SLAM STATE ESTIMATION IN DEGENERATE SCENES

ZHANG, JI, MICHAEL KAESS, AND SANJIV SINGH. "ON DEGENERACY OF OPTIMIZATION-BASED STATE ESTIMATION PROBLEMS." ROBOTICS AND AUTOMATION (ICRA), 2016 IEEE INTERNATIONAL CONFERENCE ON. IEEE, 2016.

LU YU

IMPORPHEUS.AI

iMorpheus.ai

STATE ESTIMATION IN SLAM

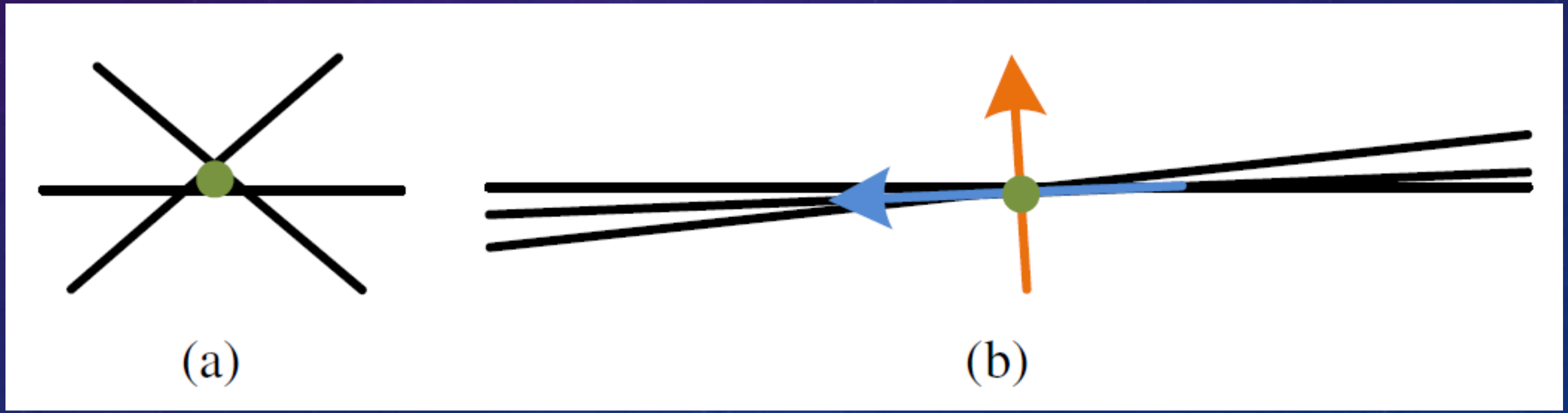
- In SLAM methods, we usually use frame by frame comparison to estimate sensor pose transform or state vector
- This is commonly done by comparing and matching texture feature (visual) or geometric feature (point cloud) between frames
- To estimate state vector, we minimize a cost function that comes from observation and puts constraints on state vector
- The cost function can be linearized

$$\arg\min_{\boldsymbol{x}} f^2(\boldsymbol{x}) \qquad \mathbf{J} = \partial f(\boldsymbol{x}) / \partial \boldsymbol{x} \qquad \arg\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$$

