Robust Multi-Stereo Visual Inertial Odometry

Joshua Jaekel and Michael Kaess

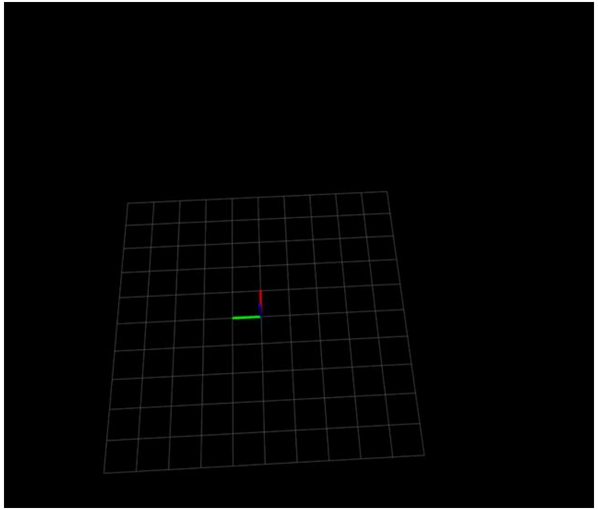
Presented by: Paloma Sodhi

Motivation

VIO algorithms continue to produce impressive results in terms of accuracy.

However, most VIO algorithms are still prone to failure under certain conditions.

Question:
Can we improve the robustness of VIO by using multiple cameras with non-overlapping FOVs?



Motivation

Using information from multiple cameras makes it more likely we maintain good visual features even if any individual camera loses them.

How do we best use information from multiple cameras?

Run multiple independent VIO algorithms

Approach 1

Run a single optimization using features from all cameras

Approach 2

Potential approaches

Approach 1	Approach 2
● Relatively simple to use and integrate ● Requires some mechanism to fuse multiple odometry estimates or switch between them ● Still difficult to recover if any single odometry estimate fails	● Optimizing on all features jointly provides better constraints on odometry ● No need to select between different odometry estimates ● Need to select features from multiple cameras to optimize

Potential approaches

Approach 1	Approach 2
● Relatively simple to use and integrate	● Optimizing on all features jointly provides better constraints on odometry
● Requires some mechanism to fuse multiple odometry estimates or switch between them	● No need to select between different odometry estimates
● Still difficult to recover if any single odometry estimate fails	● Need to select features from multiple cameras to optimize

Contributions

We propose a novel 1-point RANSAC algorithm which jointly selects features across all stereo pairs

We describe our full VIO pipeline which includes a backend based on information sparsification

VIO Overview

Feature Tracking

Joint RANSAC

Backend Optimization

KLT tracking on Shi-Tomasi corners detected in each image. Tracking is done temporally and between images in a stereo pair.

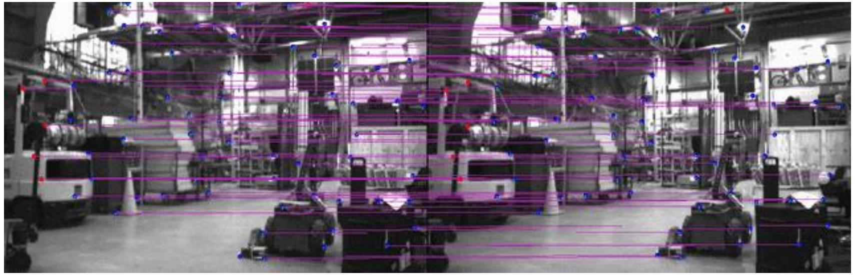
The proposed RANSAC scheme jointly selects a subset of inliers from features located across all stereo pairs.

We use a fixed-lag smoother and make use of information sparsification to accurately solve for the pose of the robot.

