Techniques for a Failsafe Visual Inertial SLAM System

Ph.D. Defense Presentation

Manohar Kuse <mpkuse@connect.ust.hk>

13th December 2019

香港科大機器人研究所
ROBOTICS INSTITUTE | HKUST

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
電子及計算機工程學系
DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

Talk Outline

Overview of SLAM Systems

Edge Alignment

Place Recognition / Image Retrieval

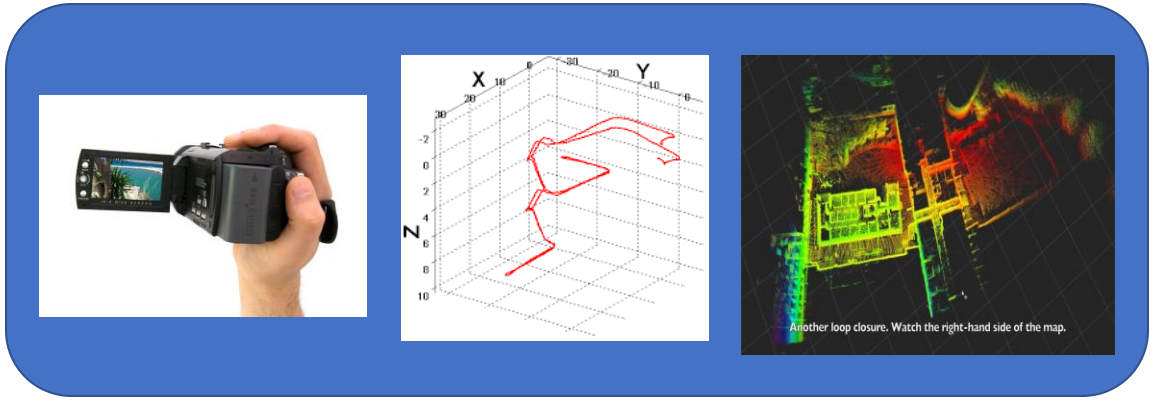
Geometry Computation

Kidnap Aware Pose Graph Solver

Overview of SLAM Systems

Simultaneous Localization and Mapping (SLAM)

- What?
  - Determination of 3D trajectory of movement from Sensors (eg. LIDAR, Cameras etc.)
  - Map of the environment
- Why?



Autonomous Driving

Augmented Reality (AR)

Autonomous Systems like Ground Robots, Drones etc.

Executive Summary:
SLAM Market

- USD 465 million market by 2023 (in 2018 USD 101 million). CAGR 35%
  - Various other firms estimate from US$ 200M to US$ 1B.
- Major growth area : Augmented Reality (AR) / Visualization
- Other substantial industry: Autonomous Robots for various applications
- Limitation of SLAM in **dynamic environments**, and **unpredictability in untested environments** is expected to restrict the market's growth to a certain extent.

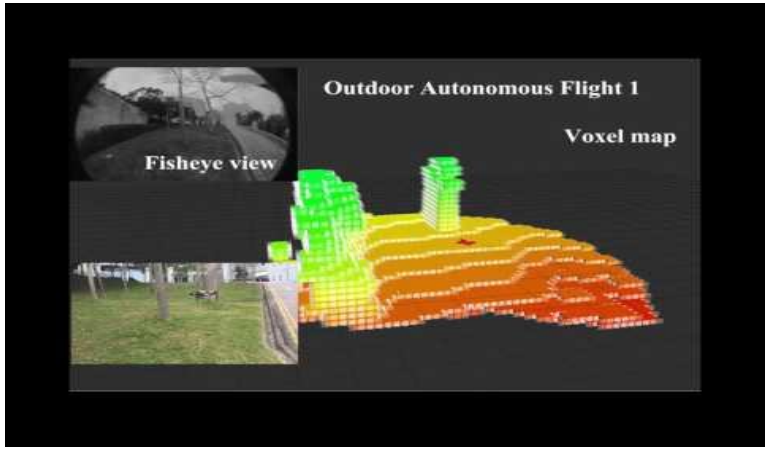
https://www.prnewswire.com/news-releases/global-465-million-simultaneous-localization-and-mapping-slam-market-to-2023-300764202.html

https://www.marketwatch.com/press-release/slam-robots-market-2019-with-top-countries-data-market-size-growth-industry-trends-share-top-key-players-analysis-and-forecast-research-2019-11-22

Realtime SLAM Applications: AR & Autonomous Robots

https://youtu.be/mcH7rtbUzWE?t=17

https://youtu.be/7fKyna9Q4GQ?t=17



https://youtu.be/2zE84HqT0es?t=35

Augmented Reality Toolkit from Tech Companies

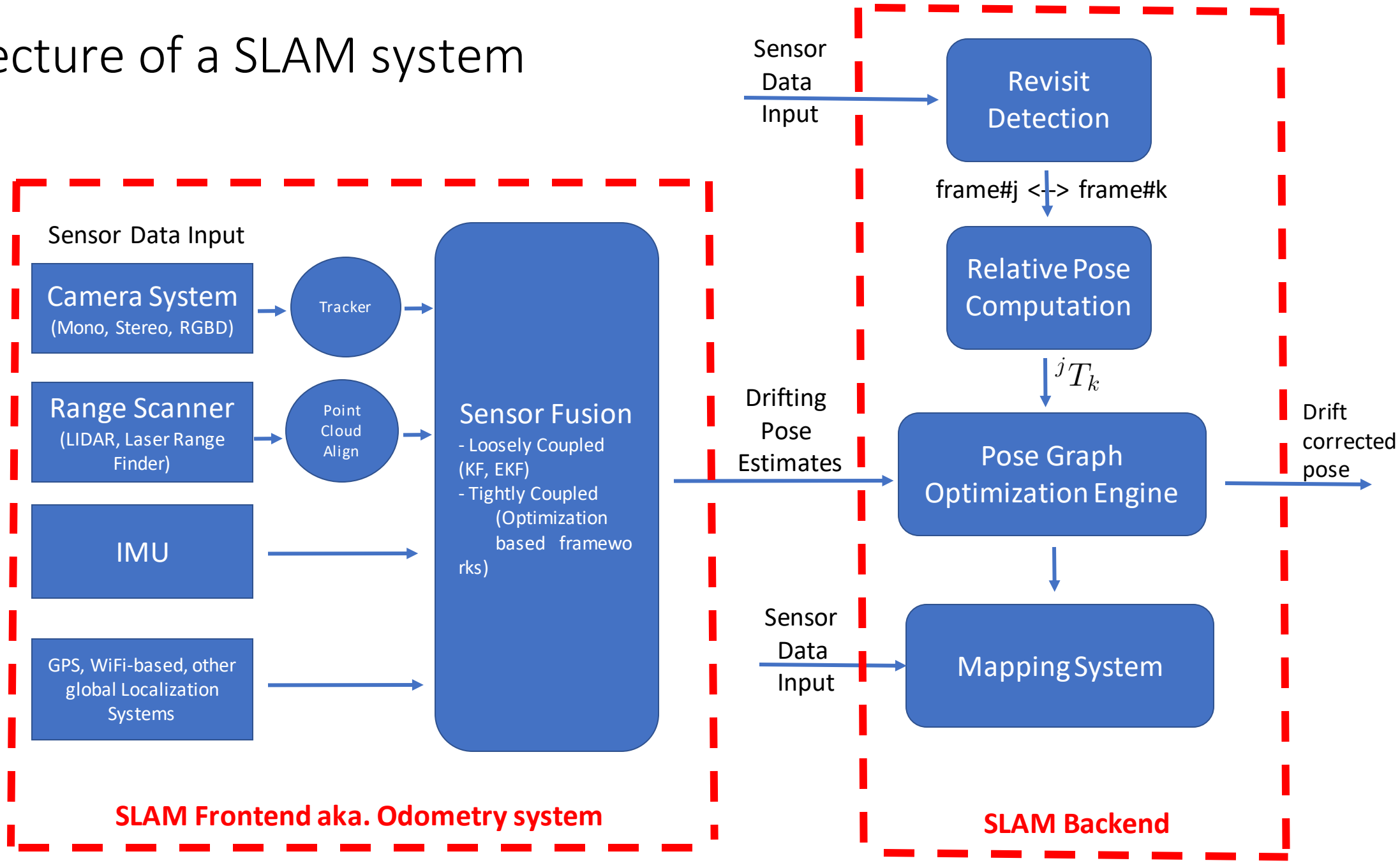
ARCore

oculus

magic leap

https://medium.com/ipg-media-lab/apples-arkit-vs-google-s-arcore-e00ff42b0547
https://medium.com/6d-ai/how-is-arcore-better-than-arkit-5223e6b3e79d

Architecture of a SLAM system



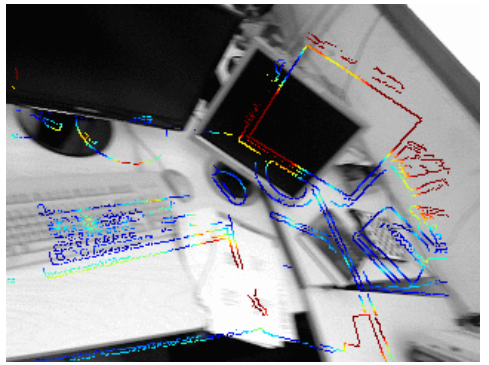
Summary of my Contributions

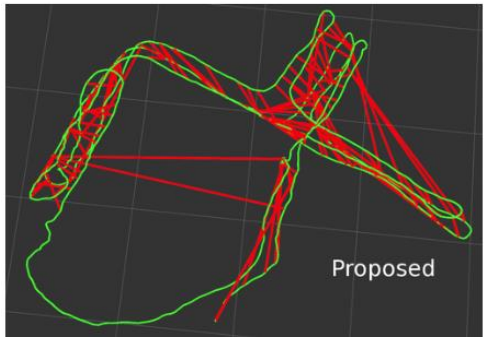
A. Edge Based Visual Odometry System

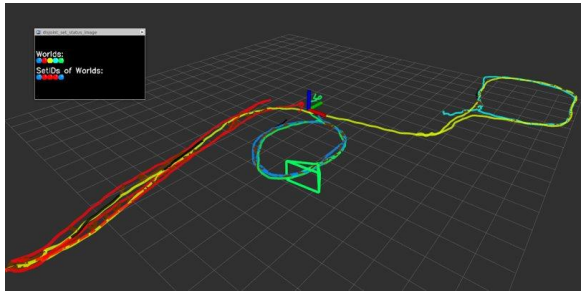
B. CNN Based Place Recognition System

C. Fast and Robust Large Viewpoint Pose Computation

D. Kidnap-aware Pose Graph Solver

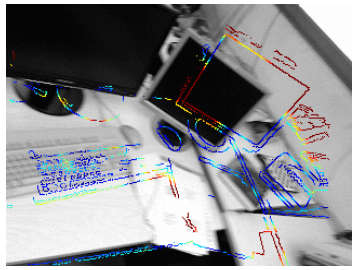






A

Edge Based Visual-Inertial Odometry System



Sensor Data Input

Camera System (Mono, Stereo, RGBD)

Tracker

Range Scanner (LIDAR, Laser Range Finder)

Point Cloud Align

IMU

GPS, WiFi-based, other global Localization Systems

Sensor Fusion
- Loosely Coupled (KF, EKF)
- Tightly Coupled (Optimization based framewo rks)

SLAM Frontend aka. Odometry system

Sensor Data Input

Revisit Detection

frame#j <-> frame#k

Relative Pose Computation

$^{j}T_{k}$

Drifting Pose Estimates

Pose Graph Optimization Engine

Drift corrected pose

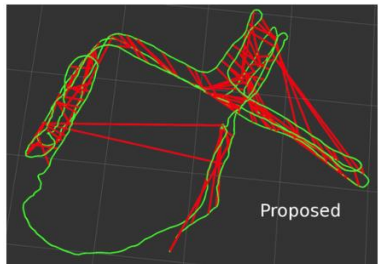
Sensor Data Input

Mapping System

SLAM Backend

B

CNN Based Visual Place Recognition System



Sensor Data Input

Camera System
(Mono, Stereo, RGBD)

Tracker

Range Scanner
(LIDAR, Laser Range Finder)

Point Cloud Align

IMU

GPS, WiFi-based, other global Localization Systems

Sensor Fusion
- Loosely Coupled (KF, EKF)
- Tightly Coupled (Optimization based framewo rks)

SLAM Frontend aka. Odometry system

Sensor Data Input

Revisit Detection

frame#j <--> frame#k

Relative Pose Computation

$^{j}T_{k}$

Drifting Pose Estimates

Pose Graph Optimization Engine

Drift corrected pose

Sensor Data Input

Mapping System

SLAM Backend

C Fast and Robust Large Viewpoint Pose Computation

Sensor Data Input

Camera System (Mono, Stereo, RGBD)

Tracker

Range Scanner (LIDAR, Laser Range Finder)

Point Cloud Align

IMU

GPS, WiFi-based, other global Localization Systems

Sensor Fusion
- Loosely Coupled (KF, EKF)
- Tightly Coupled (Optimization based framewo rks)

SLAM Frontend aka. Odometry system

Drifting Pose Estimates

Sensor Data Input

Revisit Detection

frame#j <-> frame#k

Relative Pose Computation

$^{j}T_{k}$

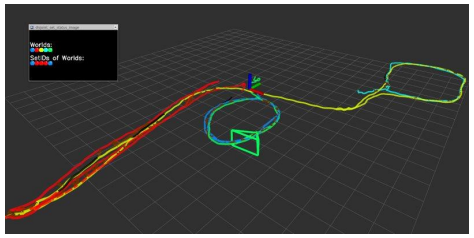
Pose Graph Optimization Engine

Drift corrected pose

Mapping System

SLAM Backend

D Kidnap-aware Pose Graph Solver



Sensor Data Input

Camera System (Mono, Stereo, RGBD)

Tracker

Range Scanner (LIDAR, Laser Range Finder)

Point Cloud Align

IMU

GPS, WiFi-based, other global Localization Systems

Sensor Fusion
- Loosely Coupled (KF, EKF)
- Tightly Coupled (Optimization based framewo rks)