iMorpheus.ai

DEGENERACY

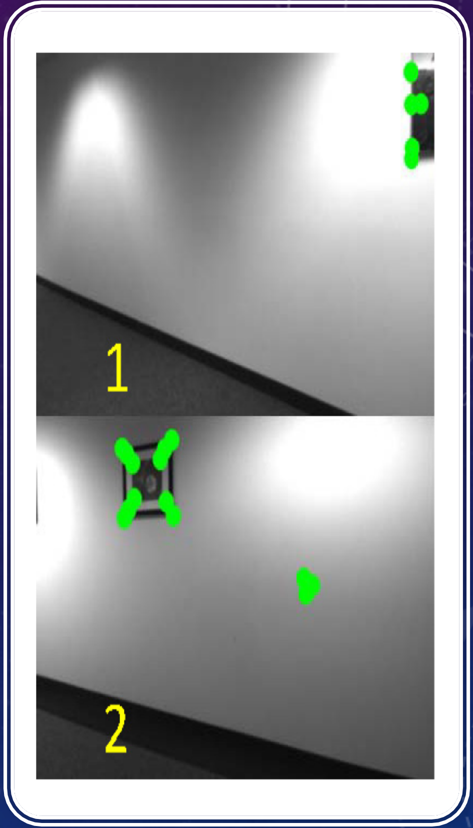
- Degeneracy occurs when constraints are ill-conditioned in some directions of state vector space
- We attempt to detect and separate these directions
- We only solve the problem in well-conditioned directions



iMorpheus.ai

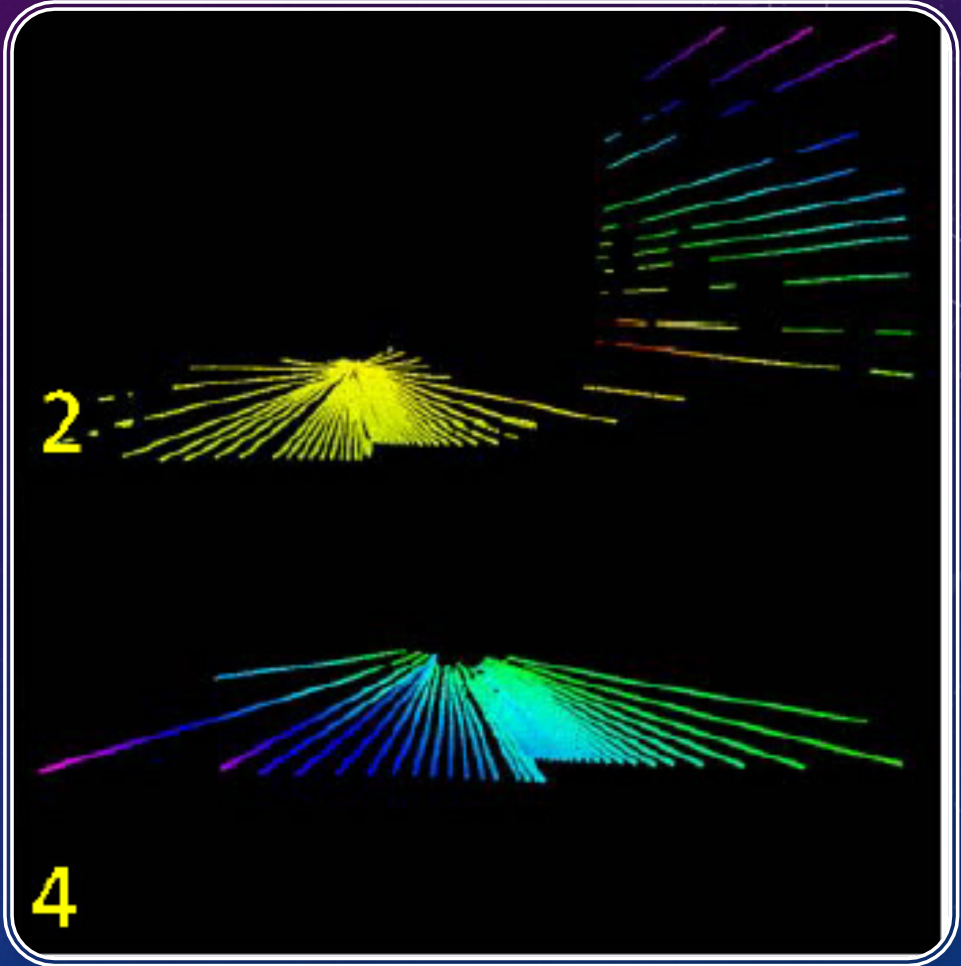
DEGENERATE SCENE - VISUAL

- Visual degeneracy occurs when there are not enough texture features
- Examples: large area of white wall, camera pointing to the sun, dark environment



