor* for general optimization-based methods, and 2) introducing solution remapping to prevent estimation failure online with little computational cost. Since the degeneracy is analyzed with a linearized system, the paper is also relevant to perturbation theory for linear systems [16]. However, perturbation theory studies characteristics of the system with changes to existing constraints, while our proposed method inserts an additional constraint into the system as a disturbance and studies change of the solution w.r.t. the disturbance. The difference is that the disturbance can be introduced in an arbitrary direction but the perturbation is limited to the existing constraints.

III. Problem Statement

The goal of this study is to evaluate degeneracy of an optimization-based state estimation problem, by studying the structure of the constraints in the state space. Define $\boldsymbol{x}$ as a $n \times 1$ state vector, where n is the dimension of the state space. Our state estimation problem is to solve a function,

$$\arg\min_{\boldsymbol{x}} \boldsymbol{f}^2(\boldsymbol{x}). \tag{1}$$

In the case that (1) is a linear function of $\boldsymbol{x}$, we can directly solve for $\boldsymbol{x}$ with the singular value decomposition or QR decomposition method [17]. On the other hand, if (1) is a nonlinear function, methods such as Gauss-Newton, gradient descent, and Levenberg-Marquardt [15] can be used. Most nonlinear optimization methods locally linearize the problem by computation of the Jacobian matrix of $\boldsymbol{f}$ w.r.t. $\boldsymbol{x}$,

$$\mathbf{J} = \partial \boldsymbol{f}(\boldsymbol{x}) / \partial \boldsymbol{x}. \tag{2}$$

Given an initial guess, the methods iteratively adjust $\boldsymbol{x}$ by usage of $\mathbf{J}$ until convergence to find an optimum.

Regardless of linearity, a linear problem is always involved, either as the problem itself or as a step when solving a nonlinear problem. Hence we investigate the linear problem

$$\arg\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2. \tag{3}$$

Our method considers each row in (3) as a (hyper-)plane in the space of $\boldsymbol{x}$, and studies geometric distribution of the planes to determine degeneracy. We make two assumptions,

- We assume that matrix $\mathbf{A}$ is appropriately weighted taking into account sensor noise. In other words, (3) is a linearized form of (1) that retains the weights of the original problem.
- We assume that the problem is full rank, i.e. not under-constrained and the state estimate is dominated by the true sensor measurements in the case that the problem itself is not degenerate.

With assumptions made, our problem can be stated as,

Problem 1: Given a linearized system as (3), determine degeneracy and corresponding degenerate directions in the state space. In the case of degeneracy, prevent faulty solutions from occurring in the degenerate directions.

IV. Degeneracy Evaluation

A. Mathematical Derivation

This section describes the methodology to evaluate degeneracy of a linearized system. We start with mathematical definition of the *degeneracy factor*. As shown in Fig. 2, the black lines represent the constraints in a system described by (3), and the blue dot indicates the true solution, denoted

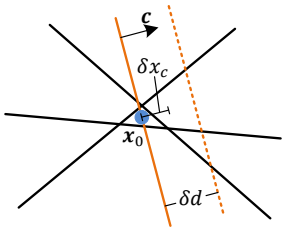


Fig. 2. Definition of *degeneracy factor*. We insert an additional constraint (orange line) as a disturbance and inspect movement of the solution $\boldsymbol{x}_0$.

as $\boldsymbol{x}_0$. To measure degeneracy of the system, we insert an additional constraint passing through $\boldsymbol{x}_0$ as the orange line,

$$\boldsymbol{c}^T(\boldsymbol{x}-\boldsymbol{x}_0)=0,\ ||\boldsymbol{c}||=1, \tag{4}$$

where $\boldsymbol{c}$ is a $n\times 1$ vector indicating the normal of the constraint (the black arrow in Fig. 2). Since the constraint intersects with $\boldsymbol{x}_0$, insertion of the constraint does not change $\boldsymbol{x}_0$. Then, we move the constraint toward its normal direction $\boldsymbol{c}$ for a certain distance δd. Correspondingly, we measure the shift of $\boldsymbol{x}_0$ in the same direction. Let δx_c be the amount of shift. For a given δd, the shift δx_c varies as a function of the direction of $\boldsymbol{c}$. Let δx_c^* be the maximum amount of shift,

$$\delta x_c^* = \max_{\boldsymbol{c}} \delta x_c. \tag{5}$$

We now define the *degeneracy factor* as follows,

Definition 1: The *degeneracy factor* $\mathcal{D}$ of a system is mathematically defined as $\mathcal{D}=\delta d/\delta x_c^*$.

By such a definition, we evaluate stiffness of the solution w.r.t. disturbances to the constraints. By maximizing δx_c, we find a direction in which the solution is the least stable. The corresponding stiffness in that direction is considered a measurement of degeneracy. In other words, the degeneracy is determined by the lower-bound of stiffness w.r.t. disturbances in all possible directions. Here, note that Definition 1 is not limited to linearized systems. However, a closed form expression of $\mathcal{D}$ is available if the system is linearized. Next, Lemma 1 and Lemma 2 help us derive $\mathcal{D}$.

Lemma 1: For the linearized system (3), the *degeneracy factor* $\mathcal{D}$ is a function of $\mathbf{A}$, but not $\boldsymbol{b}$.