Feature Tracking

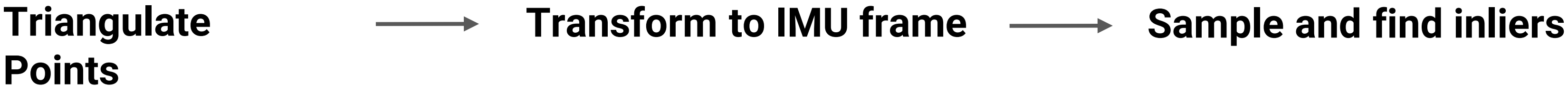
Our feature tracking pipeline is comprised of the following steps:

1. Feature detection and bucketing
2. KLT matching temporally between images
3. KLT matching between images in stereo pair
4. Replenish buckets with new features



Multi-Stereo RANSAC

Given a set of tracked feature points in each stereo pair, we can formulate RANSAC as 3 steps:

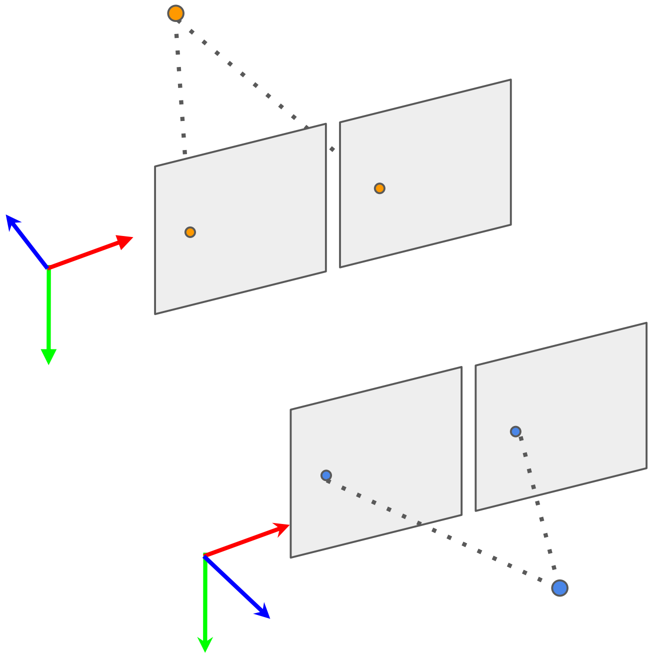


Triangulate Points → **Transform to IMU frame** → **Sample and find inliers**

Multi-Stereo RANSAC

Each stereo feature is triangulated in its respective camera frame using DLT.

We keep track of the triangulated points in the current frame and their correspondences at the previous time step.



Triangulate Points → Transform to IMU frame → Sample and find inliers

Multi-Stereo RANSAC

For each 3D correspondence we:

- Transform it into the IMU frame using the camera extrinsics
- Rotate points from the previous IMU frame into current IMU frame using integrated gyroscope measurements

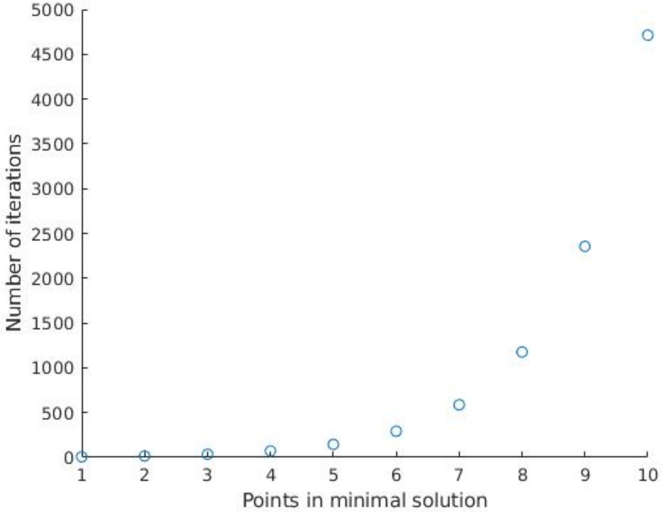
At this point corresponding feature points should only differ by temporal translation

Triangulate Points → **Transform to IMU frame** → Sample and find inliers

Multi-Stereo RANSAC

Using any single point correspondence we can estimate temporal translation. This means our RANSAC model is fully defined by only one point correspondence!