DEGENERATE SCENE - POINT CLOUD

- Point cloud (LiDAR) degenerate scene occurs when there are not enough geometric features
- Example: large area of ground, plane or other planar object



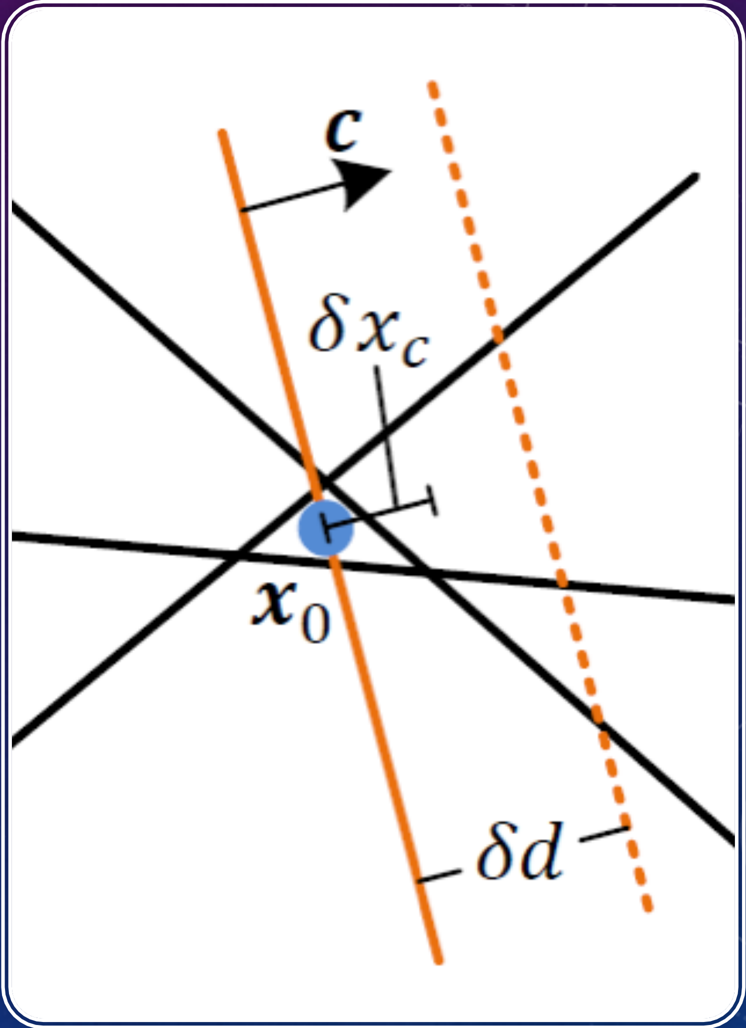
DEGENERACY DIRECTION

- Measures how sensitive the solution is to a small perturbation
- Additional constraint