Proof: Let us start with insertion of the additional constraint passing through $\boldsymbol{x}_0$. Eq. (3) can be viewed as solving $\mathbf{A}\boldsymbol{x}=\boldsymbol{b}$ with the l^2 norm. Stacking (3) with (4),

$$\begin{bmatrix}\mathbf{A}\\ \boldsymbol{c}^T\end{bmatrix}\boldsymbol{x}=\begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{c}^T\boldsymbol{x}_0\end{bmatrix}. \tag{6}$$

It is trivial to see that the solution of (6) is still $\boldsymbol{x}_0$. Now, we introduce a disturbance by shifting the constraint toward this normal direction by δd. This gives

$$\begin{bmatrix}\mathbf{A}\\ \boldsymbol{c}^T\end{bmatrix}\boldsymbol{x}=\begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{c}^T\boldsymbol{x}_0+\delta d\end{bmatrix}. \tag{7}$$

Let $\delta\boldsymbol{x}$ be the corresponding shift of $\boldsymbol{x}_0$. With the disturbance introduced, the solution of (7) becomes $\boldsymbol{x}_0+\delta\boldsymbol{x}$. Applying left pseudo-inverse to the left sides of (6) and (7), where

$$\begin{bmatrix}\mathbf{A}\\ \boldsymbol{c}^T\end{bmatrix}^{-1}_{\text{left}}=(\begin{bmatrix}\mathbf{A}^T & \boldsymbol{c}\end{bmatrix}\begin{bmatrix}\mathbf{A}\\ \boldsymbol{c}^T\end{bmatrix})^{-1}\begin{bmatrix}\mathbf{A}^T & \boldsymbol{c}\end{bmatrix}, \tag{8}$$

and subtracting (6) from (7), we can compute $\delta\boldsymbol{x}$,

$$\delta\boldsymbol{x}=(\mathbf{A}^T\mathbf{A}+\boldsymbol{c}\boldsymbol{c}^T)^{-1}\boldsymbol{c}\delta d. \tag{9}$$

Recall that $||\boldsymbol{c}||=1$, the shift in the direction of $\boldsymbol{c}$, δx_c, can be calculated as the dot product of $\boldsymbol{c}$ and $\delta\boldsymbol{x}$,

$$\delta x_c=\boldsymbol{c}^T\delta\boldsymbol{x}=\boldsymbol{c}^T(\mathbf{A}^T\mathbf{A}+\boldsymbol{c}\boldsymbol{c}^T)^{-1}\boldsymbol{c}\delta d. \tag{10}$$

Eq. (10) tells us that δx_c is a function of $\mathbf{A}$, $\boldsymbol{c}$, and δd, but not a function of $\boldsymbol{b}$. Hence, we complete the proof. ■

Lemma 1 indicates that the degeneracy is only determined by directions of the constraints, represented by $\mathbf{A}$, and irrelevant to positions of the constraints, $\boldsymbol{b}$. This confirms our intuition in Fig. 1 that parallelism of the constraints introduces degeneracy while different directions of the constraints maintain a well-conditioned system. Further, the following Lemma 2 will help us derive the expression of $\mathcal{D}$.

Lemma 2: For the linearized system (3), $\mathcal{D}=\lambda_{\min}+1$, where $\lambda_{\min}$ is the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$.

Proof: Since $||\boldsymbol{c}||=1$, the left and right pseudo-inverse of $\boldsymbol{c}$ are $\boldsymbol{c}^{-1}_{\text{left}}=(\boldsymbol{c}^T\boldsymbol{c})^{-1}\boldsymbol{c}^T=\boldsymbol{c}^T$ and $\boldsymbol{c}^{-T}_{\text{right}}=\boldsymbol{c}(\boldsymbol{c}^T\boldsymbol{c})^{-1}=\boldsymbol{c}$. Substituting these two equations into (10), we obtain

$$\begin{aligned}\delta x_c &= \boldsymbol{c}^{-1}_{\text{left}}(\mathbf{A}^T\mathbf{A}+\boldsymbol{c}\boldsymbol{c}^T)^{-1}\boldsymbol{c}^{-T}_{\text{right}}\delta d\\ &= (\boldsymbol{c}^T(\mathbf{A}^T\mathbf{A}+\boldsymbol{c}\boldsymbol{c}^T)\boldsymbol{c})^{-1}\delta d\\ &= (\boldsymbol{c}^T\mathbf{A}^T\mathbf{A}\boldsymbol{c}+1)^{-1}\delta d.\end{aligned} \tag{11}$$

In (11), the value of δd is given. To maximize δx_c, we equally minimize $\boldsymbol{c}^T\mathbf{A}^T\mathbf{A}\boldsymbol{c}$ in the following problem,

Problem 2: Compute $\boldsymbol{c}^*$ to minimize function

$$\boldsymbol{c}^*=\arg\min_{\boldsymbol{c}}\ \boldsymbol{c}^T\mathbf{A}^T\mathbf{A}\boldsymbol{c},\ s.t.\ ||\boldsymbol{c}||=1. \tag{12}$$

In (12), since $||\boldsymbol{c}||=1$, Problem 2 is equal to

$$\boldsymbol{c}^*=\arg\min_{\boldsymbol{c}}\ \frac{\boldsymbol{c}^T\mathbf{A}^T\mathbf{A}\boldsymbol{c}}{\boldsymbol{c}^T\boldsymbol{c}}. \tag{13}$$

The term to be minimized in (13) is a Rayleigh quotient [18]. Since $\mathbf{A}^T\mathbf{A}$ is a symmetric matrix, the minimum of the quotient is equal to the minimum eigenvalue of $\mathbf{A}^T\mathbf{A}$, namely $\lambda_{\min}$. This happens when $\boldsymbol{c}^*$ is the corresponding eigenvector of $\lambda_{\min}$. Substituting $\boldsymbol{c}^*$ into (11), we can derive

$$\delta x_c^*=\frac{\delta d}{\lambda_{\min}+1}. \tag{14}$$

Therefore, the *degeneracy factor* $\mathcal{D}=\delta d/\delta x_c^*=\lambda_{\min}+1$. ■