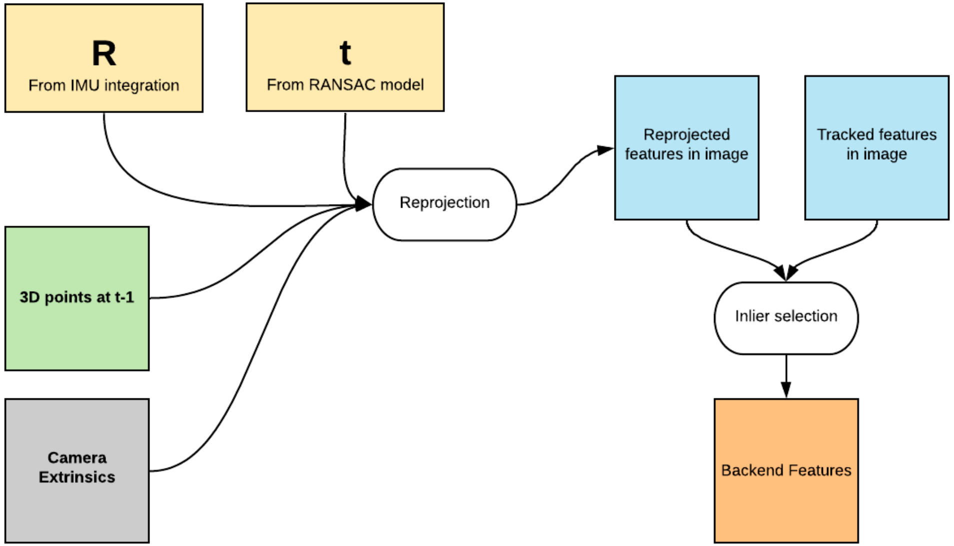
$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}$$



Triangulate Points ⟶ Transform to IMU frame ⟶ **Sample and find inliers**

Multi-Stereo RANSAC

Using our estimates of temporal rotation and translation we can reproject 3D features from the previous time step into the current image and determine inliers



Triangulate Points → Transform to IMU frame → **Sample and find inliers**

Uncertainty Compensation

In the process of performing RANSAC we use camera extrinsics twice:

1. Transforming 3D features from individual camera frames to the IMU frame
2. Reprojecting 3D points back to the original images to determine inliers

What happens if these extrinsics are noisy?

Uncertainty Compensation

In the process of performing RANSAC we use camera extrinsics twice:

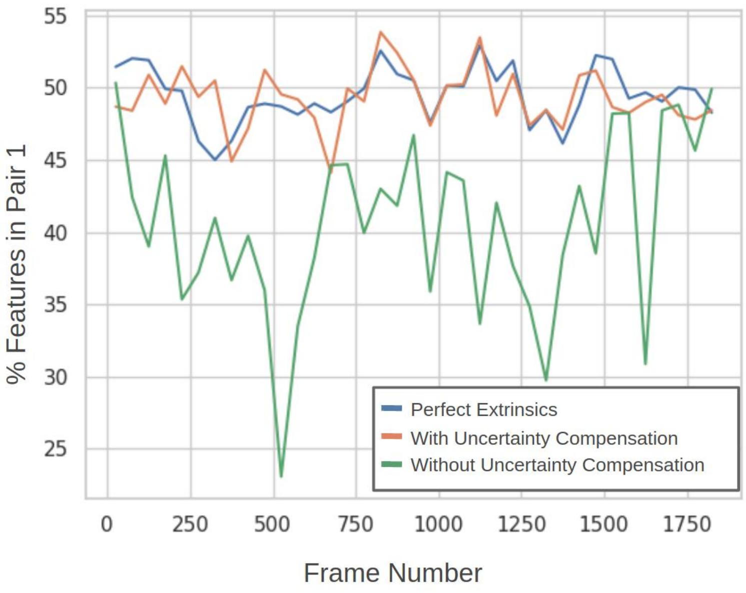
1. Transforming 3D features from individual camera frames to the IMU frame
2. Reprojecting 3D points back to the original images to determine inliers

What happens if these extrinsics are noisy?