SLAM Frontend aka. Odometry system

Drifting Pose Estimates

Sensor Data Input

Revisit Detection

frame#j <-> frame#k

Relative Pose Computation

$^{j}T_{k}$

Pose Graph Optimization Engine

Drift corrected pose

Mapping System

SLAM Backend

Contributions

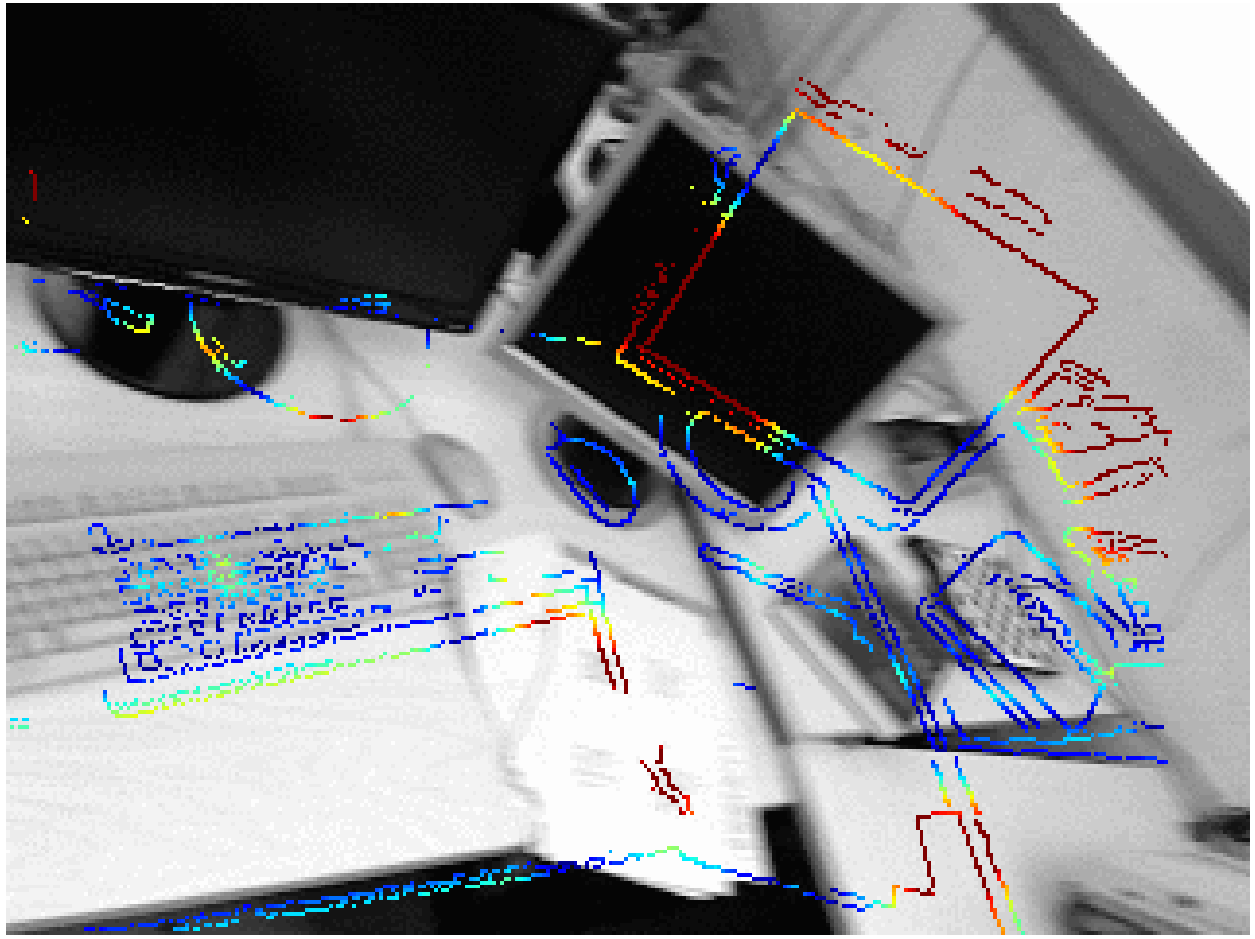
A: Edge Based Visual Odometry System

B: A Neural Net Based Place Recognition System

C: Large Viewpoint Pose Computation

D: Kidnap Aware Pose Graph Solver

Contribution-A:
Edge Based Visual Odometry System



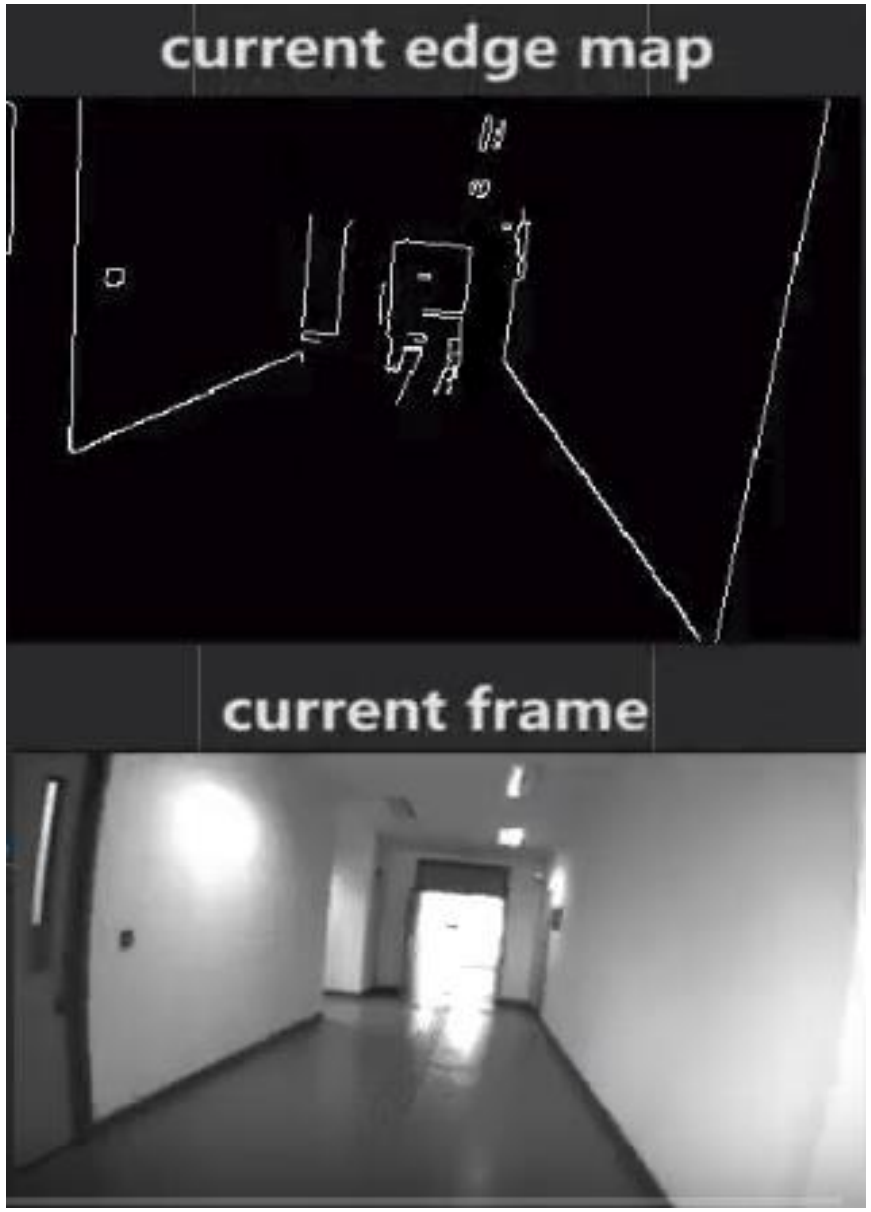
Pose Estimation: Literature Review

- Point Feature based methods
  - 5-point algorithm
  - PnP-like methods
- Direct Methods
  - Photometric Alignment, eg. Kerl *et al.,* Steinbrucker *et al.*, Tykkalaet *et al.*
- ICP-like methods

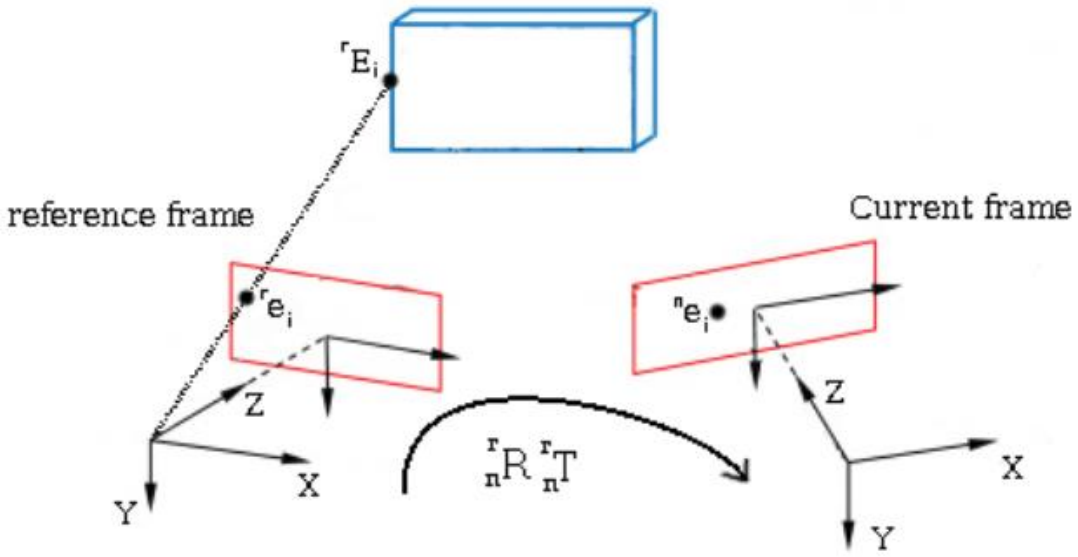
1. C. Kerl, J. Sturm, and D. Cremers, “Robust odometry estimation forRGB-D cameras.,” in ICRA, pp. 3748–3754, IEEE, 2013
2. F. Steinbr ̈ucker, J. Sturm, and D. Cremers, “Real-time visual odometryfrom dense RGB-D images,” in Computer Vision Workshops (ICCVWorkshops), 2011 IEEE International Conference on, pp. 719–722,IEEE, 2011.
3. Tykkalaet , C. Audras, A. Comport,et al., “Direct iterative closestpoint for real-time visual odometry,” in Computer Vision Workshops(ICCV Workshops), 2011 IEEE International Conference on, pp. 2050–2056, IEEE, 2011

Why Edge Align?

- Point-feature based methods
  - Fail in a low texture region (eg corridors)
  - Don't capture correlation between points
- Photometric consistency-based (direct)
  - Small convergence basin
  - Fails in changing lighting conditions.
- Why Edges
  - Plentiful in indoor environments
  - Can be reliably and cheaply detected
  - Invariant to changing lighting conditions



Problem Formulation



$$f(\mathbf{R},\mathbf{T}) = \sum_i \min_j D^2\big(\ \Pi[\mathbf{R}^T({}^r\mathbf{P}_i - \mathbf{T})],\ {}^n\mathbf{u}_j\big).$$

$$\underset{\mathbf{R},\mathbf{T}}{\text{minimize}}\ f(\mathbf{R},\mathbf{T})$$

$$\text{subject to } \mathbf{R} \in SO(3)$$

Distance Transform Trick

$$f(\mathbf{R},\mathbf{T}) = \sum_i \min_j D^2\Big(\Pi[\mathbf{R}^T({}^r\mathbf{P}_i - \mathbf{T})],\ {}^n\mathbf{u}_j \Big).$$

$$\upsilon_{e_i}(\xi) = V^{(n)}\big(\Pi[\tau(\tilde{\Pi}({}^r\mathbf{e}_i, Z_r({}^r\mathbf{e}_i)), \xi)]\big)$$

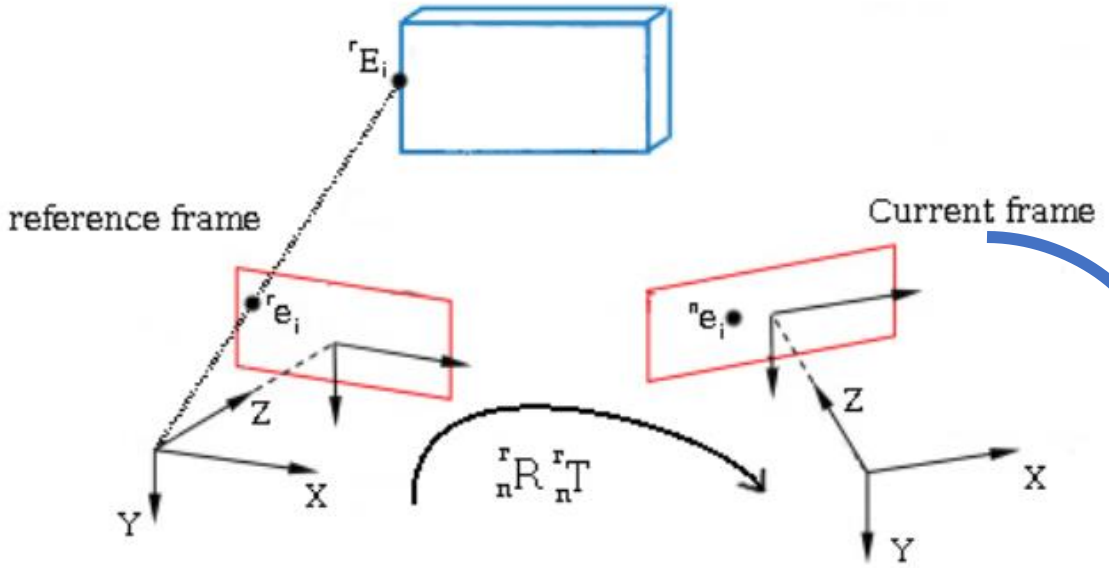
$$f(\xi) = \sum\nolimits_{\forall \mathbf{e}_i} (\upsilon_{e_i}(\xi))^2. \qquad (3)$$