Lemma 2 indicates that the degeneracy is determined by $\lambda_{\min}$ of $\mathbf{A}^T\mathbf{A}$. The associated eigenvector, denoted as $\boldsymbol{v}_{\min}$, represents the first degenerate direction. Let λ_i and $\boldsymbol{v}_i$ be the i-th smallest eigenvalue and eigenvector of $\mathbf{A}^T\mathbf{A}$, $i=1,...,n$, where $\lambda_1=\lambda_{\min}$ and $\boldsymbol{v}_1=\boldsymbol{v}_{\min}$. Further expanding the above result for one more step using the theory of Rayleigh quotient, we conclude that the degeneracy in the perpendicular direction to $\boldsymbol{v}_1$,..., $\boldsymbol{v}_{i-1}$ is λ_i+1, and the corresponding $\boldsymbol{v}_i$ indicates the i-th degenerate direction.

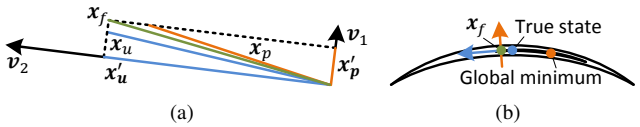


Fig. 3. (a) Illustration of solution remapping in a linear problem. (b) An example of solution remapping applied to a nonlinear problem.

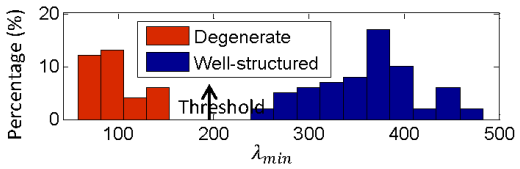


Fig. 4. Determining threshold. From one sample dataset, we calculate the distribution of $\lambda_{\min}$ in both well-conditioned scenes and degenerate scenes. The threshold is set at the mid-point of the margin between both groups.

B. Solution Remapping

In this section, we introduce a simple technique which we call solution remapping to handle degeneracy. We start with a linear problem and then discuss usage of the technique in a nonlinear problem. We first find a number m, $0 \leq m \leq n$, of eigenvalues λ_1,..., λ_m that are smaller than a threshold. Here, $m = 0$ indicates a well-conditioned system and $m = n$ indicates a completely degenerate system. Let us construct three matrices as follows,

$$\mathbf{V}_p = [\boldsymbol{v}_1, ..., \boldsymbol{v}_m, 0, ..., 0]^T, \tag{15}$$

$$\mathbf{V}_u = [0, ..., 0, \boldsymbol{v}_{m+1}, ..., \boldsymbol{v}_n]^T, \tag{16}$$

$$\mathbf{V}_f = [\boldsymbol{v}_1, ..., \boldsymbol{v}_m, \boldsymbol{v}_{m+1}, ..., \boldsymbol{v}_n]^T. \tag{17}$$

Following the convention of Kalman filters, let us define $\boldsymbol{x}_p$ as a prediction which is the best guess of the true state. Let $\boldsymbol{x}_u$ be an update obtained from solving the system equation described in (3). Here, note that even though the problem is degenerate, (3) is still solvable and yields a solution due to noise contained in the system. The key idea of solution remapping is to use $\boldsymbol{x}_p$ in the degenerate directions, $\boldsymbol{v}_1$,..., $\boldsymbol{v}_m$, and $\boldsymbol{x}_u$ in the well-conditioned directions, $\boldsymbol{v}_{m+1}$,..., $\boldsymbol{v}_n$. The final solution, $\boldsymbol{x}_f$, is a linear combination of $\boldsymbol{x}_p$ and $\boldsymbol{x}_u$,

$$\boldsymbol{x}_f = \boldsymbol{x}'_p + \boldsymbol{x}'_u, \tag{18}$$

where $\boldsymbol{x}'_p = \mathbf{V}_f^{-1}\mathbf{V}_p\boldsymbol{x}_p$ and $\boldsymbol{x}'_u = \mathbf{V}_f^{-1}\mathbf{V}_u\boldsymbol{x}_u$.

Fig. 3(a) explains the intuition behind solution remapping, in a two dimensional example. The black axes represent the eigenvectors of a system and the lengths of the axes indicate the eigenvalues. In this example, $\boldsymbol{v}_1$ is a degenerate direction and $\boldsymbol{v}_2$ is a well-conditioned direction. With (18), we project $\boldsymbol{x}_p$ onto $\boldsymbol{v}_1$ to obtain $\boldsymbol{x}'_p$, and $\boldsymbol{x}_u$ onto $\boldsymbol{v}_2$ to obtain $\boldsymbol{x}'_u$. Finally, $\boldsymbol{x}_f$ is the vector sum of $\boldsymbol{x}'_p$ and $\boldsymbol{x}'_u$. Here, the threshold determining degeneracy is calculated from one test containing both well-conditioned scenes and degenerate scenes. Fig. 4 provides an example. The threshold $\lambda_{\min}$ is generated by a scan matching method which encounters a planar environment. The threshold is set at the mid-point of the margin between both groups.