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

- Make perturbation δd along c
- Find maximum shift direction

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



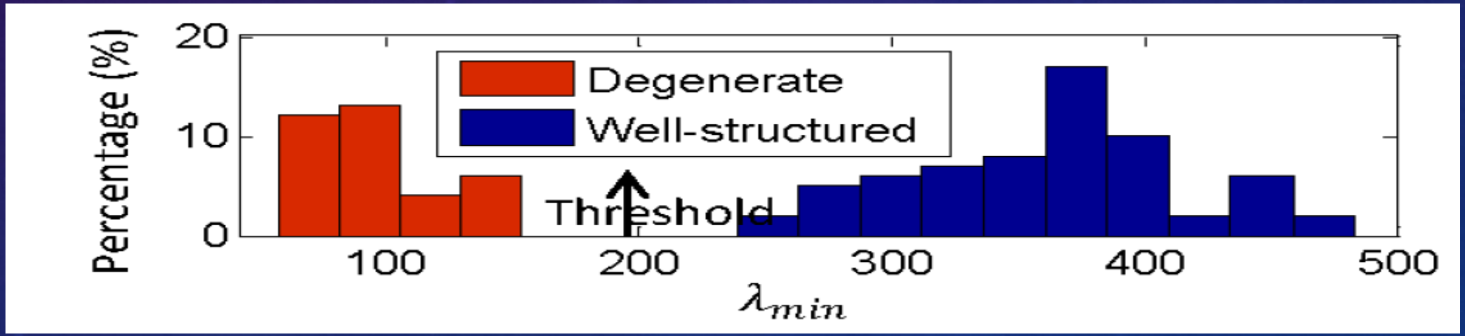
DEGENERACY FACTOR

- Maximum shift δx_c^* gives the least stable direction of the solution, which is the direction degeneracy is defined
- Degeneracy factor is the ratio of perturbation to maximum solution shift $\mathcal{D} = \delta d / \delta x_c^*$
- Greater degeneracy factor means more stable solution
- For linearized problem $\arg\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$, degeneracy factor D only depends on A, but not b
- For linearized problem $\arg\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$, degeneracy factor $D = \lambda_{min} + 1$, where λ_{min} is the smallest eigenvalue of $A^T A$

iMorpheus.ai

EIGENVALUE AND EIGENVECTOR

- In linearized problem $\arg\min_{x} \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$, we can assume matrix A is non singular, thus $A^T A$ is positive definite
- Eigenvalues of $A^T A$ are listed, in increasing order, as $\lambda_1, \ldots, \lambda_n$ and the corresponding eigenvectors are $v_1, \ldots, v_n$
- Each eigenvalue corresponds to one direction (dimension) in state space
- Find certain threshold to determine which dimensions are degenerate



iMorpheus.ai

SOLUTION REMAPPING

- Suppose the smallest m eigenvalues are determined to be degenerate, construct three matrices, which represent degenerate, non-degenerate, whole space respectively

$$\mathbf{V}_p = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, 0, \ldots, 0]^T,$$
$$\mathbf{V}_u = [0, \ldots, 0, \boldsymbol{v}_{m+1}, \ldots, \boldsymbol{v}_n]^T,$$
$$\mathbf{V}_f = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, \boldsymbol{v}_{m+1}, \ldots, \boldsymbol{v}_n]^T$$

- x_p is the best guess for true state and x_u is the unaltered solution
- x_p is given by some a priori model, such as constant velocity model
- The final solution is $\boldsymbol{x}_f = \boldsymbol{x}'_p + \boldsymbol{x}'_u$, where $\boldsymbol{x}'_p = \mathbf{V}_f^{-1}\mathbf{V}_p\boldsymbol{x}_p$ and $\boldsymbol{x}'_u = \mathbf{V}_f^{-1}\mathbf{V}_u\boldsymbol{x}_u$