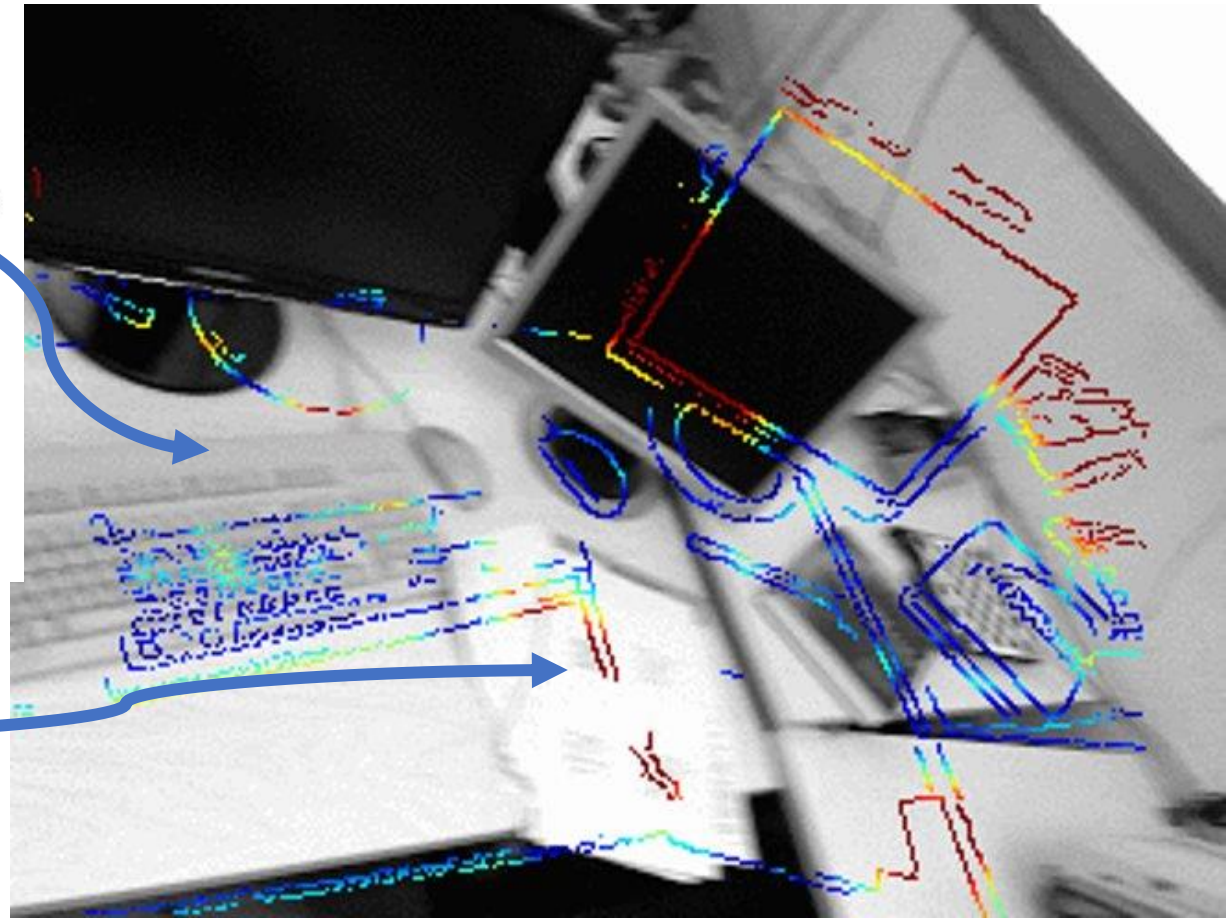
To summarize, the proposed direct edge alignment (D-EA) formulation is given by

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \sum\nolimits_{\forall \mathbf{e}_i} (\upsilon_{e_i}(\xi))^2 \qquad (4)$$

Use **Distance Transform** as a proxy for, minimum distance between projected point and an edge point

Additional Explanation

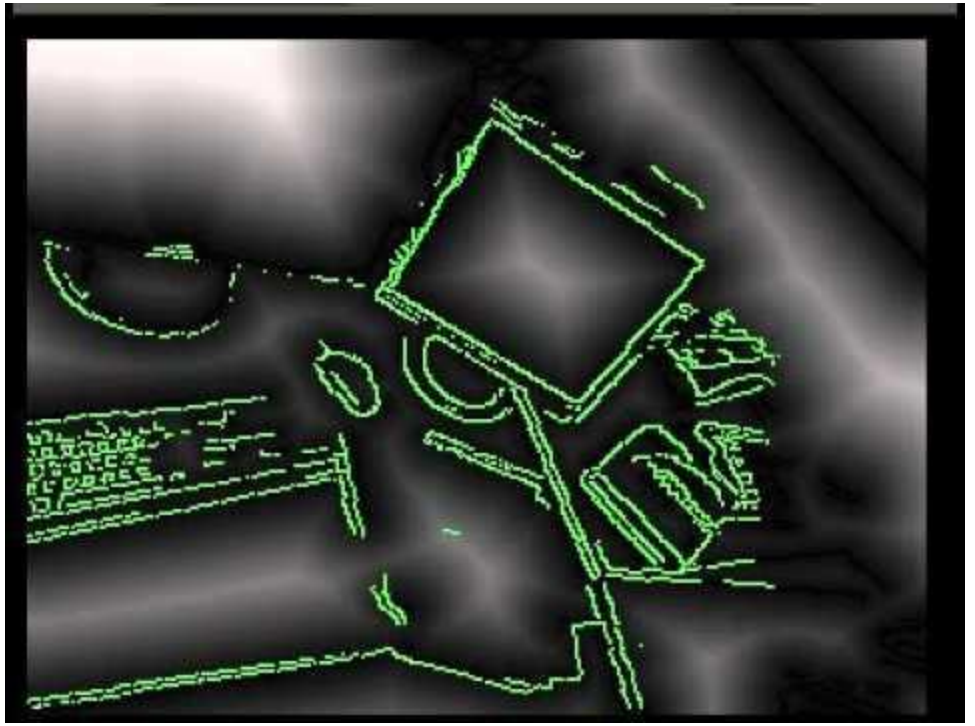




Reprojections of edges of reference frame on current frame. Color coded by distance from the nearest edge

Alignment as iterations progress…

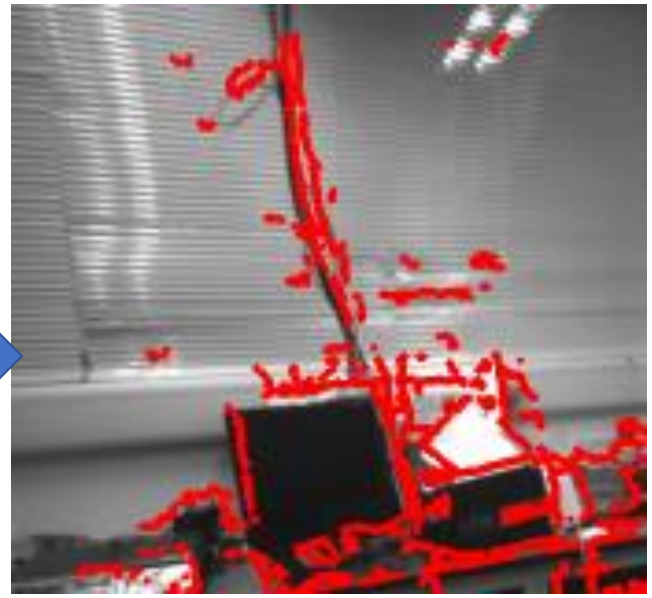
https://www.youtube.com/watch?v=W6IP-tPAm3E

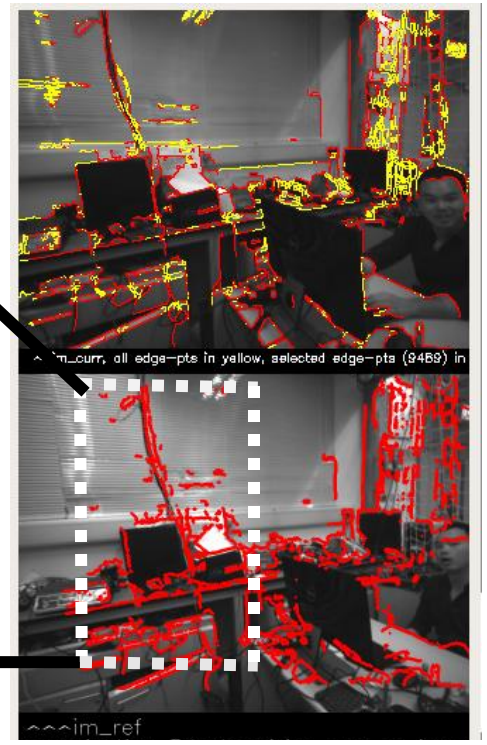


https://www.youtube.com/watch?v=ODOm8adfruc

In this case, only the edge points were shown for visualization.

PnP performed using ORB feature matching





Current Image

Yellow: All Detected Edges

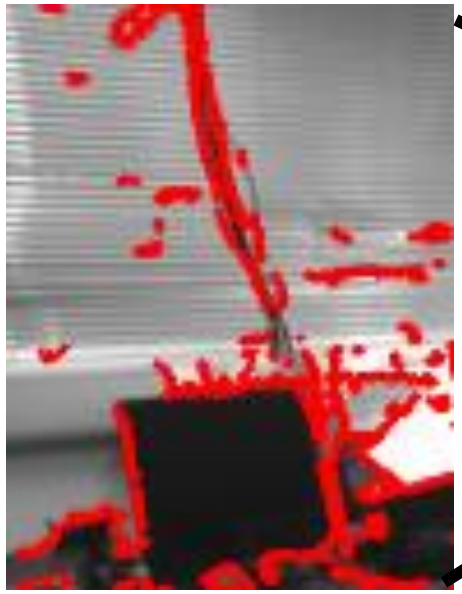
RED: selected edges

$${}^{c}X$$

Alignment with PNP

Edges of im_curr on im_ref

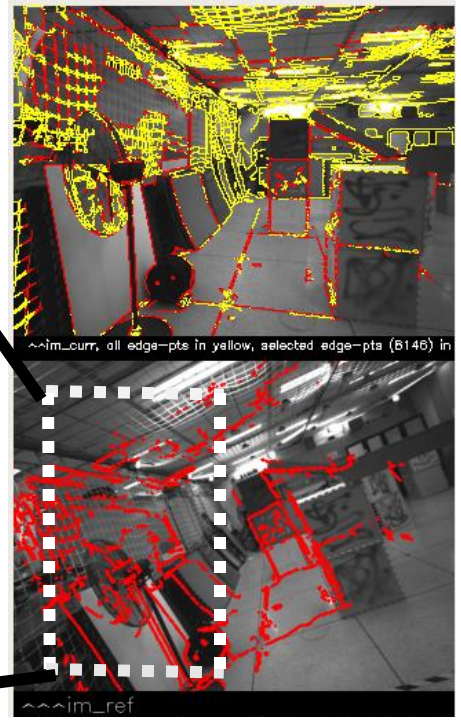
$${}^{r}\hat{T}_{c}(PNP)\ {}^{c}X$$



Edge Alignment (proposed)

Edges of im_curr on im_ref

$${}^{r}\hat{T}_{c}(EA)\ {}^{c}X$$



Current Image

Yellow: All Detected Edges

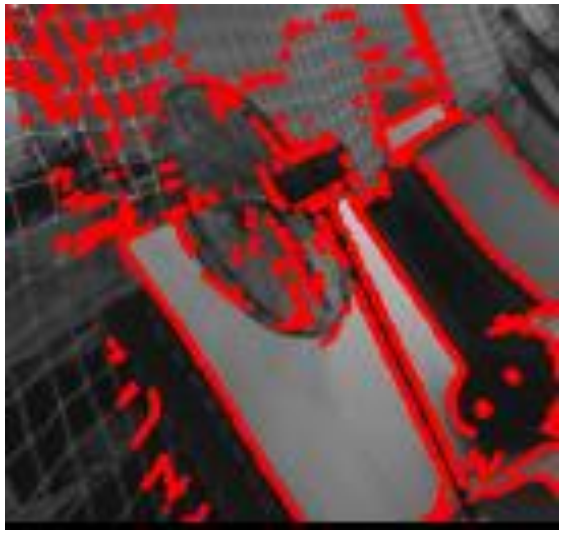
RED: selected edges

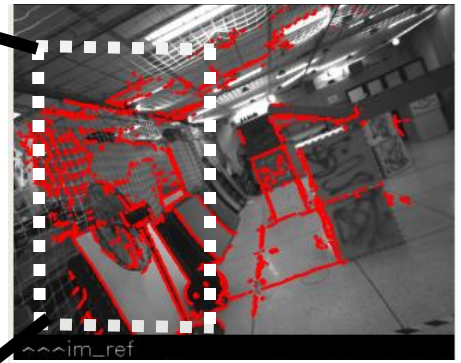
$${}^{c}X$$

Alignment with PNP

Edges of im_curr on im_ref

$${}^{r}\hat{T}_{c}(PNP)\ {}^{c}X$$





Edge Alignment (proposed)

Edges of im_curr on im_ref

$${}^{r}\hat{T}_{c}(EA)\ {}^{c}X$$

Comparison with Direct Alignment

Sequence	D-EA		Kerl *et al.* [2]	
	$\delta = 1$	$\delta = 20$	$\delta = 1$	$\delta = 20$
fr2/desk	0.0324	0.1529	0.0333	0.2217
fr1/desk	0.0289	0.0948	0.0346	0.4286
fr1/desk2	0.0335	0.1818	0.0343	0.3658
fr1/floor	0.0355	0.1988	0.0330	0.3380
fr1/room	0.0353	0.2514	0.0307	0.3399
fr2/desk_with_person	0.0125	0.0594	0.0137	0.1516
fr3/sitting_halfsphere	0.0208	0.1462	0.0181	0.2599
fr2/pioneer_slam2	0.0593	0.4447	0.0847	0.4707

TABLE I: RMSE values of translation component of RPE for various sequences.