Algorithm 1: Nonlinear Solver with Solution Remapping

1 **input** : $\boldsymbol{f}$ (nonlinear function), $\boldsymbol{x}_p$ (predicted solution)
2 **output** : $\boldsymbol{x}_f$ (final solution)
3 **begin**
4 $\quad \boldsymbol{x}_f \leftarrow \boldsymbol{x}_p$; $O(*)$
5 $\quad$ Linearize $\boldsymbol{f}$ at $\boldsymbol{x}_p$ to get $\mathbf{A}$, $\boldsymbol{b}$, and $\mathbf{A}^T\mathbf{A}$; $O(*)$
6 $\quad$ Compute λ_i and $\boldsymbol{v}_i$ of $\mathbf{A}^T\mathbf{A}$, $i = 1, ..., n$; $O(n^3)$
7 $\quad$ Determine a number m of λ_i smaller than a threshold, construct $\mathbf{V}_p$, $\mathbf{V}_u$, and $\mathbf{V}_f$ based on (15)-(17); $O(n^2)$
8 $\quad$ **while** nonlinear iterations **do** $O(kn^2 + *)$
9 $\quad\quad$ Compute update $\Delta\boldsymbol{x}_u$; $O(*)$
10 $\quad\quad$ $\boldsymbol{x}_f \leftarrow \boldsymbol{x}_f + \mathbf{V}_f^{-1}\mathbf{V}_p\Delta\boldsymbol{x}_u$; $O(n^2)$
11 $\quad$ **end**
12 $\quad$ Return $\boldsymbol{x}_f$;
13 **end**

Now, let us adapt solution remapping to a nonlinear solver in Algorithm 1. The algorithm takes a nonlinear function $\boldsymbol{f}$ and a prediction $\boldsymbol{x}_p$ as input, and computes the final solution $\boldsymbol{x}_f$. Solution remapping is done by lines 6, 7, and 10. On line 6, we compute the eigenvalues λ_i and eigenvectors $\boldsymbol{v}_i$, $i = 1, ..., n$. On line 7, we construct the three matrices $\mathbf{V}_p$, $\mathbf{V}_u$, and $\mathbf{V}_f$ by comparing λ_i to a threshold. The solution is only updated in the well-conditioned directions on line 10, leaving $\boldsymbol{x}_p$ in the degenerate directions unchanged. From our experience it is not necessary to update the directions for each nonlinear iteration but the first iteration only, saving computation time. Here, we use $O(*)$ to denote the original complexity of the nonlinear solver. By introduction of solution remapping, the additional add-in complexity is stated in the following theorem.

Theorem 1: Algorithm 1 adds $O(kn^2 + n^3)$ time to the original nonlinear solver, where k is the number of iterations, and n is the dimension of the state space.

Here, note that when the dimension of the state space n is constant and relatively small, the add-in complexity becomes $O(k)$. This is often the case, for example when tracking the state under 6-DOF motion. Finally, let us give an example to explain the intuition behind Algorithm 1. As shown in Fig. 3(b), the black curves represent elevation curves. In this example, the global minimum (orange dot) is far away from the true state (blue dot) because of the degenerate constraint structure. In fact, the global minimum is determined by noise along the degenerate direction (indicated by the blue arrow). To prevent the solution from moving toward the faulty global minimum, we use $\boldsymbol{x}_p$ in the degenerate direction. The solution is only updated along the orange arrow during optimization. The result is that we find a solution as the green dot, much closer to the true state than the global minimum.

V. Vision-Lidar Ego-Motion Estimation System

A. Sensor Hardware

The study of this paper is validated on, but not limited to a custom built camera and lidar system. The camera is a uEye monochrome camera configured at 60Hz frame rate and 752×480 pixel resolution with 76° horizontal field of view. The lidar is a Hokuyo UTM-30LX laser scanner which

Fig. 5. Sensors involved in the study. The sensors are composed of an uEye camera and a custom built 3D lidar based on a Hokuyo laser scanner.

has $180°$ field of view and $0.25°$ resolution with 40 lines/sec scanning rate. A motor rotates the laser scanner at $180°/s$ average angular speed back and forth between $-90°$ and $90°$ to realize 3D scanning. An encoder measures the motor rotation angle with $0.25°$ resolution.

B. Software System

The vision and lidar integrated motion estimation system built in our previous work [19] takes visual images and lidar clouds from the sensors in Fig. 5 and estimates its ego-motion to build a map of the traversed environment. The system uses a visual odometry method running at a high frequency (60Hz) followed by scan matching at a low frequency (1Hz) to refine motion estimates, hence is able to map in real-time on the move. We have chosen this system as it combines multiple components for evaluation of the proposed method. The system combines three modules as shown in Fig. 6, each formulates and solves a nonlinear optimization problem with the Levenberg-Marquardt method [15], and is adapted with solution remapping.

1) Frame to Frame Visual Odometry: The visual odometry estimates motion between two consecutive frames. We track Harris corners [20] by the Kanade Lucas Tomasi (KLT) method [21]. The scale of translation is determined from lidar range measurements. We register lidar clouds on a depthmap in the camera field of view and associate depth to visual features from the depthmap. For a feature point, we find three points on the depthmap that form a planar patch with a KD-tree [22]. The depth is calculated by projecting a ray from the camera center to the planar patch.

When solving for motion, the method first tries to associate depth from the depthmap. However, if depth is unavailable from the depthmap, it tries to reconstruct the depth by triangulation using the estimated motion if the feature is tracked long enough. As the last choice, the method uses the feature without depth by using a different type of constraint. The motion estimation solves an optimization problem including constraints from features both with and without depth.

2) Sweep to Sweep Refinement: The refinement module matches lidar clouds between consecutive sweeps to refine the motion estimates. Here, a sweep is the process that the lidar completes for one full scan coverage, or a $180°$ rotation

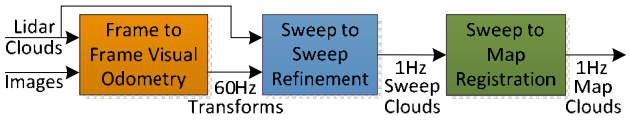


Fig. 6. Block diagram of the motion estimation software system.

of the Hokuyo laser scanner (lasting for 1s). The drift of the visual odometry is modeled with constant velocity within a sweep. This module combines a linear motion model to remove distortion in the lidar clouds caused by drift of the visual odometry.