Features observed by cameras with uncertain extrinsics will tend to have a higher reprojection error than those observed by cameras with accurate extrinsics

Uncertainty Compensation

In simulated environments we have perfect control over extrinsics and can directly observe the effects of noisy extrinsics!



Uncertainty Compensation

Represent uncertain extrinsic parameter as a member of SE(3) perturbed by Gaussian noise,

$$\mathbf{T}_{S_i}^{B} = \exp(\xi_i^{\wedge})\bar{\mathbf{T}}_{S_i}^{B}$$

$$\xi_i \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{S_i}^{B})$$

By propagating the uncertainty through the coordinate frame transformation and nonlinear reprojection function, we get a covariance in the pixel space of the left image [1,2].

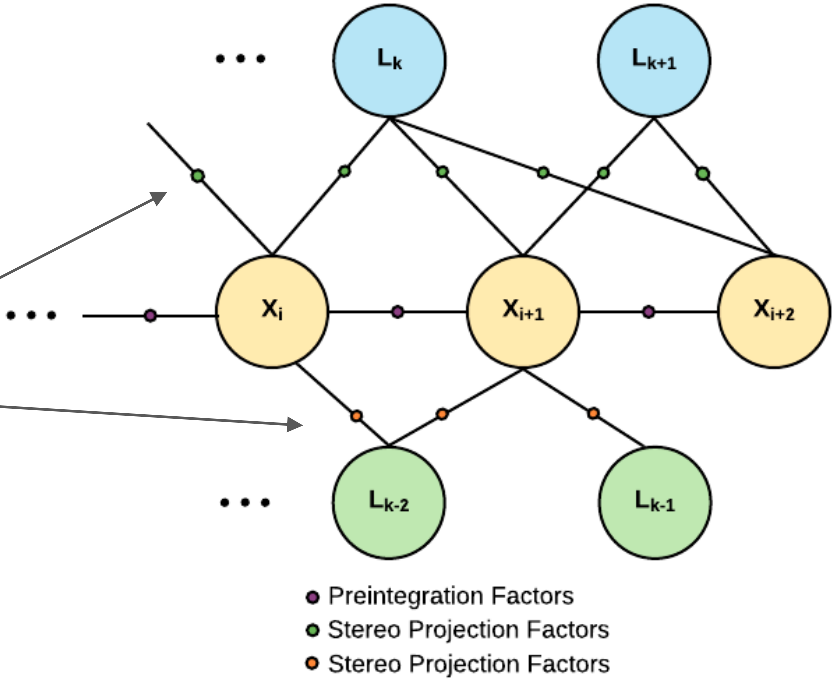
$$\mathbf{\Sigma}_{S_i}^{B} \longrightarrow \mathbf{\Upsilon}_{\tilde{\mathbf{p}}_L}$$

Rather than computing inliers using Euclidean distance to compute inliers, we use Mahalanobis distance in the pixel space.