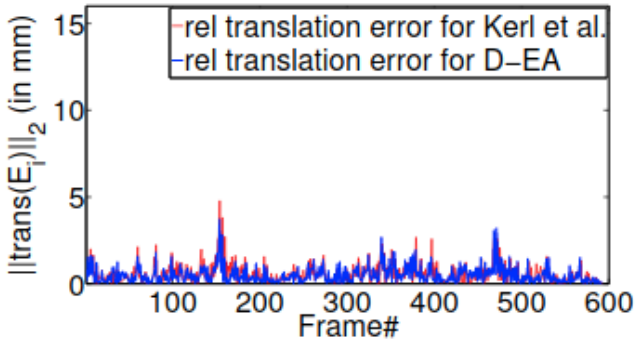
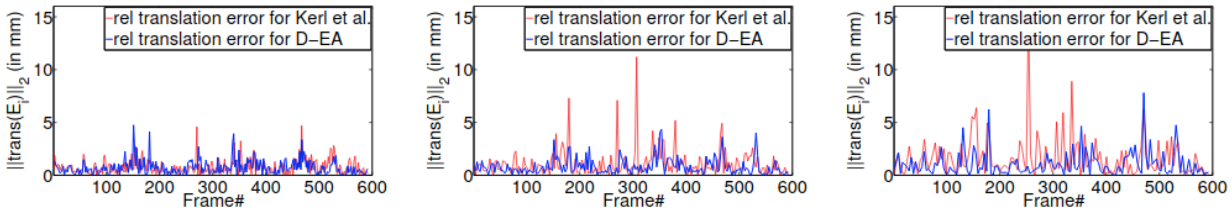


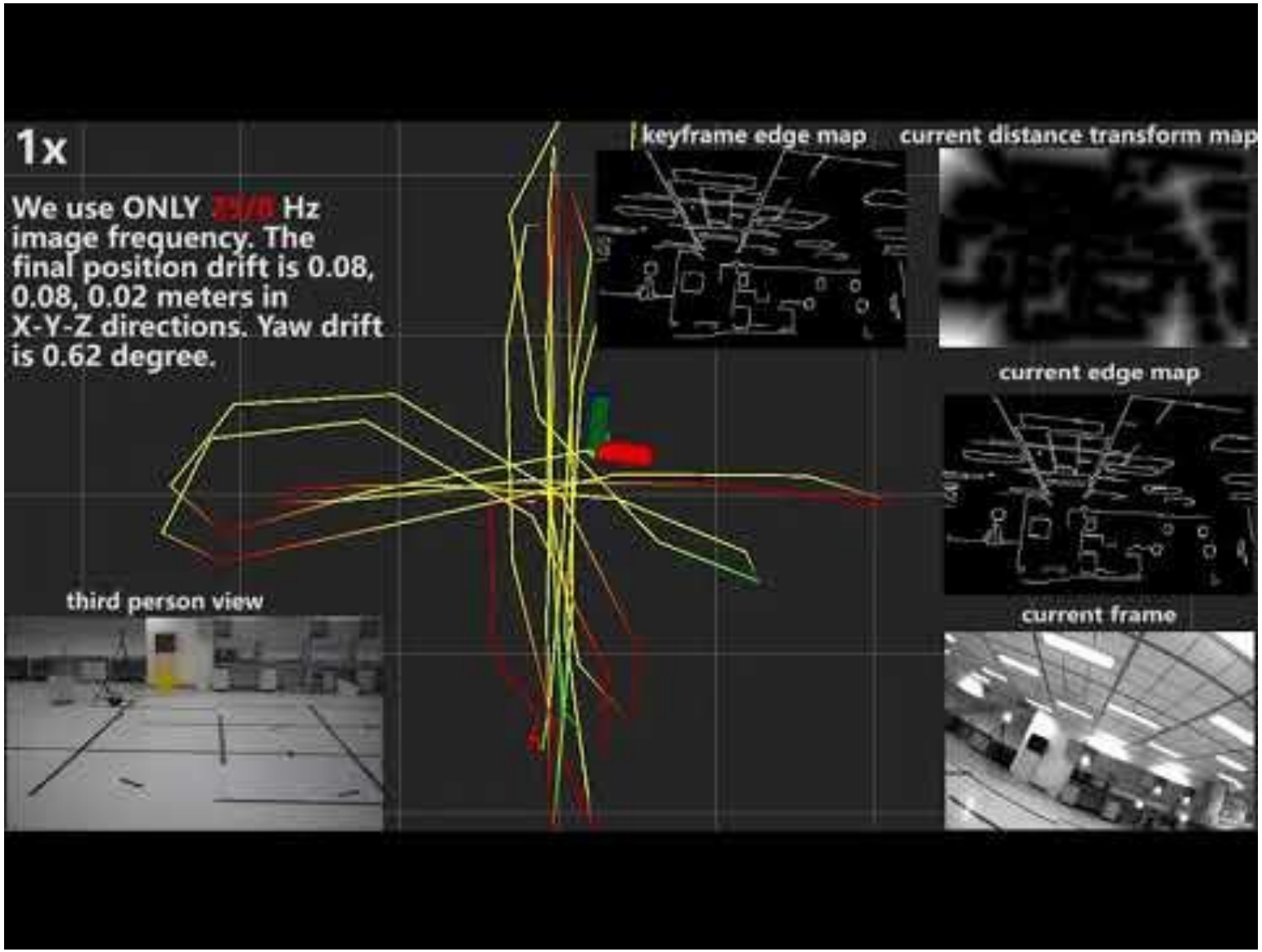
Fig. 4: Translation component of relative pose error at each frame for the sequence 'fr1/desk'. Best viewed in color.



(a) Processing only frames $0, 2, 4, 6 \cdots$ (b) Processing only frames $0, 3, 6, 9 \cdots$ (c) Processing only frames $0, 4, 8, 12 \cdots$

Fig. 5: Robustness for large motions. Relative pose estimation of 'fr1/desk' by skipping frames. Best viewed in color.

Edge Alignment based Visual-Inertial Odometry System



https://youtu.be/Pctn3jrBk4w

Conclusion from Edge Alignment

- Provides a robust way for pose estimation
- Larger convergence radius
- Comparable accuracy to direct methods
- Generally more accurate than sparse-feature based methods

Contributions

A: Edge Based Visual Odometry System

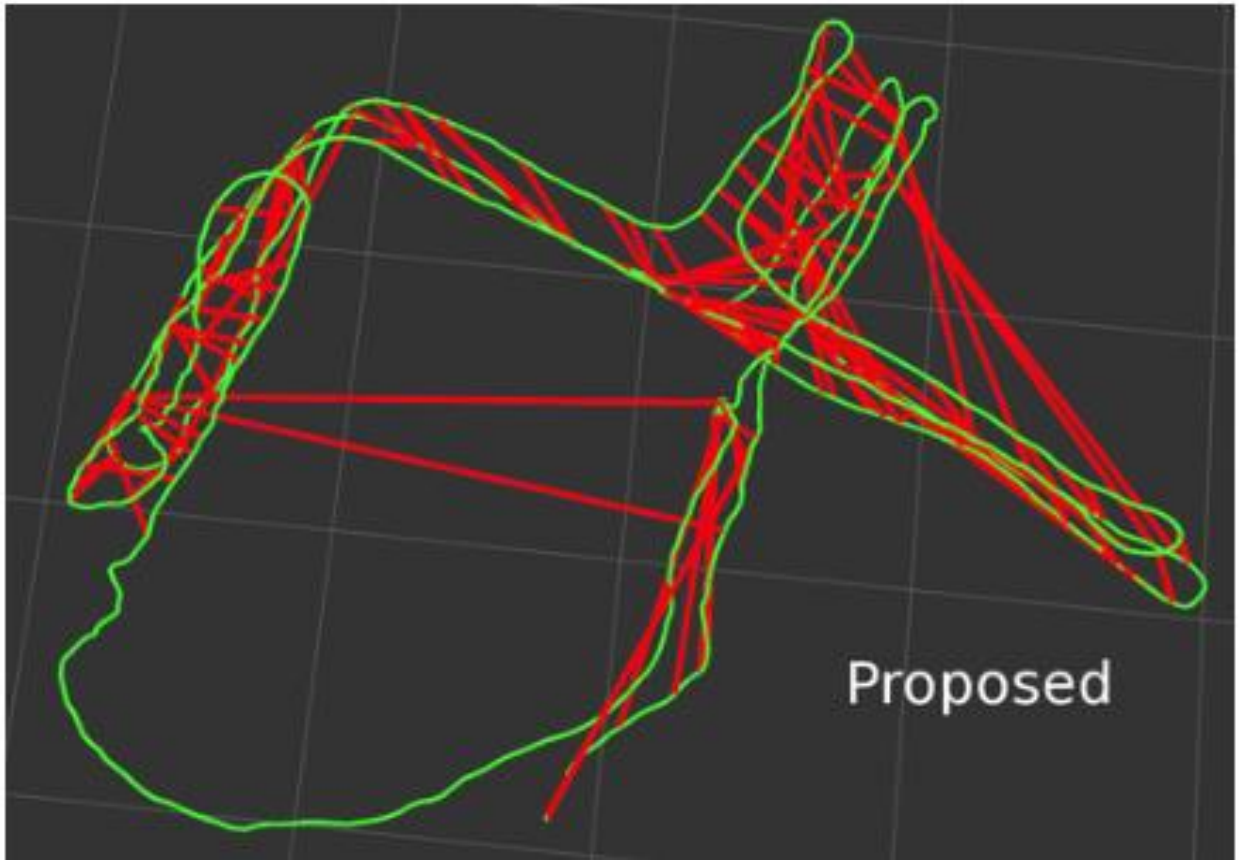
B: A Neural Net Based Place Recognition System

C: Large Viewpoint Pose Computation

D: Kidnap Aware Pose Graph Solver

Contribution-B:

A Neural Net Based Place Recognition System



Literature Review of Loop Detection Methods

Representation	Retrieval	Method Description
SPF-real	BOW	[19, 20] soft-real-time (can run @5-10hz)
SPF-binary	BOW	[7, 22, 43] real-time (10hz or more)
SPF-real	Inc-BOW	[1, 36, 44] soft-real-time (1-5 Hz). [8, 64] make use of temporal information to form visual words scene representation.
SPF-binary	Inc-BOW	[24, 25, 35, 66] soft-real-time to 1-5Hz processing
SPF	graph	[55]
Pretrained-CNN	NN	[4, 6, 58] provide for real-time descriptor computation (10-15hz). dimensionality reduction accomplished at 5hz, NN with 64K dim is really slow, NN after dimemsionality reduction (4000d) is about 5-15 hz.
Pretrained-CNN	BOW	[14, 15, 29]
Custom-CNN	NN	[2, 13, 39, 63] provides for real-time descriptor computation. dim-reduction and NN search are bottle necks.
Custom-CNN with region-proposals	regionwise-NN	[59] very slow representation vector computation. [34] region descriptor encoding computation 2-3 Hz. Reported matching times is several seconds.
Unsupervised Learning	NN	[23, 38, 41] descriptors are not descriptive enough after dim-reduction. real-time desc computation.
Intensity agnostic	NN optimization	[42, 45] [27, 37, 67] generally slow.

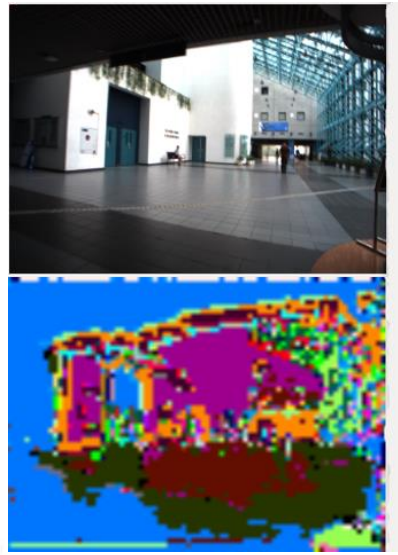
SPF = Sparse Point Features
BOW = Bag-of-words
NN = Nearest Neighbours

Table borrowed from our paper : https://arxiv.org/pdf/1904.06962.pdf

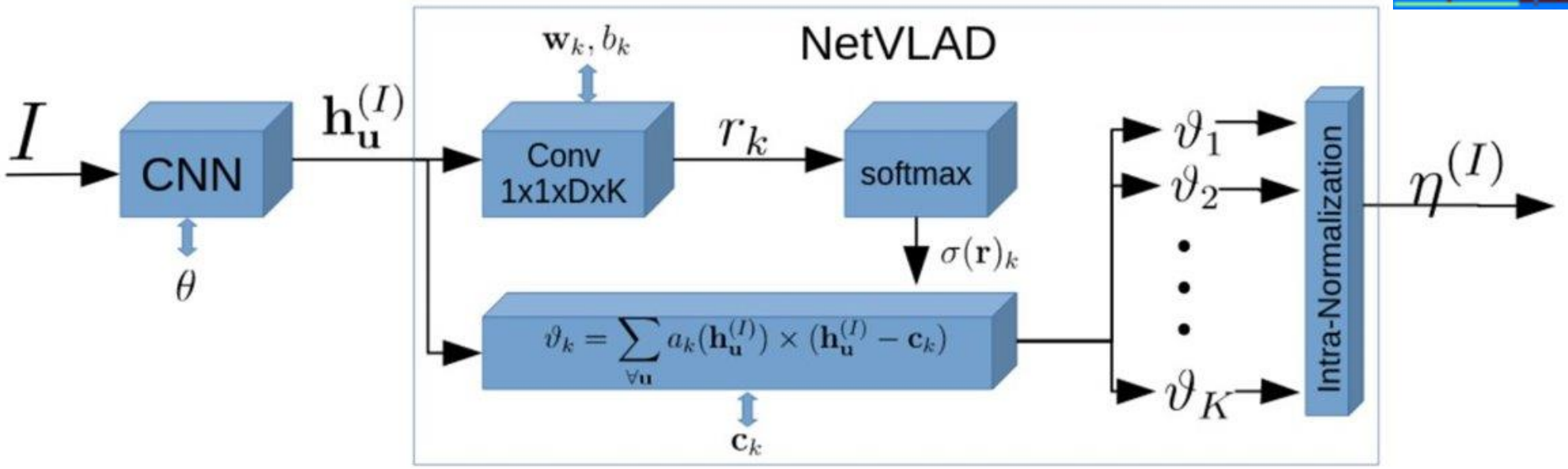
Revisit Detection

- Revisit detection in the wild is hard
  - Issue with similar looking place (eg. Hang Hau station and Choi Hung station)
  - On the stairs (G/F and 3/F looks the same)
  - Long corridors (Perceptual Aliasing)
  - Ambient lighting (day scene vs. Night, hot summer vs snowy winter)
- Loop detection in real-time SLAM system only focus on frontal scenes
- A revisit detection method for real-time SLAM system need to be fast and robust (a balancing act)

Method



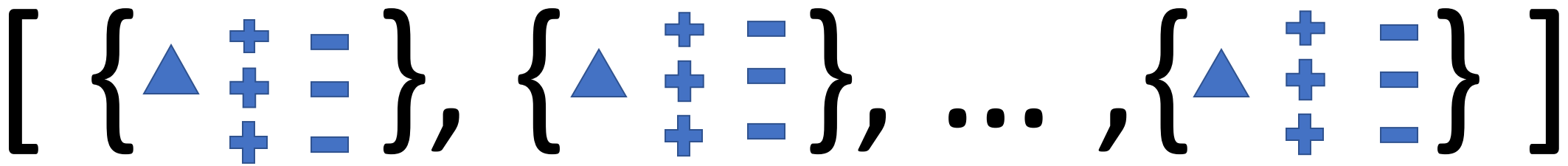
- We propose an CNN based Image Descriptor method



[NetVLAD] : Arandjelovic, Relja, et al. "NetVLAD: CNN architecture for weakly supervised place recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. CVPR 2016.