The scan matching uses geometric features located on local edges and planar surfaces, namely edge points and planar points. It matches an edge point and a planar point from the current sweep to an edge line segment and a planar surface patch from the previous sweep. The motion refinement minimizes overall distances from the edge points and planar points to their correspondences.

3) Sweep to Map Registration: The registration module matches lidar clouds from sweeps to the current map and registers the lidar clouds to incrementally expand the map. As distortion caused by drift of the visual odometry is removed, this module simply assumes rigid body transformation, similar to the standard iterative closest point method [23]. Again, both edge points and planar points are used in scan matching.

VI. Experiments

We conduct experiments with the vision-lidar motion estimation system introduced in Section V. For each test, the sensor hardware in Fig. 5 is carried by a person who walks at a speed of 0.5m/s. The proposed *degeneracy factor* is compared with two other terms:

- *Inverse Maximum Covariance Eigenvalue* (*IMCE*): From the least-square regression theory [17], one is able to determine the covariance of a linear solution,

$$\Sigma = \sigma^2(\mathbf{A}^T\mathbf{A})^{-1}, \tag{19}$$

  where σ^2 is a variance computed from the residuals, $\sigma^2 = ||\mathbf{A}\boldsymbol{x}_0 - \boldsymbol{b}||^2/(n-m)$. Here, recall that n is the number of constraints and m is the dimension of the linear problem. Since we propose in Section IV to use the eigenvalues of $\mathbf{A}^T\mathbf{A}$ to evaluate degeneracy, let us also use the eigenvalues of Σ. Here, *IMCE* is defined as inverse of the maximum eigenvalue of Σ. Additionally, to inspect how each element contributes to the covariance, let us define another term $\mathcal{R}$ as squared sum of the residuals, $\mathcal{R} = ||\mathbf{A}\boldsymbol{x}_0 - \boldsymbol{b}||^2$.

- *Inverse Condition Number* (*ICN*): The condition number [24] of a linear system is determined by the ratio between the minimum and maximum eigenvalues of $\mathbf{A}^T\mathbf{A}$, defined as $\sqrt{\lambda_m/\lambda_1}$. The term evaluates numerical condition of a linear system. A large condition number indicates ill-conditioning. In this case, the resulting solution suffers from inaccuracy due to numerical calculation errors. For comparison purposes, we take its inverse and denote it as *ICN*.

The overall experiments consist of four tests. Test 1 validates the proposed method for visual odometry (first module in Fig. 6), Test 2 concentrates on the two scan matching sections (second and third modules in Fig. 6), and Test 3 covers all three modules. We consider failure cases of the visual odometry and classify them into motion blur,

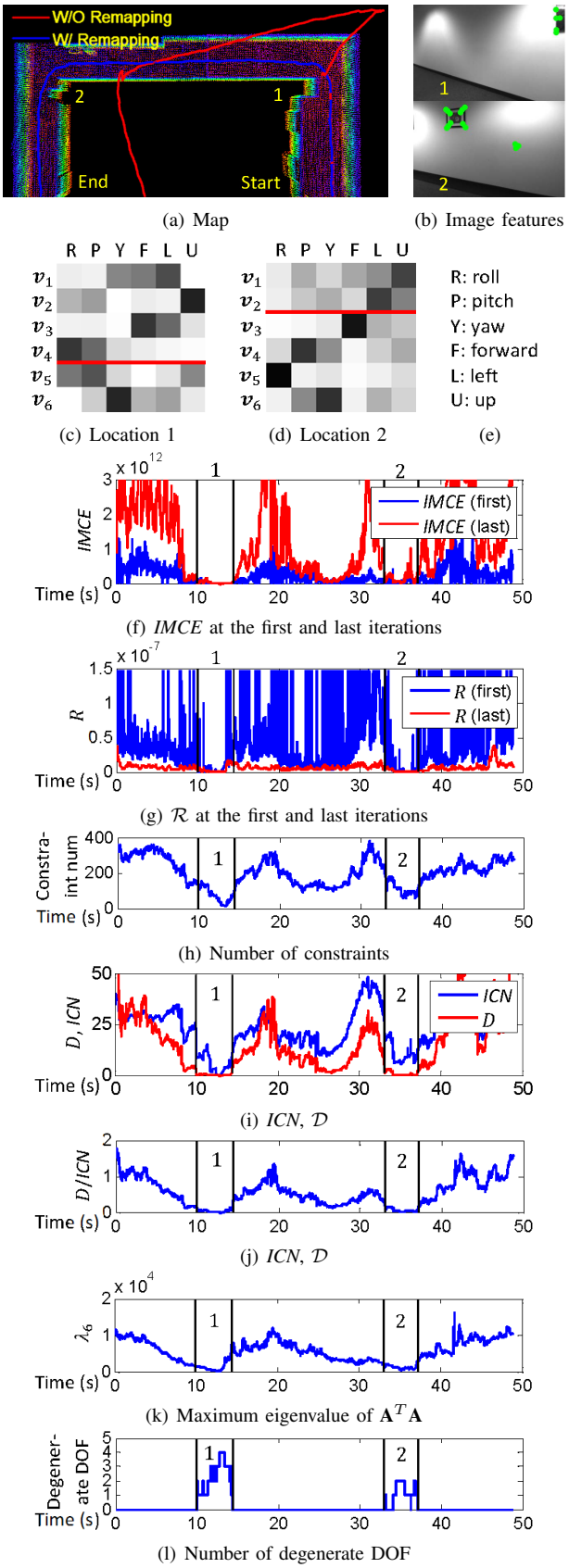
dynamic environment, and feature-poor scene. We think that motion blur can be handled by a fast camera and image frame rate, and dynamic environments can be addressed by outlier rejection or distraction suppression technique [25]. A feature-poor scene is the most relevant to degeneracy. Typically, this occurs where the camera faces a texture-less environment, points to the sun, or is in a dark environment. Likely, few features are available or the features are extracted from a concentrated area within the images.