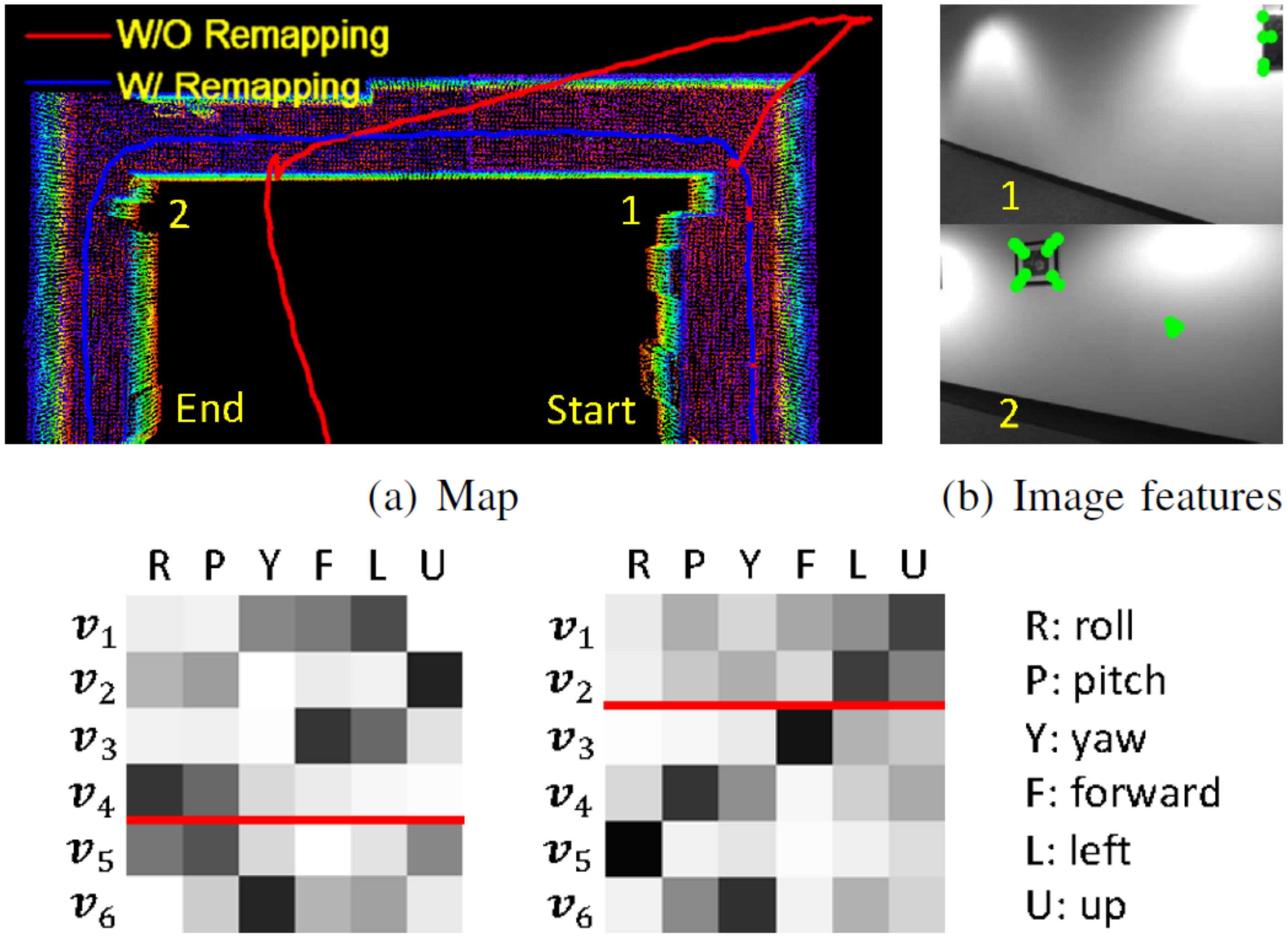
iMorpheus.ai

Algorithm 1: Nonlinear Solver with Solution Remapping

input : $\boldsymbol{f}$ (nonlinear function), $\boldsymbol{x}_p$ (predicted solution)
output : $\boldsymbol{x}_f$ (final solution)
begin
    $\boldsymbol{x}_f \leftarrow \boldsymbol{x}_p$; $O(*)$
    Linearize $\boldsymbol{f}$ at $\boldsymbol{x}_p$ to get $\mathbf{A}$, $\boldsymbol{b}$, and $\mathbf{A}^T\mathbf{A}$; $O(*)$
    Compute λ_i and $\boldsymbol{v}_i$ of $\mathbf{A}^T\mathbf{A}$, $i = 1, ..., n$; $O(n^3)$
    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); $O(n^2)$
    while nonlinear iterations **do** $O(kn^2 + *)$
        Compute update $\Delta\boldsymbol{x}_u$; $O(*)$
        $\boldsymbol{x}_f \leftarrow \boldsymbol{x}_f + \mathbf{V}_f^{-1}\mathbf{V}_p\Delta\boldsymbol{x}_u$; $O(n^2)$
    end
    Return $\boldsymbol{x}_f$;
end