Training: Weakly Supervised

- Google Street View Data (Pittsburg Dataset)
- We use a batch size of 24,
- 1 query image, 9 +ve samples, 9 –ve samples per batch



Terminology

- Query Image

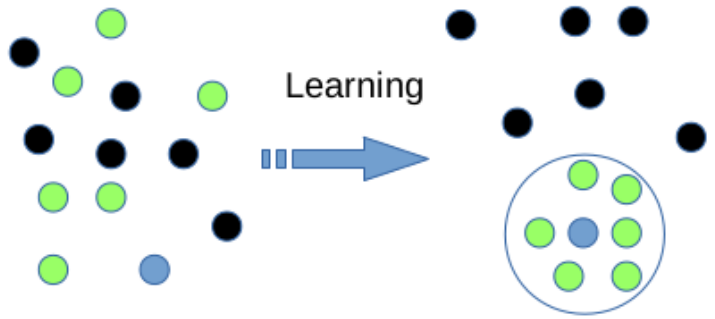
- Positive Sample

- Negative Sample

The Loss Function: Motivation for Design

$$\langle \eta^{(I_q)}, \eta^{(N_j)} \rangle > \langle \eta^{(I_q)}, \eta^{(P_i)} \rangle; \qquad \langle \eta^{(I_q)}, \eta^{(N_j)} \rangle < \langle \eta^{(I_q)}, \eta^{(P_i)} \rangle.$$

- We do **NOT desire** this property for the image descriptors, so penalize this case
- This is the **desired** property for the image descriptor, so have a zero loss



η is the image descriptor, a 1000-Dim float vector

Loss Function / Semi Supervised Learning

Triplet loss, used by NetVLAD's authors

used triplet loss function can be rewritten in our notations as, $L_{triplet-loss}$:

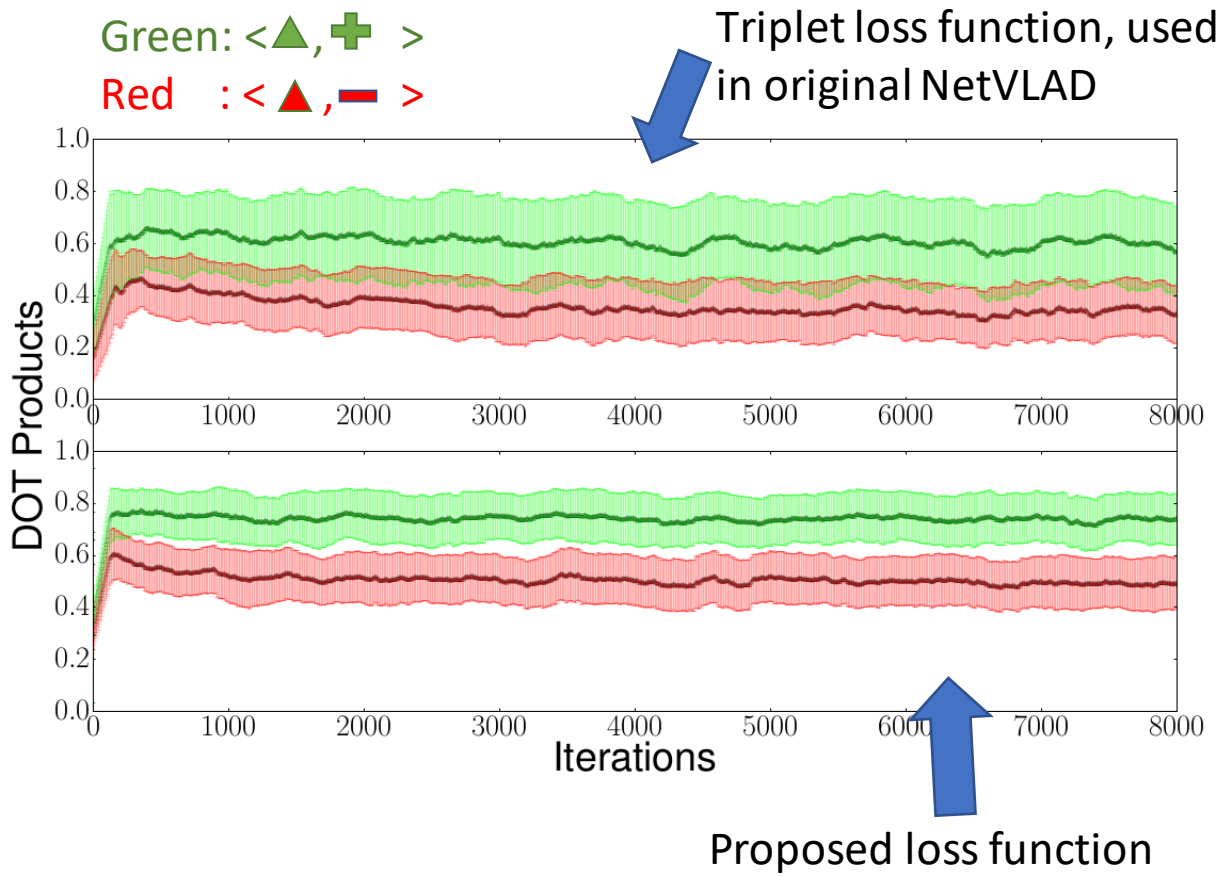
$$\sum_j \max\big(0, \langle \eta^{(I_q)}, \eta^{(N_j)} \rangle - \min_i(\langle \eta^{(I_q)}, \eta^{(P_i)} \rangle) + \epsilon\big)$$

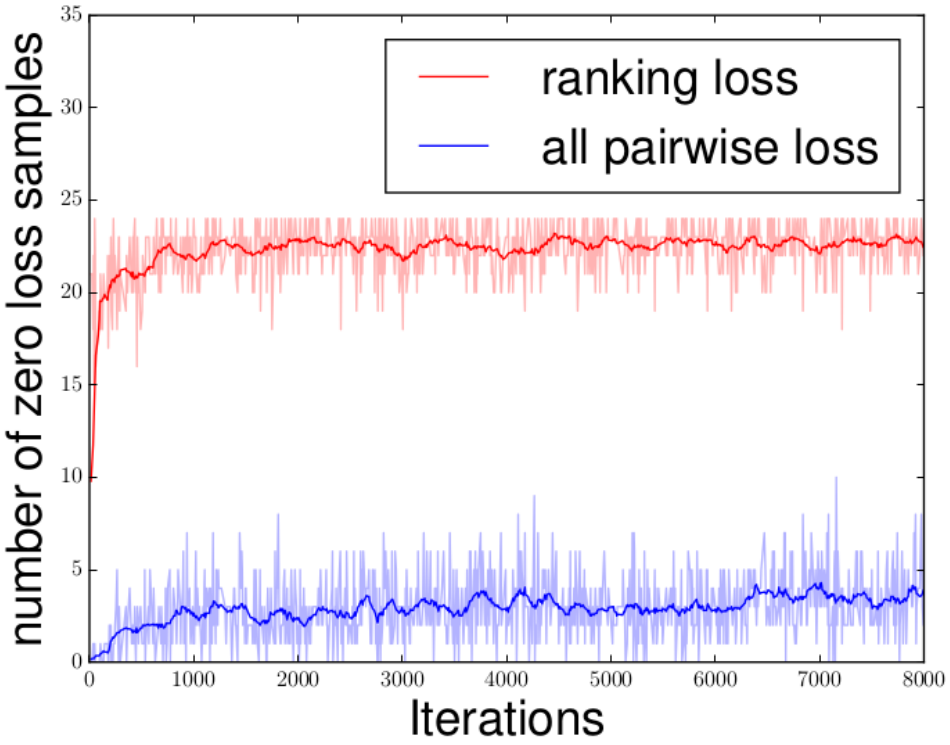
I use this because, mine is smoother resulting in stable learning, more separability in +ve and –ve sets

$\{I_q, \{P_i\}_{i=1,\ldots,m}, \{N_j\}_{j=1,\ldots,n}\}$) is given as, $L_{proposed}$:

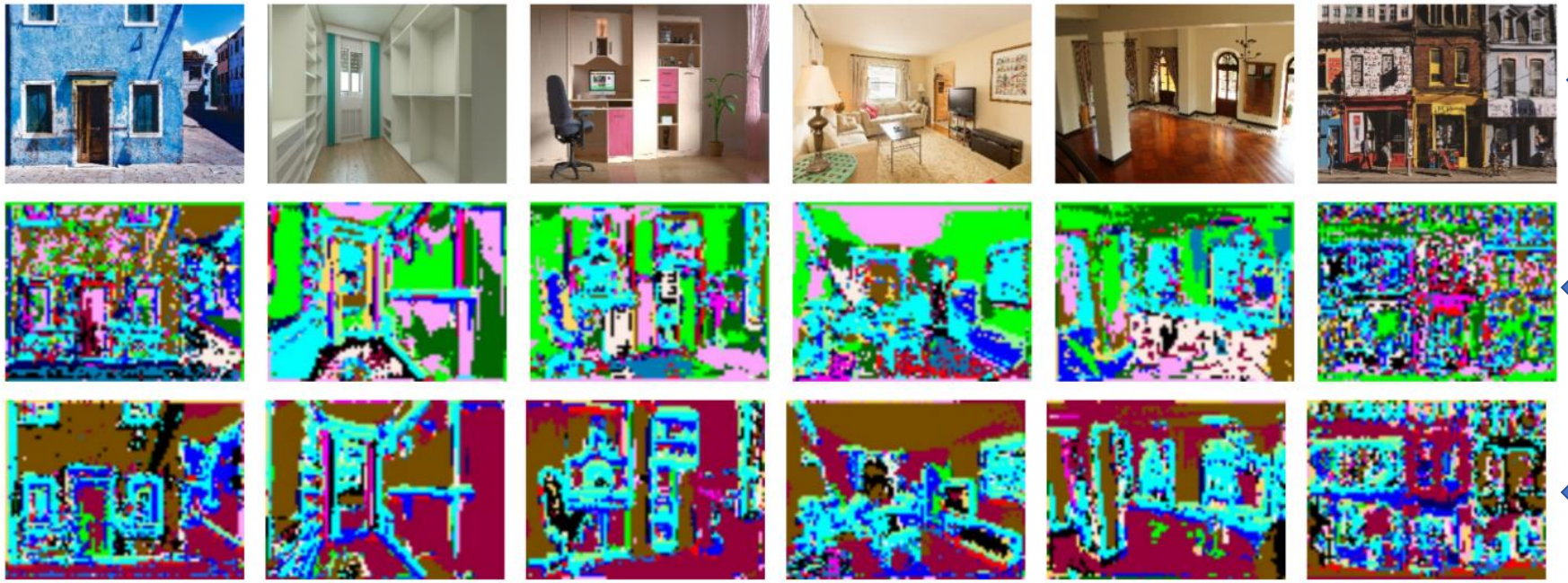
$$L = \sum_{i=1}^{m} \sum_{j=1}^{n} \max(0, \langle \eta^{(I_q)}, \eta^{(N_j)} \rangle - \langle \eta^{(I_q)}, \eta^{(P_i)} \rangle + \epsilon)$$

Effect of Proposed Loss





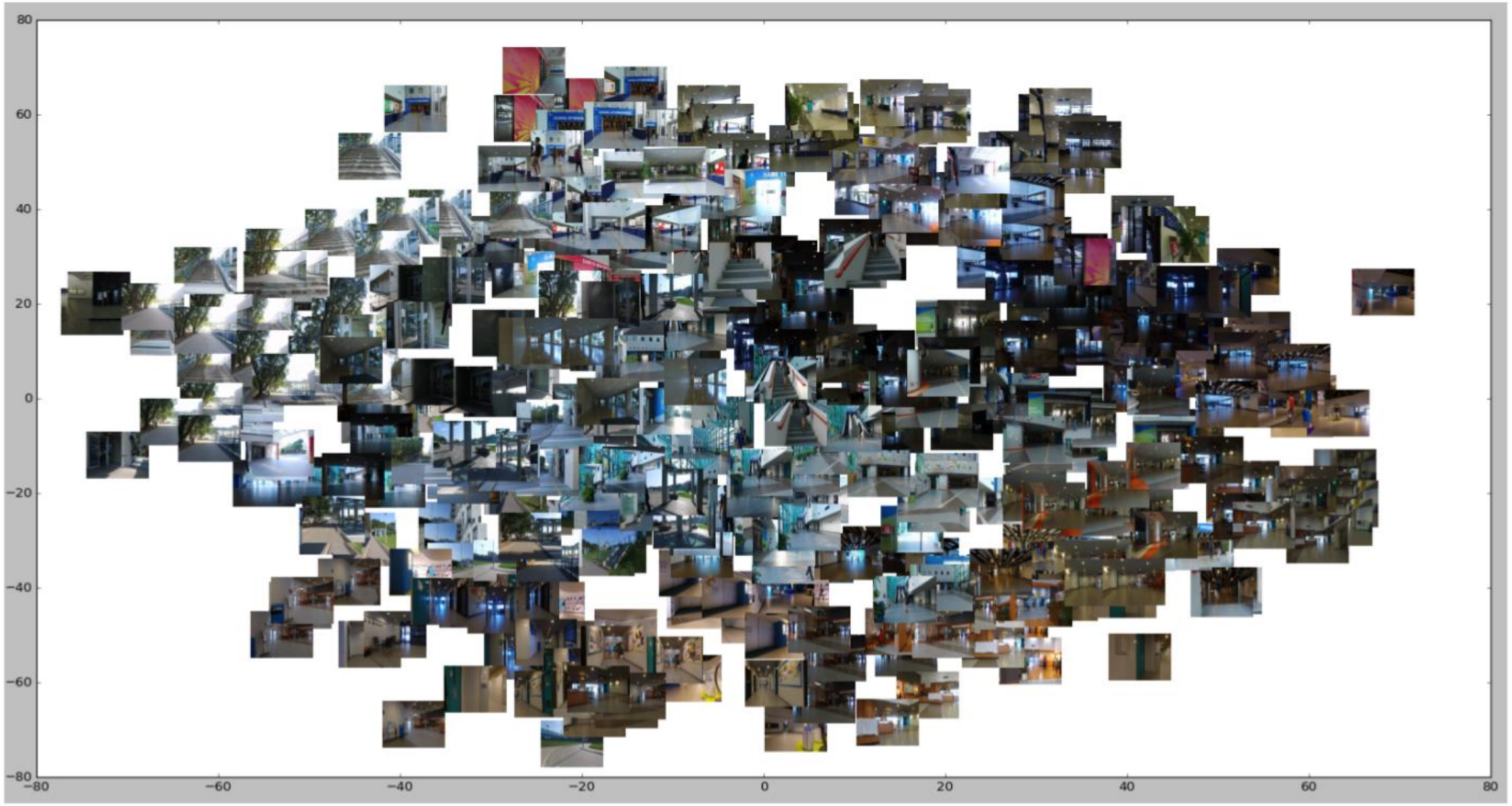
Comparison of Association Maps



Input Images

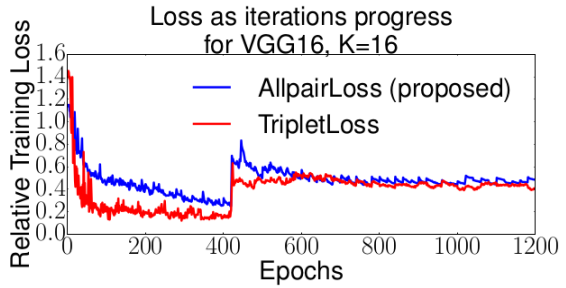
Association maps
Network trained with TripletLoss