With such consideration, Test 1 is conducted in a corridor. As shown in Fig. 7, the path goes through two feature-poor corners labeled with numbers 1 and 2. Fig. 7(a) presents the estimated trajectory and the map built when degeneracy is eliminated by solution remapping (Algorithm 1). Here, prediction is provided by a constant velocity model. Fig. 7(b) shows one sample image from each of the labeled corners. In Fig. 7(c)-(d), we show the eigenvectors of matrix $\mathbf{A}^T\mathbf{A}$ corresponding to the two images in Fig. 7(c). Darker blocks indicate larger values, and rows in top-down order correspond to small to large eigenvalues. The directions above the red lines are degenerate. Careful comparison finds that in location 1, the most degenerate direction is lateral translation (the darkest block on the first row is labeled with "L: left"). This makes sense as features in location 1 are vertically distributed, resulting in lateral translation to be poorly constrained. Correspondingly, the red trajectory jumps leftward. In location 2, Fig. 7(d) indicates degeneracy mostly in vertical translation (the darkest block on the first row is labeled with "U: up"). This is because the features span horizontally. Accordingly, the red trajectory jumps upward.

Next we examine *IMCE*. In Fig. 7(f), we show the values for both the first and last iterations in the nonlinear optimization. We see that the values of *IMCE* are noisy mostly due to the noisy nature of the squared sum of the residuals $\mathcal{R}$ (Fig. 7(g)). Note that the value at the first iteration is more meaningful as ideally we want to detect degeneracy from the beginning of the optimization so that solution remapping can be introduced. However, we also see *IMCE* at the first iteration is much noisier than at the last iteration as $\mathcal{R}$ is not yet minimized. In Fig. 7(h), we show the number of constraints, which is also involved in the computation of the covariance Σ and possibly brings in uncertainty.

Finally we compare *ICN* and *degeneracy factor* $\mathcal{D}$ in Fig. 7(i). Here, *ICN* is manually scaled to match with $\mathcal{D}$. Further, we compare the ratio of the two terms $\mathcal{D}$/*ICN* in Fig. 7(j). It is apparent that the value of the ratio reduces in locations 1 and 2, indicating $\mathcal{D}$ is more effective. The reason is that *ICN* calculates the ratio between the minimum and maximum eigenvalues of $\mathbf{A}^T\mathbf{A}$ such that the maximum eigenvalue also has effect on the term. In Fig. 7(k), we show the maximum eigenvalue λ_6 which decreases in locations 1 and 2. The reduction of λ_6 contributes to the increase of *ICN*. Here, the consideration is that degeneracy should not be determined by the well-conditioned directions but by the degenerate directions themselves. Finally, we show the number of degenerate DOF during Test 1 in Fig. 7(l).

Then, in Test 2, we choose an environment which contains a piece of flat ground as in Fig. 8. Traveling on the flat ground results in sliding of the scan matching on the red



(a) Map (b) Image features

(c) Location 1 (d) Location 2 (e)

(f) *IMCE* at the first and last iterations

(g) $\mathcal{R}$ at the first and last iterations

(h) Number of constraints

(i) *ICN*, $\mathcal{D}$

(j) *ICN*, $\mathcal{D}$

(k) Maximum eigenvalue of $\mathbf{A}^T\mathbf{A}$

(l) Number of degenerate DOF

Fig. 7. Test 1: Visual odometry degeneracy in feature-poor environment.

curve in Fig. 8(a). Correspondingly, the top three rows of Fig. 8(c)-(d) indicate that degeneracy occurs in the directions of forward translation, lateral translation, and yaw rotation, meaning that translation parallel to the ground and rotation perpendicular to the ground are poorly constrained.

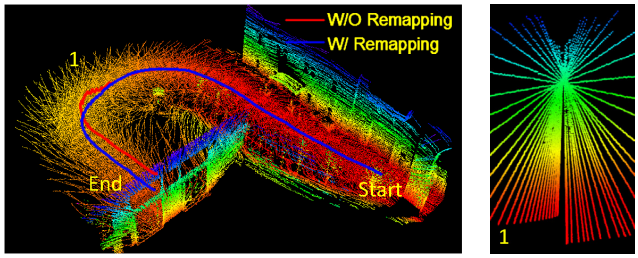
Looking into the rest of the figure, we find that the value of *IMCE* is either noisy or does not decrease obviously (for the sweep to sweep refinement section in Fig. 8(f) and the sweep to map registration section in Fig. 8(g)). Again, this is because *IMCE* is determined by multiple terms including squared sum of the residuals $\mathcal{R}$ and the number of constraints. Here, note that the values of *IMCE* do not differ much between the first and last iterations because the scan matching only refines motion estimates generated by the visual odometry. The value of R reduces little during the course of optimization. Fig. 8(h)-(i) compare *ICN* and $\mathcal{D}$. We can see an obvious drop of value between 35-70s when the degeneracy occurs. In Fig. 8(j), the ratio $\mathcal{D}$/*ICN* also decreases slightly during this interval. In Fig. 8(k), we see that the method detects three degenerate DOF corresponding to the top three rows in Fig. 8(c)-(d).