ALGORITHM

- Local linearization for non linear system
- Degenerate directions only calculated once
- This algorithm adds $O(kn^2 + n^3)$ in time complexity, where k is iteration steps and n is dimension of state space

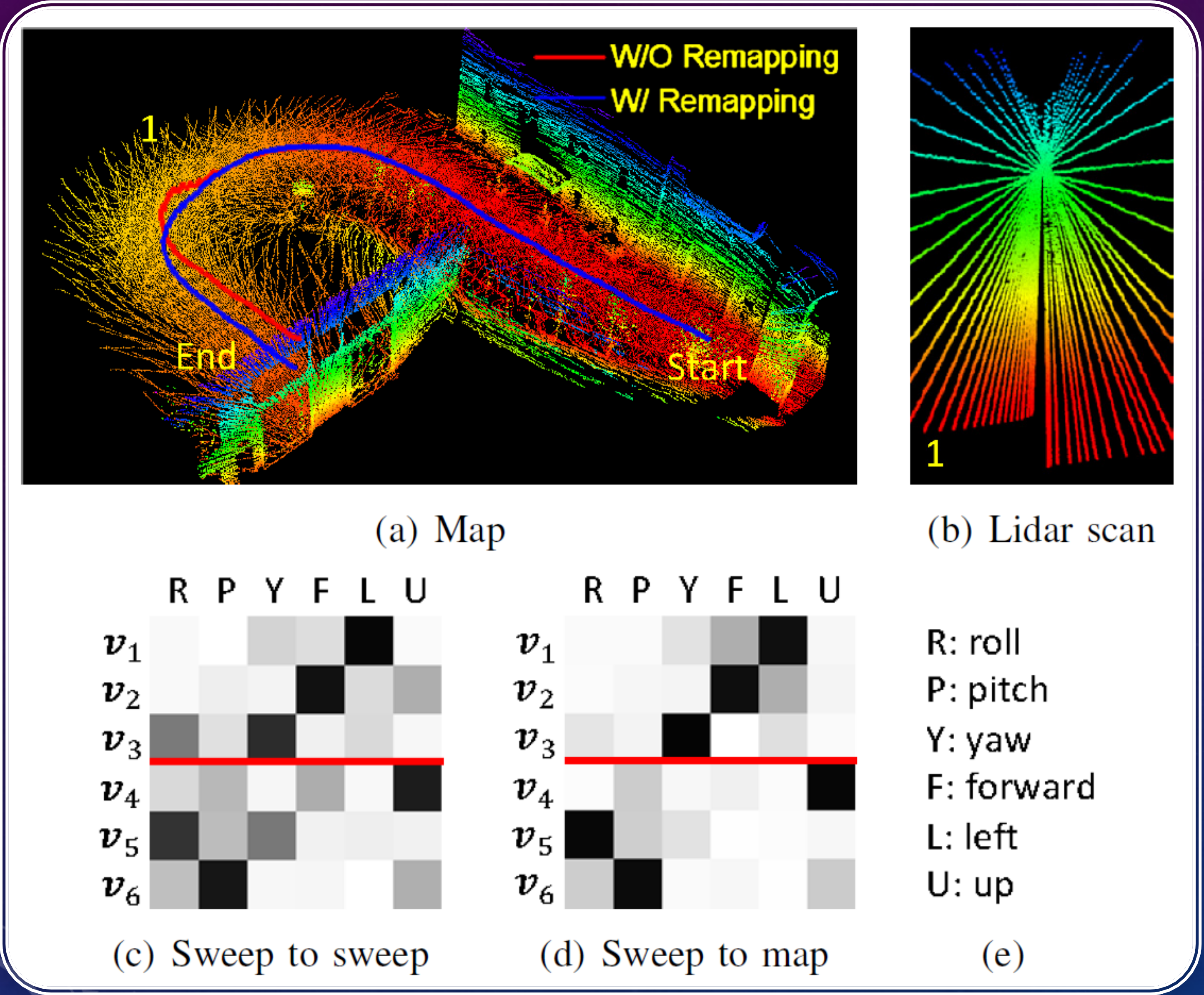


(a) Map (b) Image features
(c) Location 1 (d) Location 2 (e)

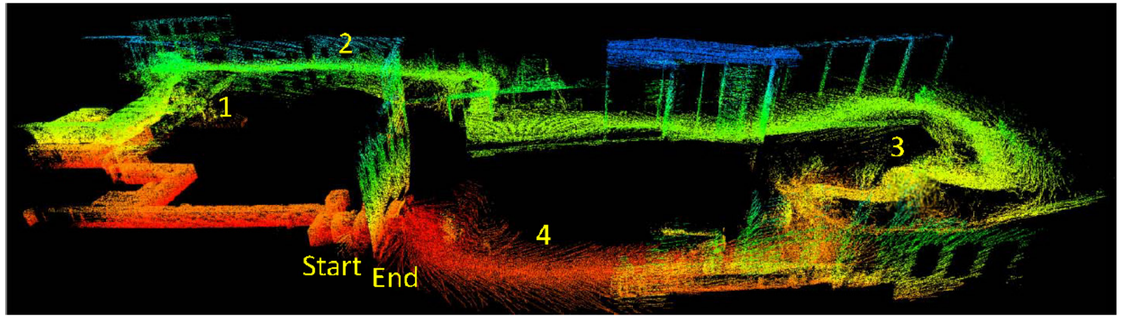