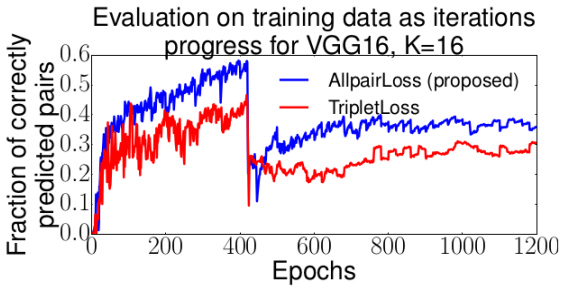
Network trained with Proposed AllPairLoss

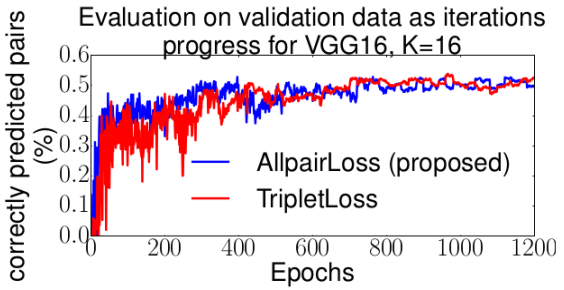


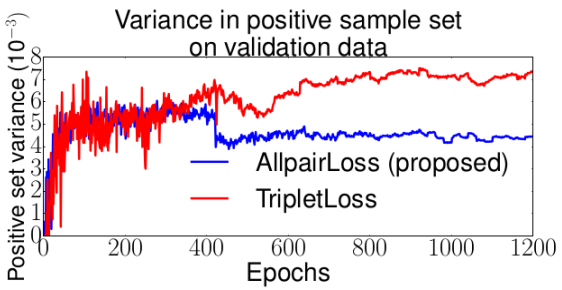
Visualizing HKUST with T-SNE

VGG vs Separable Convolutions (Mobilenets)





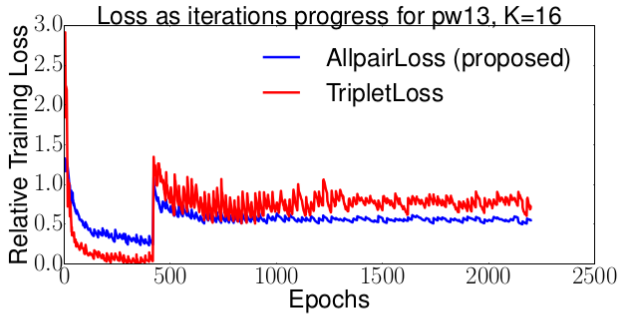


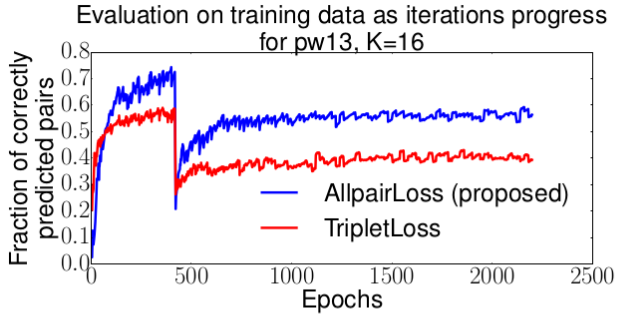


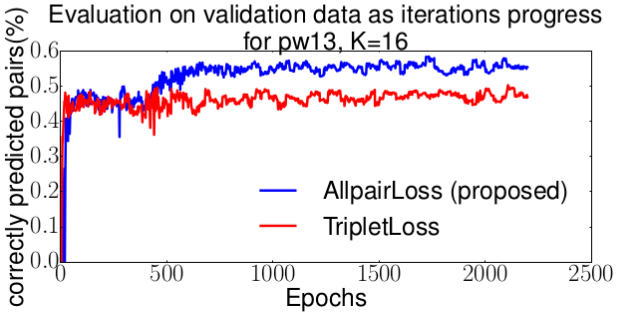
Conclusion:

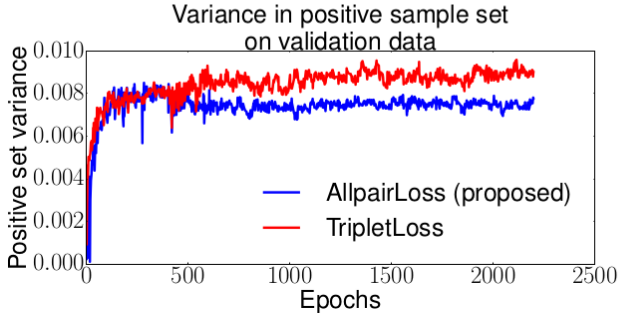
- Faster learning when using proposed loss function
- Lower variance in positive sample sets with proposed loss function

VGG vs Separable Convolutions (Mobilenets)









Standard Datasets – Mappilary

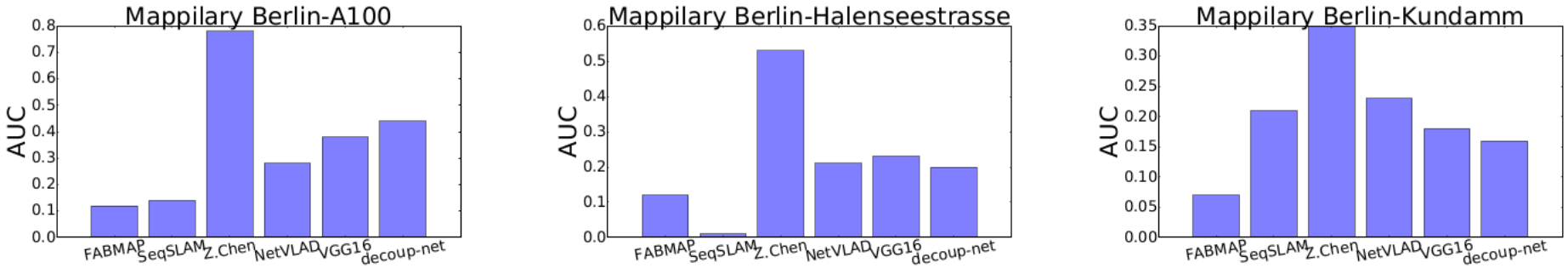
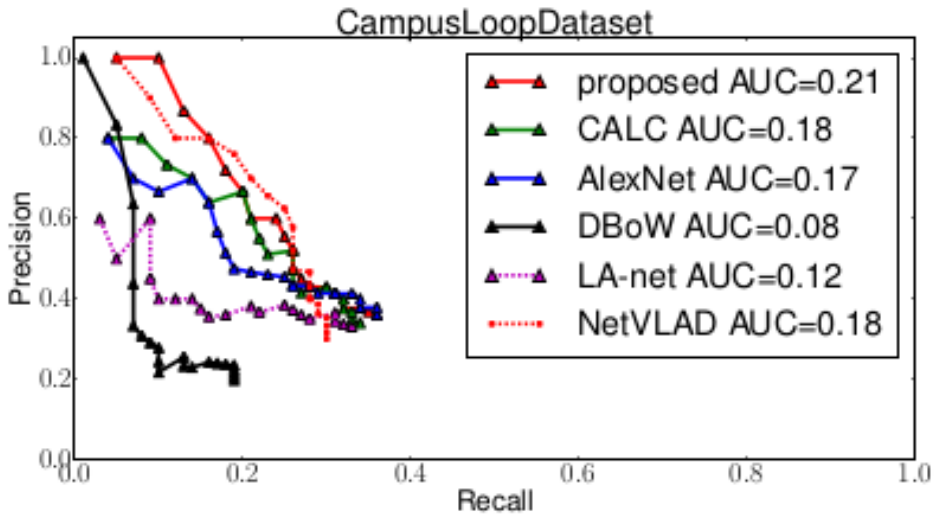
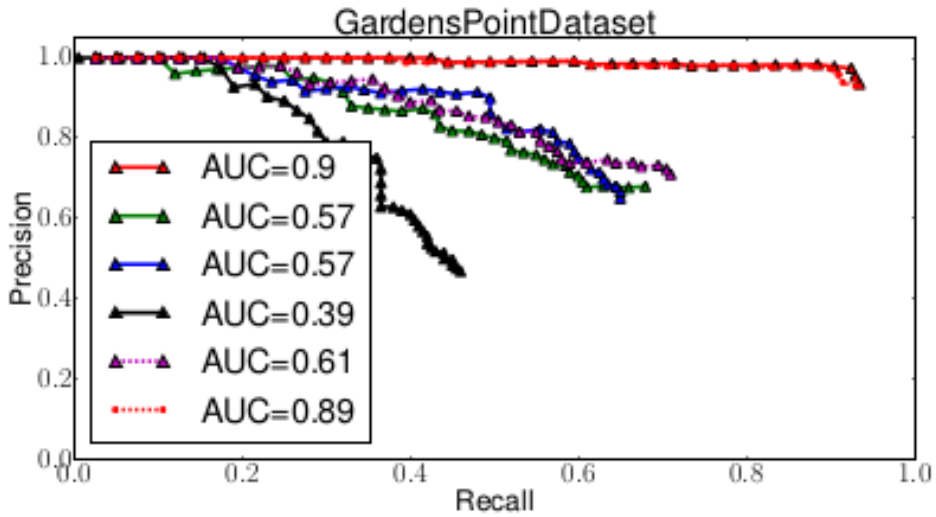


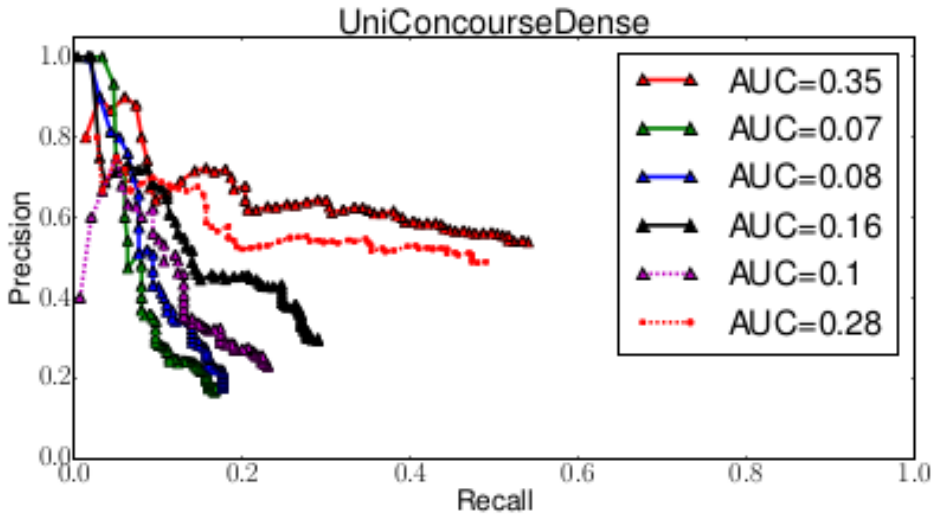
Figure 4.10: Comparing the methods with area under the curve (AUC) of the precision-recall plots for the mappilary dataset. The following methods were compared: FABMAP [41], SeqSLAM [138], Z.Chen [35], NetVLAD [5], proposed with VGG16 backend net, proposed with decoupled net as backend net.

- Slower methods are generally better at the retrieval tasks
- When comparing with common loop detection methods (viz. FABMAP, DBOW, SeqSLAM) proposed method outperforms

Standard Datasets – CampusLoop Dataset







Runtime Performance: NetVLAD vs Proposed

Learnable Parameters

Image Descriptor Dimension

CNN Layer	# L	D-Size	Model (MB)	Fwd pass memory (MB)		GFLOPS	
VGG16_K16				**320x240**	**640x480**	**320x240**	**640x480**
block5_pool	14.7M	8192	56.19	234.94	767.56	47.04	188.08
block4_pool	7.6M	8192	29.19	174.06	725.32	47.05	188.117
block3_pool	1.7M	4096	6.65	165.47	641.86	47.05	188.167
VGG16_K64				**320x240**	**640x480**	**320x240**	**640x480**
block5_pool	14.78M	32768	56.38	234.32	711.46	47.05	188.11
block4_pool	7.70M	32768	29.38	203.53	696.23	47.08	188.26
block3_pool	1.76M	16384	6.75	158.86	635.26	47.13	188.46
decoup_K16				**320x240**	**640x480**	**320x240**	**640x480**
pw13	3.2M	8192	12	197.97	792.21	1.742	7.01
pw10	1.36M	8192	5	189.1	734.48	1.749	7.03
pw7	554K	4096	2	164.9	652.25	1.749	7.04
decoup_K16_r				**320x240**	**640x480**	**320x240**	**640x480**
pw13	3.5M	512	12	211.58	793.46	1.742	7.01
pw10	1.49M	512	5	193.86	739.73	1.749	7.03
pw7	686K	512	2	167.67	657.97	1.749	7.04
decoup_K64				**320x240**	**640x480**	**320x240**	**640x480**
pw13	3.33M	32768	12	210.98	805.21	1.76	7.08
pw10	1.40M	32768	5	189.38	734.58	1.78	7.18
pw7	600K	16384	2	162.86	652.35	1.78	7.186

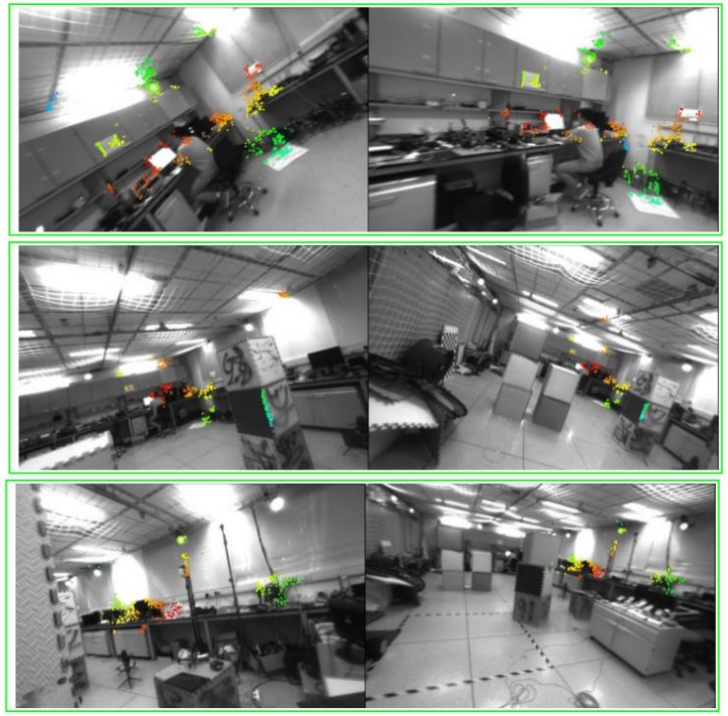
Running Time Performance

- 1000-dim vectors vs Original NetVLAD 64K-dim reduced to 4K by matrix-multiple (Random Gaussian Matrix)
  - 640x480 can compute in 12-15ms on TitanX.
  - 15-20ms on TX2 with TensorRT
  - 40-50ms in TX2 with TensorFlow
  - Original NetVLAD takes about 1sec for 640x480 image
- Memory consumption is 40kb/s (assuming images are processed at 10hz). About 6hrs will need 1GB.
  - Original NetVLAD gives 64K or 4K descriptors. Depending on which you use consumption will be from 4X to 64X.
- Nearest Neighbor Query
  - Bruteforce: 30-50ms for 10000 keyframes (or 30min of walking).
  - Locality Sensitive Hashing (FAISS): 10-20ms for 10K keyframes. Scales better.