Finally, we conduct a larger scale test in Test 3 containing indoor and outdoor environments, as shown in Fig. 9. The path starts in front of a building, passes through the building and exits to the outside, climbs stairs, and follows a small trail to come back and finish at the exact starting position. The overall traveling distance is 538m. The path contains two degenerate scenes for the visual odometry in locations 1 and 3 due to undesirable lighting conditions, and two degenerate scenes for the scan matching in locations 2 and 4. In location 2, the lidar sees the flat ground and one wall on its right side causing degeneracy in forward translation. In location 4, the lidar only sees the flat ground similar to Test 2. We observe that the value of $\mathcal{D}$ drops in locations 1 and 3 in Fig. 9(c) and in locations 2 and 4 in Fig. 9(d)-(e).

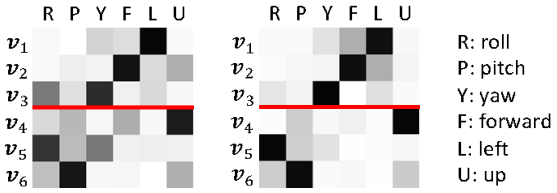
Fig. 10 further compares the estimated trajectories. The green curve is without solution remapping, hence jumps occur along the path. The red curve is estimated with consistent motion prior added to the motion estimation problems. The visual odometry section takes motion prior from a constant velocity model, and each scan matching section takes output from the previous section. Here, the motion prior is given as little as possible to eliminate degeneracy. However, it still causes drift at the end to be about three times as large as on the blue curve (proposed method). Thanks to solution remapping, the system is able to conquer all degeneracy resulting in a 0.71% relative position error at the end of the blue curve compared to the distance traveled.

VII. Discussion

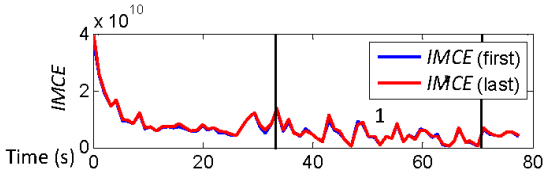
As the solution remapping linearly combines the prediction and update, one can argue that this is a variant of the Kalman filter. Our response is that a filter handles accuracy, while the proposed method is devoted to degeneracy and robustness. The difference is that filters average multiple noisy measurements to gain better accuracy. When working with degeneracy, we consider the solution to be completely unusable in the degenerate directions and simply take the prediction instead. Our second response is that solution



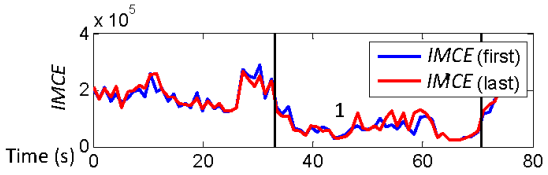
(a) Map (b) Lidar scan



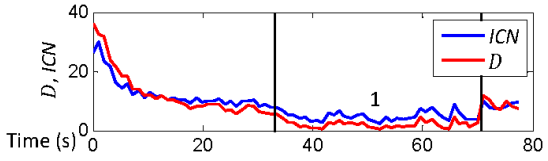
(c) Sweep to sweep (d) Sweep to map (e)



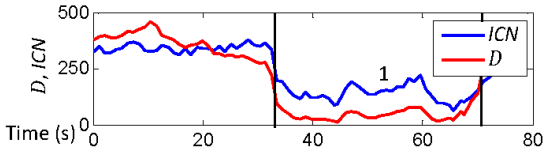
(f) *IMCE* of sweep to sweep refinement section



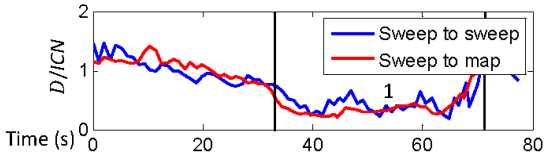
(g) *IMCE* of sweep to map registration section



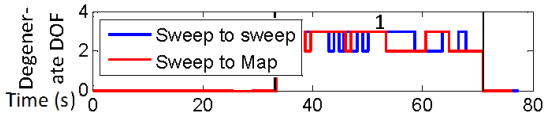
(h) *ICN*, $\mathcal{D}$ of sweep to sweep refinement section



(i) *ICN*, $\mathcal{D}$ of sweep to map registration section

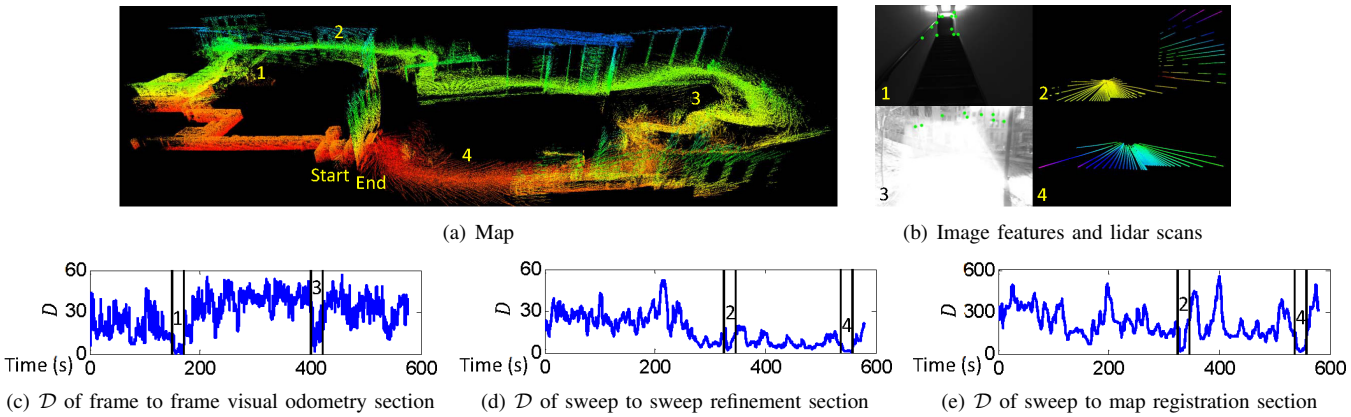


(j) $\mathcal{D}$/*ICN* for both sections



(k) Number of degenerate DOF

Fig. 8. Test 2: Scan matching degeneracy on flat ground.