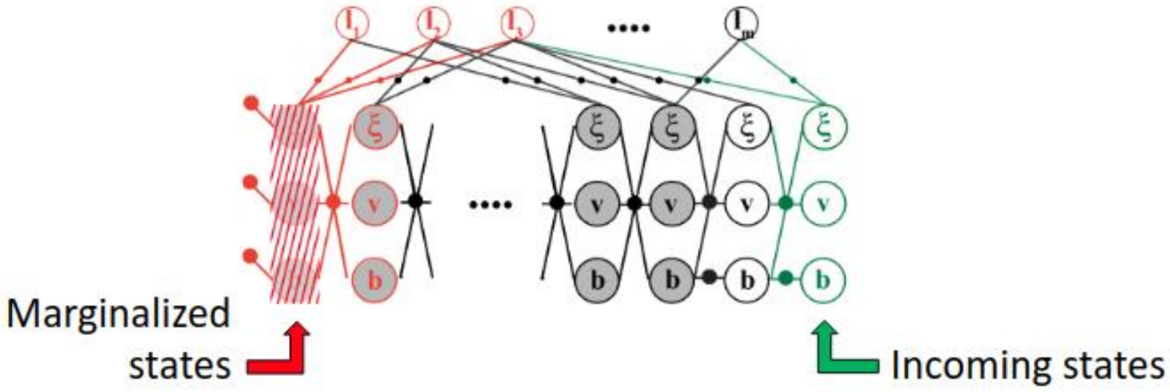
1. Associating uncertainty with three-dimensional poses for use in estimation problems. Barfoot and Furgale.
2. Characterizing the uncertainty of jointly distributed poses in the lie algebra. Mangelson et al.

Uncertainty Compensation

Noise model of each projection factor is modified to account for the extrinsic uncertainty



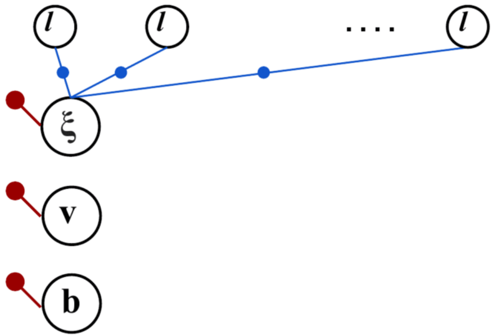
Backend Using Information Sparsification

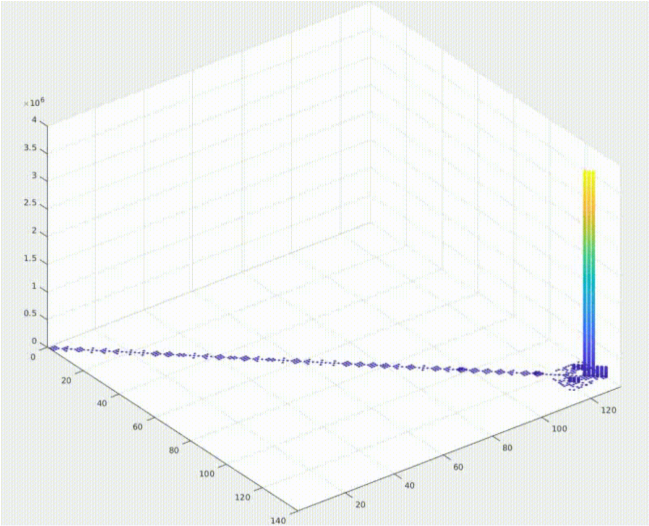


- Formulate backend as a factor graph with pre-integration factors, IMU bias random walk factors, and stereo projection factors

- At each new image frame we marginalize out a node from the smoother window and add a node for the incoming state