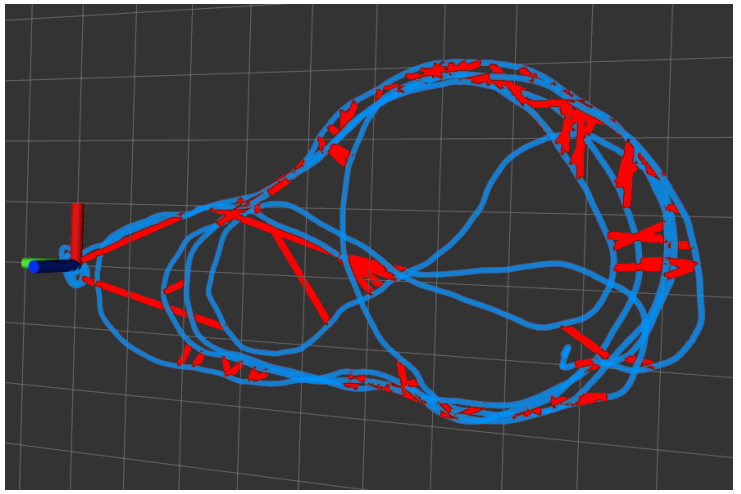
Loop Detection Module of VINS-Fusion vs Proposed

Conclusion:

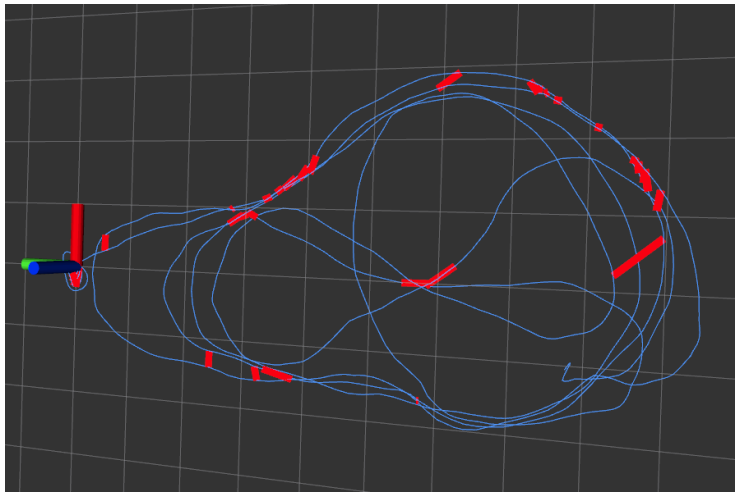
Proposed method has a higher recall rate



Example pairs not detected in the VINS-Fusion system but detected in the proposed method



Loop Detection using the proposed method



Loop Detection in VINS-Fusion (Bag-of-words based)

[VINS-Fusion] https://github.com/HKUST-Aerial-Robotics/VINS-Fusion

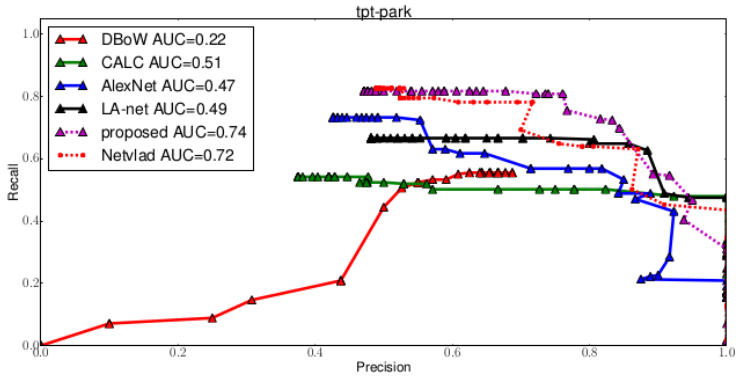
Precision-Recall Analysis on Real-world SLAM Dataset

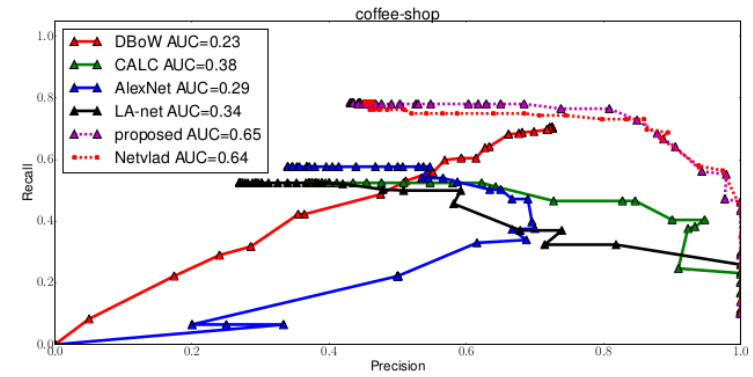
- Tpt-park
- Coffee-shop
- Seng-base
- Lsk-1
- Base-2

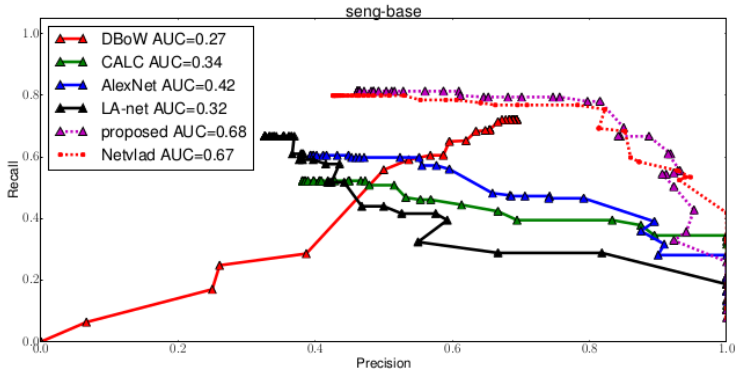
General Conclusion:

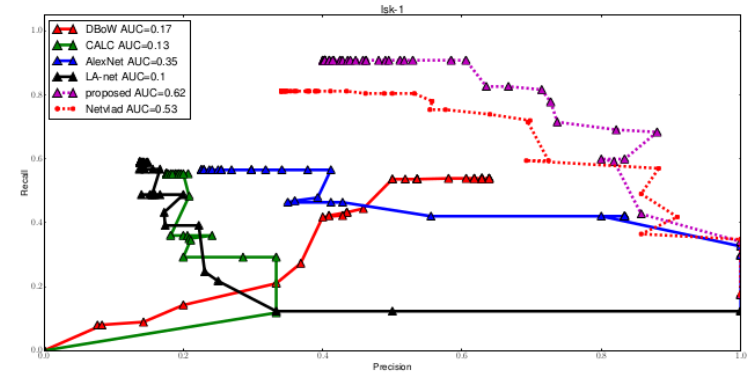
NetVLAD with VGG and proposed give comparable performance on realworld SLAM data.

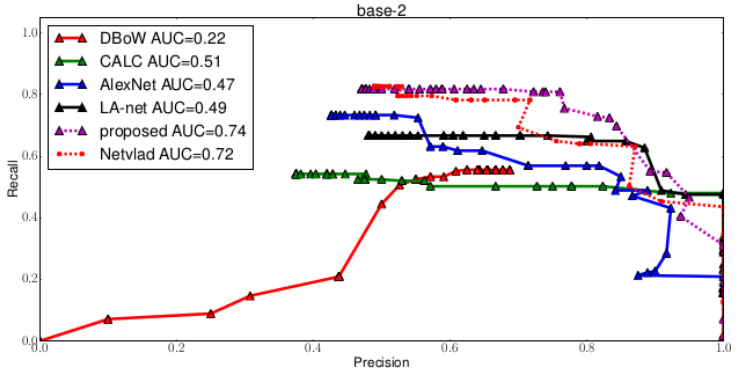
Compared to the bag-of-words and other deep-learning based method proposed method outperforms.