(a) Map (b) Image features and lidar scans

(c) $\mathcal{D}$ of frame to frame visual odometry section (d) $\mathcal{D}$ of sweep to sweep refinement section (e) $\mathcal{D}$ of sweep to map registration section

Fig. 9. Test 3: Complete test including indoor and outdoor environments. The overall path is 538m long, starting in front of a building, passing through the building with stairs and following a trail on hilly terrain to return to the start. The path encounters four degenerate scenes labeled with 1-4.

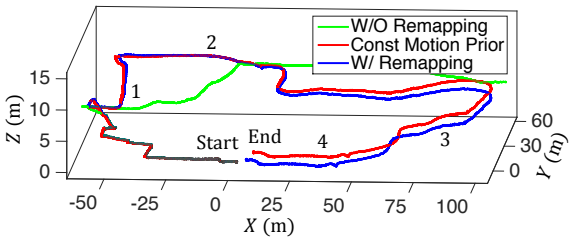


Fig. 10. Trajectories of Test 3. The green curve is estimated without solution remapping, and jumps occur along the path. The red curve uses consistent motion prior to eliminate degeneracy. This results in more drift consequently. The blue curve uses the proposed solution remapping. The position error at the end is 0.71% of the 538m trajectory length.

remapping can be adapted to individual iterations of non-linear optimization. A filter only takes the final solutions to seed steps. Finally, a filter can be the following step of the proposed method taking its output for further integration.

VIII. Conclusion

Robustness of estimation is critical for state estimation and especially for autonomous vehicles. This paper improves robustness by handling environmental degeneracy. A *degeneracy factor* is mathematically defined and derived. Degeneracy is evaluated through computation of the associated eigenvalues and eigenvectors. When degeneracy occurs, the proposed method automatically separates the state space and partially solves the problem only in well-conditioned directions. In degenerate directions, a best guess is used instead. The method is tested with a custom-built vision and lidar system in a number of challenging scenarios, for online positioning and mapping. Experimental results show that the system is able to conquer environmentally degenerate moments, to reliably estimate state, and to use the state to build accurate 3D representations of the environment.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, The MIT Press, 2005.

[2] Z. Dong and Z. You, "Finite-horizon robust Kalman filtering for uncertain discrete time-varying systems with uncertain-covariance white noises," *IEEE Signal Processing Letters*, vol. 13, no. 8, pp. 493–496, 2006.

[3] A. Eudes and M. Lhuillier, "Error propagations for local bundle adjustment," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 2411–2418.

[4] A. Censi, "On achievable accuracy for pose tracking," in *IEEE Intl. Conf. on Robotics and Automation*, 2009, pp. 1–7.

[5] ——, "On achievable accuracy for range-finder localization," in *IEEE Intl. Conf. on Robotics and Automation*, 2007, pp. 4170–4175.

[6] A. Isidori, *Nonlinear control systems*. Springer, 1999, vol. 2.

[7] P. Ioannou and J. Sun, *Robust adaptive control*. Dover Pub., 2012.

[8] P. Parks, "A. M. Lyapunov's stability theory–100 years on," *IMA Journal of Math. Control and Info.*, vol. 9, no. 4, pp. 275–303, 1992.

[9] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 1990, pp. 384–389.

[10] J. Fink, N. Michael, S. Kim, and V. Kumar, "Planning and control for cooperative manipulation and transportation with aerial robots," *The Intl. Journal of Robotics Research*, vol. 30, no. 3, pp. 324–334, 2011.

[11] P. Sujit, S. Saripalli, and J. Sousa, "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, 2014.

[12] J. J. d'Azzo and C. D. Houpis, *Linear control system analysis and design: conventional and modern*. McGraw-Hill Higher Edu., 1995.

[13] B. Hinson and K. Morgansen, "Observability optimization for the nonholonomic integrator," in *American Control Conference (ACC)*, June 2013.

[14] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust RGB-D SLAM algorithm," in *IEEE/RSJ Intl. Conf. on Intel. Robots and Systems (IROS)*, Vilamoura, Portugal, Oct. 2012.

[15] J. Nocedal and S. Wright, *Numerical Optimization*. New York, Springer-Verlag, 2006.

[16] S. Chandrasekaran and I. Ipsen, "Perturbation theory for the solution of systems of linear equations," Yale University, Tech. Rep., 1991.

[17] L. Trefethen and D. B. III, *Numerical linear algebra*. SIAM, 1997.

[18] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[19] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

[20] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, Cambridge University Press, 2004.

[21] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, 1981, pp. 121–130.

[22] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computation Geometry: Algorithms and Applications*. Springer, 2008.

[23] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Autonomous Robots*, 2013.

[24] E. Cheney and D. Kincaid, *Numerical mathematics and computing*. Cengage Learning (6th Edition), 2007.

[25] C. McManus, W. Churchill, A. Napier, B. Davis, and P. Newman, "Distraction suppression for vision-based pose estimation at city scales," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 3762–3769.