EXPERIMENT - VISUAL

- Two feature-poor scenes 1 and 2, with four and two degenerate directions respectively
- Huge error at 1 and small error at 2 without remapping
- Constant velocity model

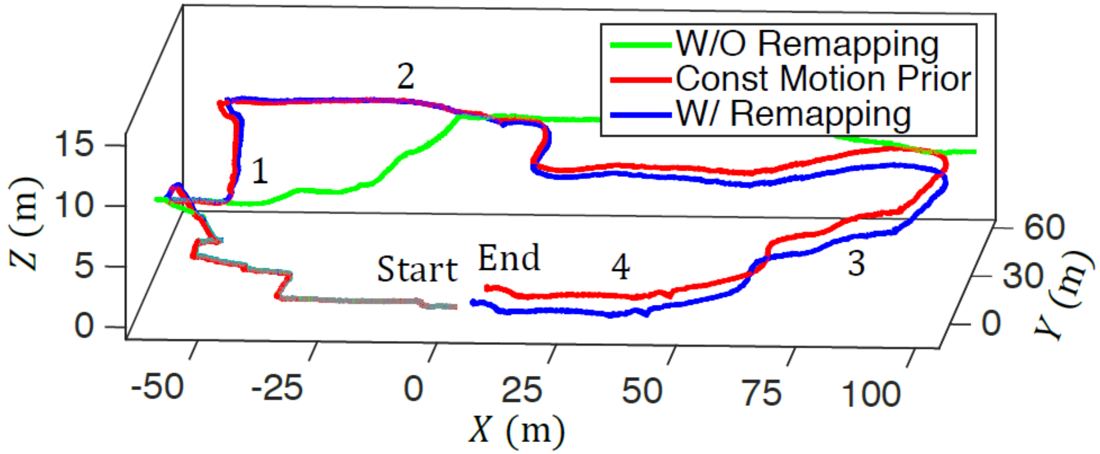
EXPERIMENT - POINT CLOUD



- Planar scene at 1 (flat ground)
- Odometry error around scene 1
- Constant velocity model