Backend Using Information Sparsification

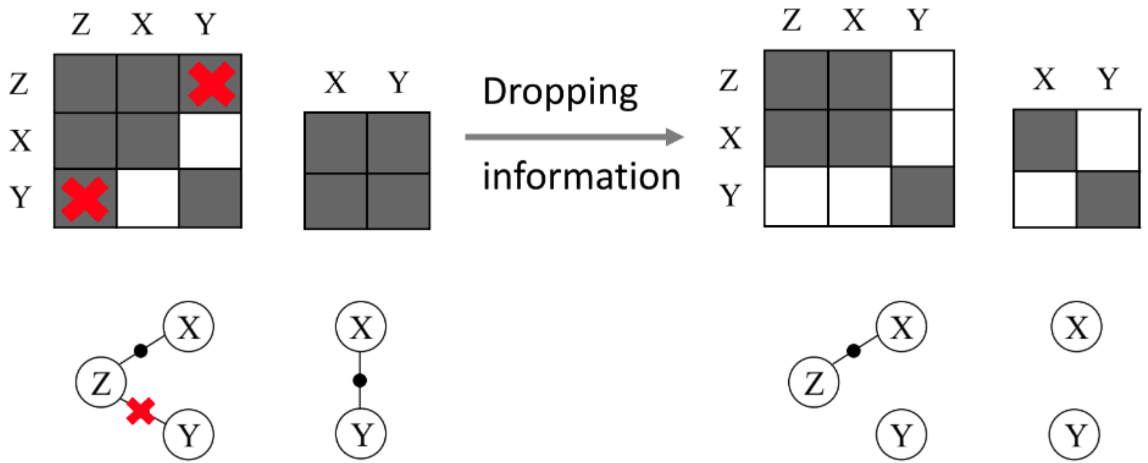




Marginalization creates densely connected priors which creates “fill-in” in the information matrix!

Backend Using Information Sparsification

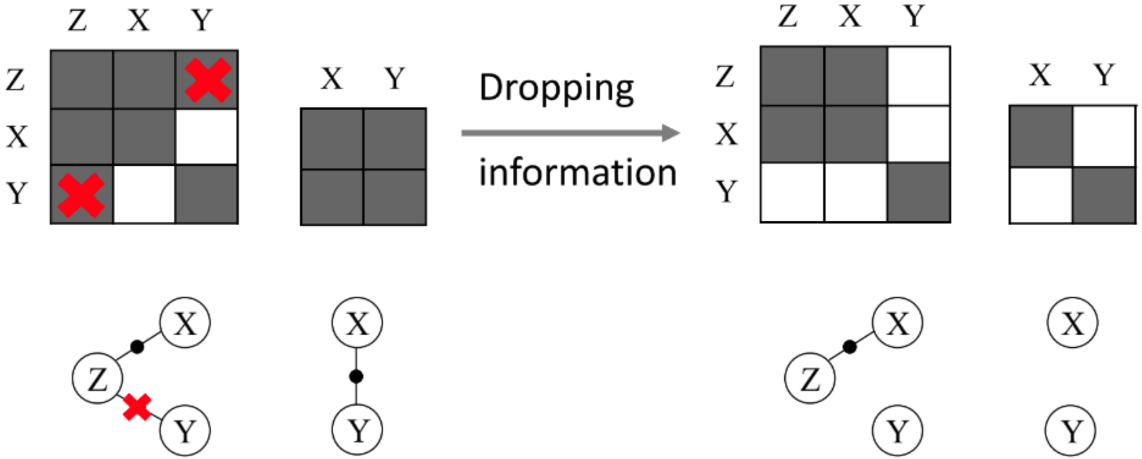
A typical approach to deal with the densified information matrix is to selectively discard entries in the information matrix