(a) Map

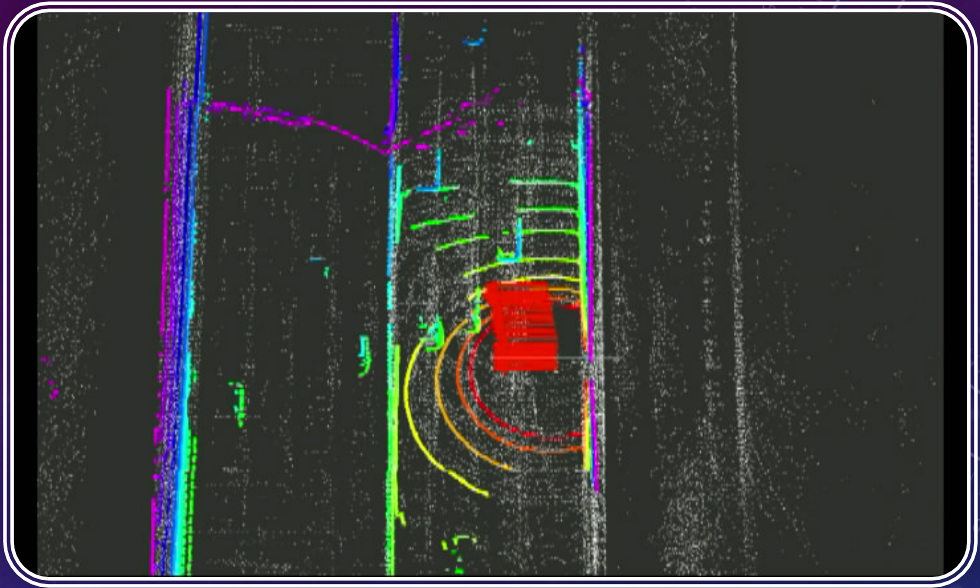


EXPERIMENT - VISUAL AND POINT CLOUD

- Visual degeneracy at 1 and 3 due to poor lighting
- Point cloud degeneracy at 2 and 4 due to flat ground/wall
- Total distance 538 meters and final relative position error 0.71% (3.8 meters)

OUR EXPERIENCE

- Our drive test in a tunnel with LiDAR point cloud
- Scene dominated by a flat wall on one side and moving vehicles (dynamic noise)
- LOAM produces stationary or slightly backwards odometry



OUR PLANNED TEST

- In LOAM algorithm, current frame pose is estimated by combining edge point planar point constraints $f(\boldsymbol{T}_k^L(t)) = \boldsymbol{d}$
- The Jacobian is calculated as $\mathbf{J} = \partial f / \partial \boldsymbol{T}_k^L(t)$
- So we can calculate and order eigenvalues of $J^T J$ to detect degeneracy
- Actual threshold needs to be tested and determined depending on scene
- Dynamic noise (moving vehicles, cyclists, people) is still a concern

iMorpheus.ai

FINAL THOUGHTS

- Runs efficiently if dimension of state space is small (time complexity is $O(kn^2 + n^3)$)
- Threshold of eigenvalues need to be chosen based on experiments and thus depends on scenes
- Able to detect degeneracy, but correction is another story
- In degenerate directions, prediction (e.g. constant velocity model) is used to estimate state
- Other sensor, such as IMU or odometer, can provide more accurate prediction than constant velocity model
- Other than degeneracy, dynamic noise is another obstacle to SLAM

iMorpheus.ai

IMORPHEUS JOURNAL CLUB

Next Friday, 22/12/2017 12:00PM GMT+8

Enabling Aggressive Motion Estimation at Low-drift and Accurate Mapping in Real-time

Website : **http://imorpheus.ai**
Email Address : **live@imorpheus.ai**

iMorpheus.ai