Dropping information

Backend Using Information Sparsification

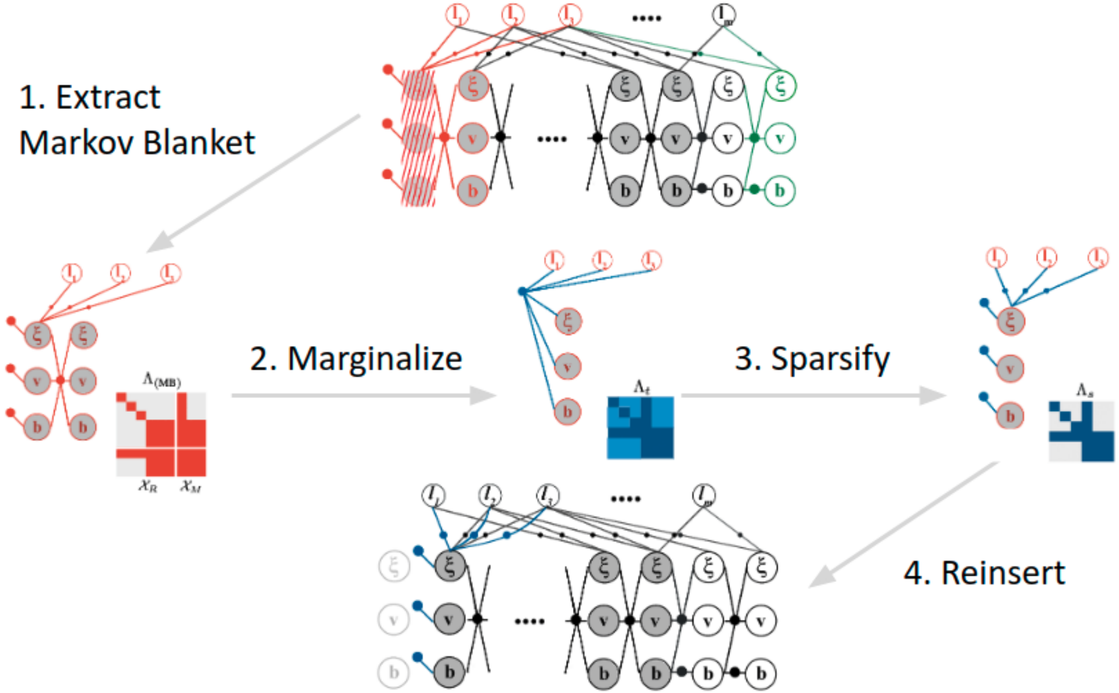
A typical approach to deal with the densified information matrix is to selectively discard entries in the information matrix



Dropping information

Can we do better?

Backend Using Information Sparsification


1. Extract Markov Blanket

2. Marginalize

3. Sparsify

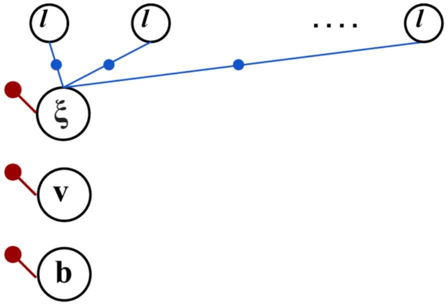
4. Reinsert

1. Nonlinear Factor Recovery for Long-Term SLAM. Maruzan et al.
2. Information Sparsification in Visual-Inertial Odometry. Hsiung et al.

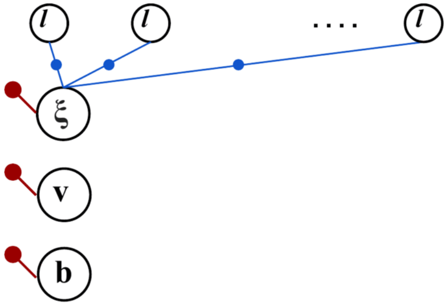
Backend Using Information Sparsification

We sparsify the information matrix using the method described in [1] by enforcing a sparse topology on priors without losing the information encoded in the dense priors

Fixed-Lag Smoother



Sparsified Fixed-Lag Smoother

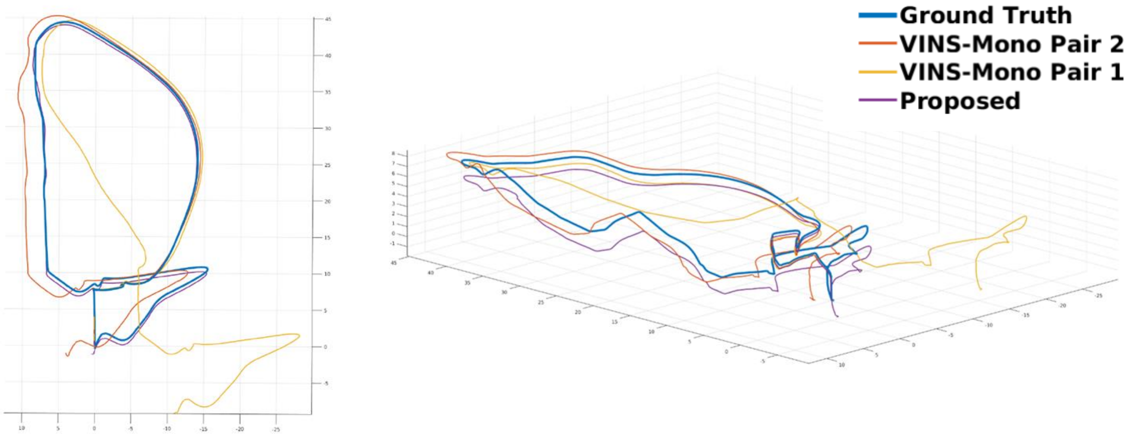


1. Information Sparsification in Visual-Inertial Odometry. Hsiung et al.

Simulation Results

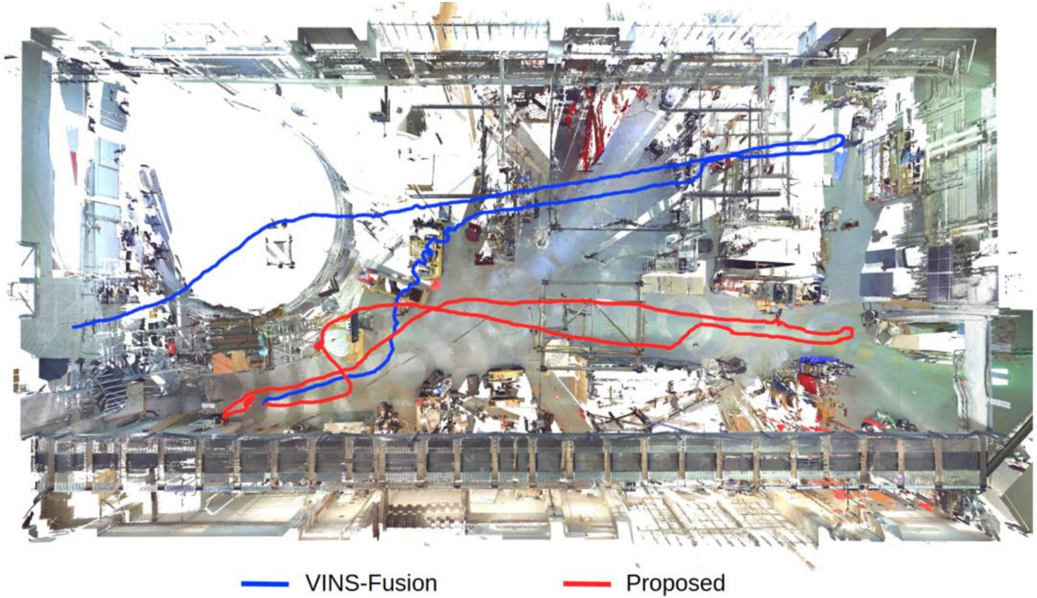
Method	ATE (m)
Proposed	0.700
VINS-Mono[1]	1.561
VINS-Mono[2]	7.048

[1] Forward facing camera
[2] Backward facing camera



Real-world Results

Method	FTE (m)
Proposed	1.77
Proposed[1]	6.980
VINS-Fusion[2]	Failed
VINS-Fusion[3]	Failed



[1] Proposed algorithm without uncertainty compensation
[2] Forward facing stereo pair
[3] Downward facing stereo pair

For any questions feel free to contact:

Joshua Jaekel

jjaekel@andrew.cmu.edu