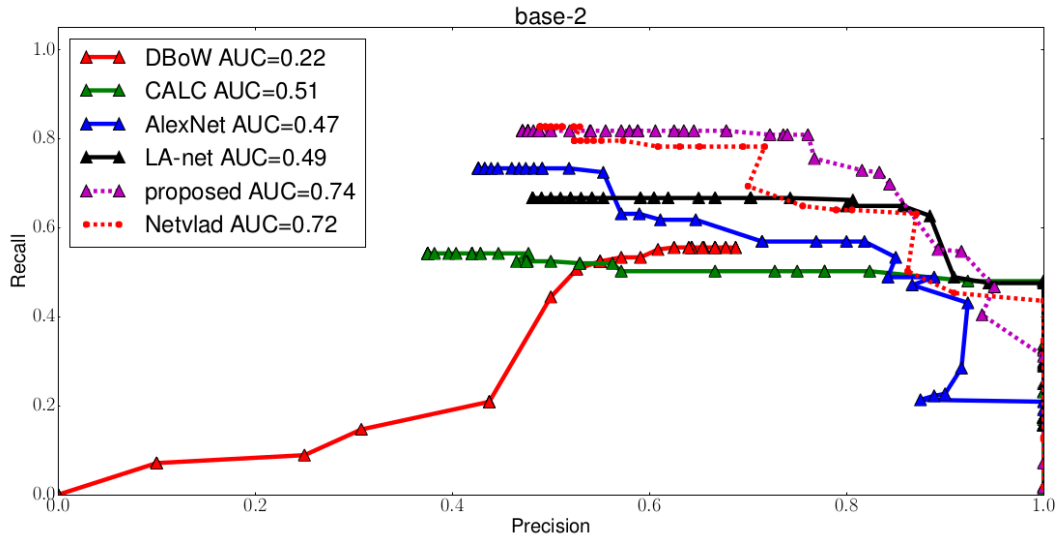


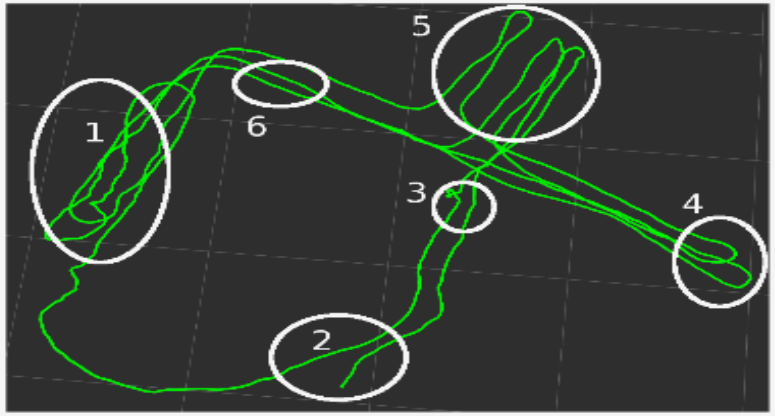


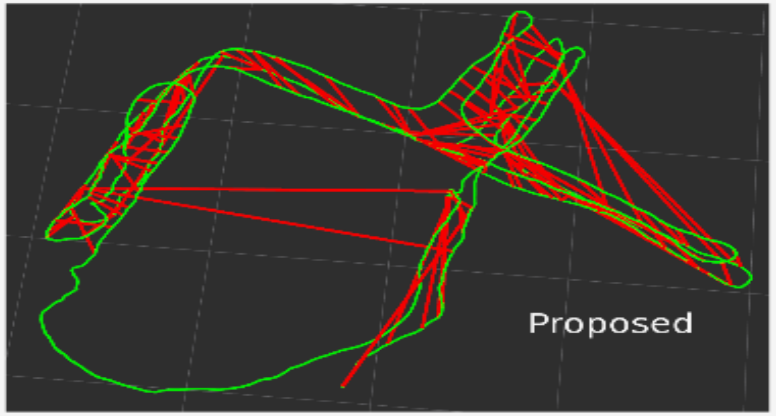


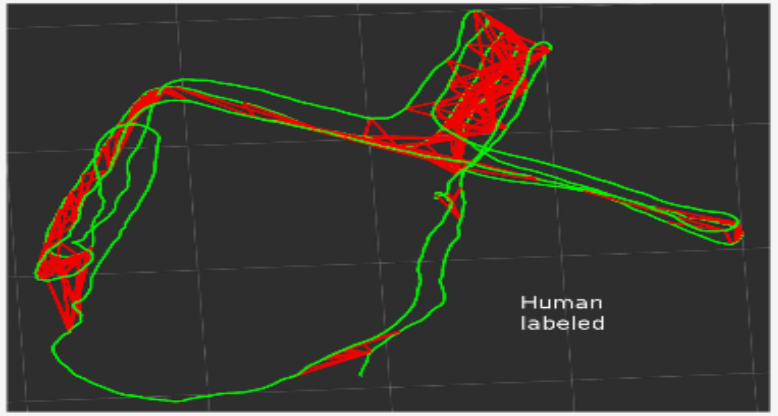


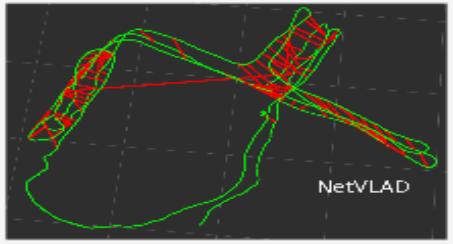
Sequence: Base-2

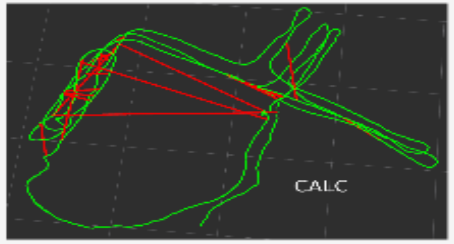


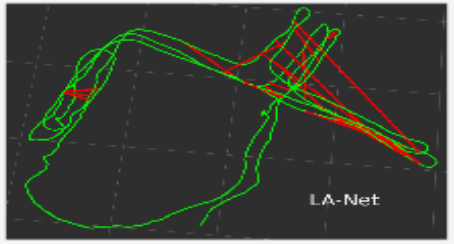


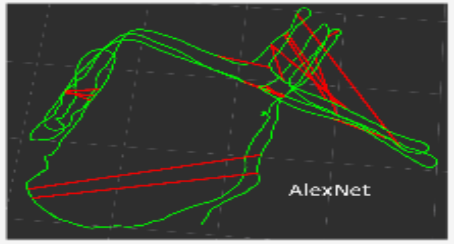


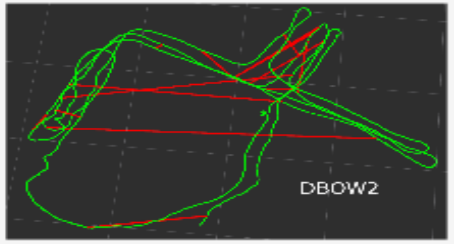




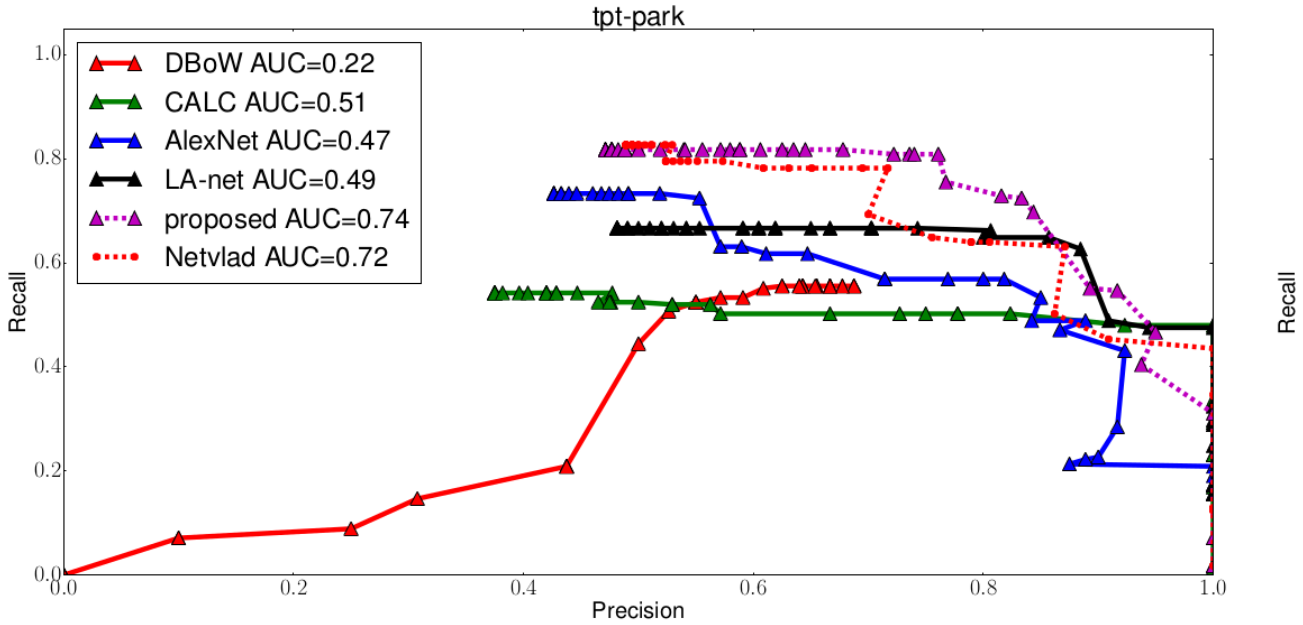


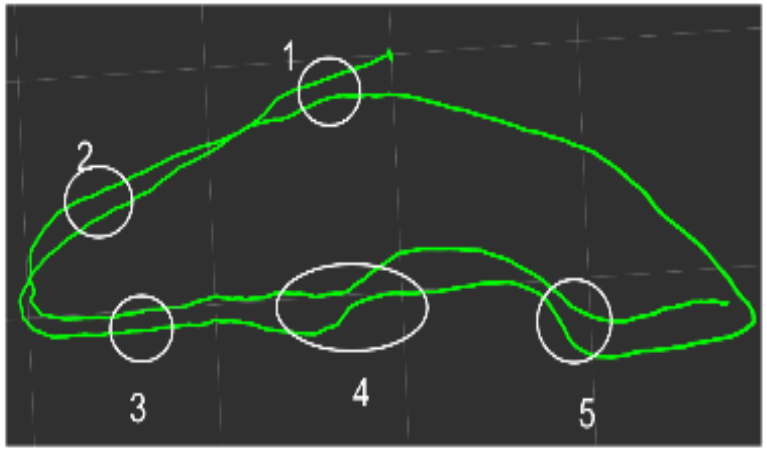


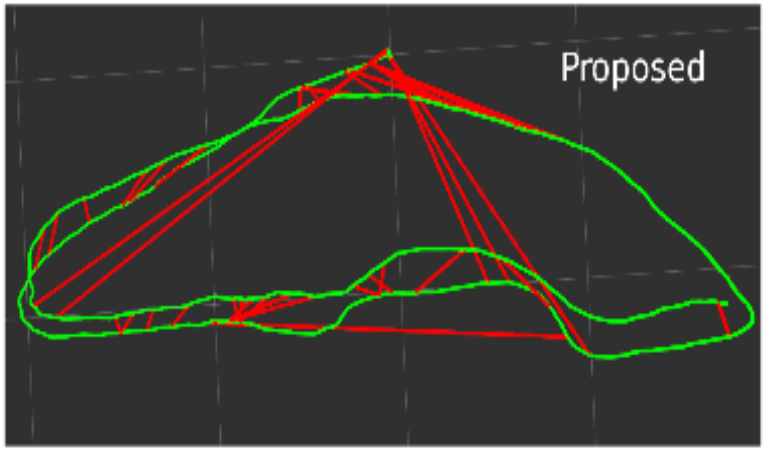


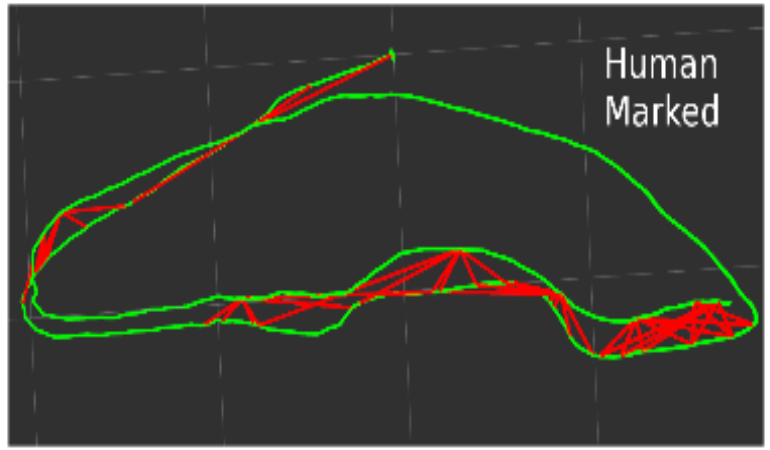


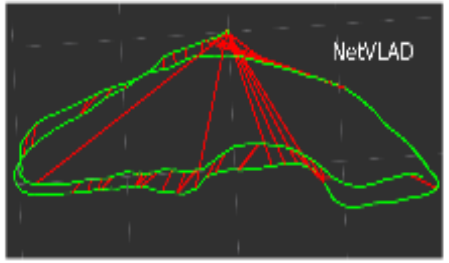
Sequence: TPT-Park

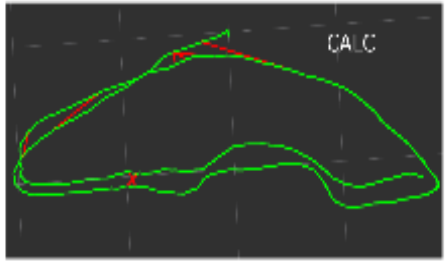


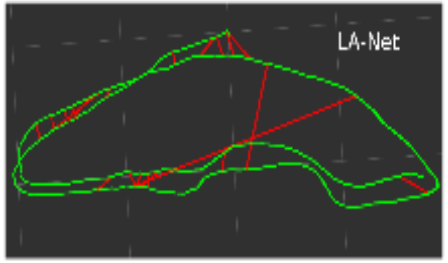


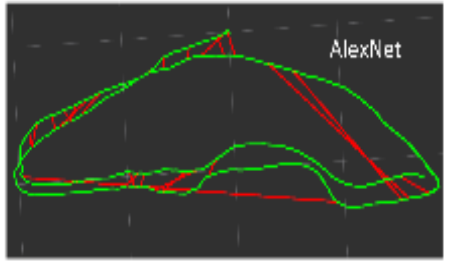


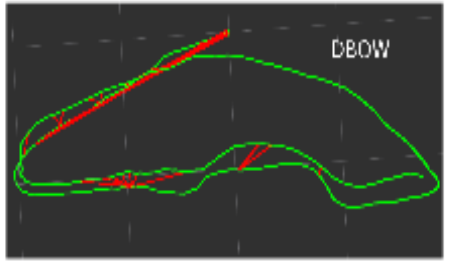




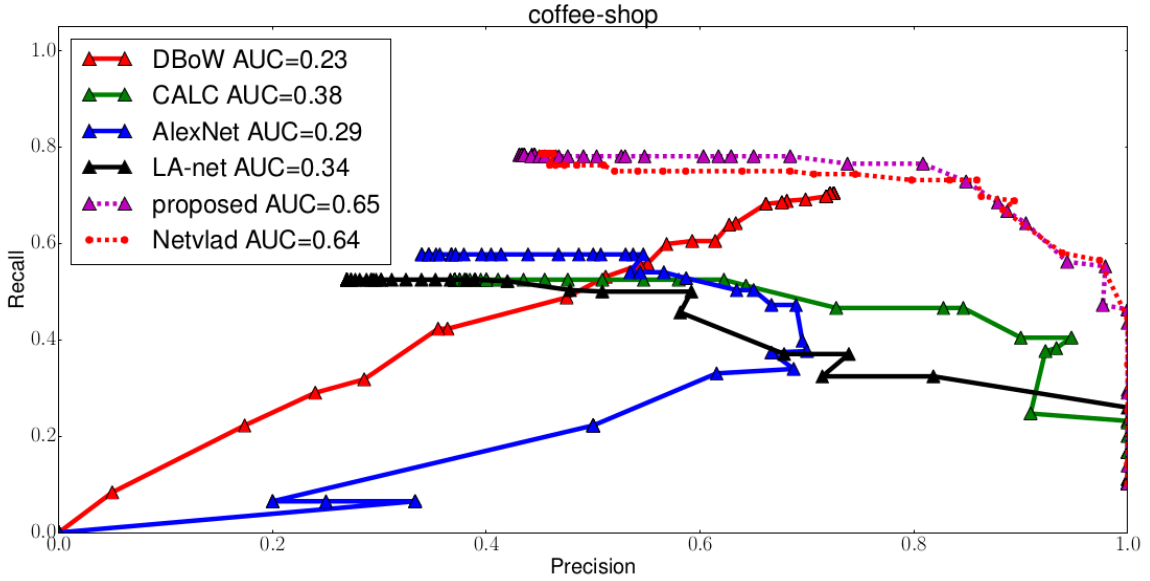


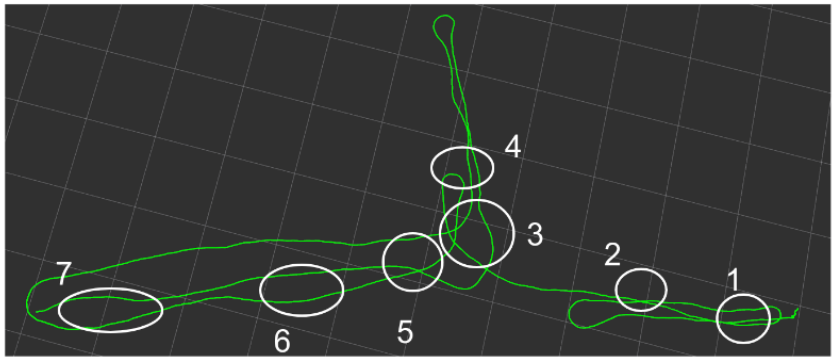


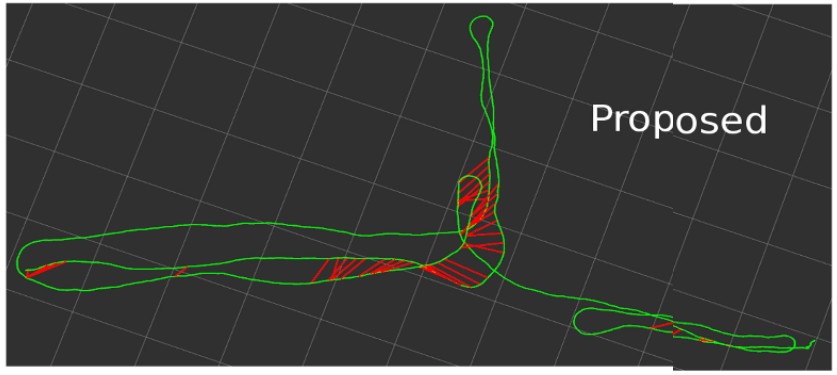


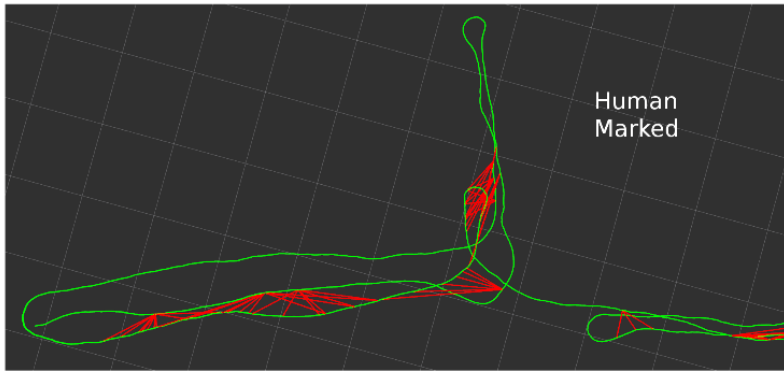


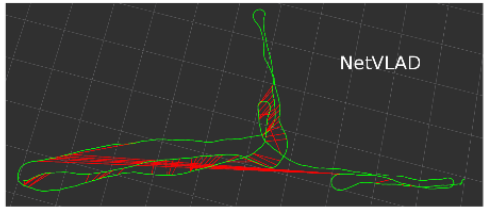
Sequence: Coffee-shop

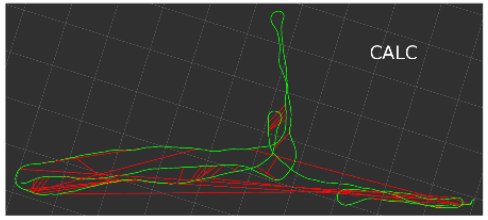


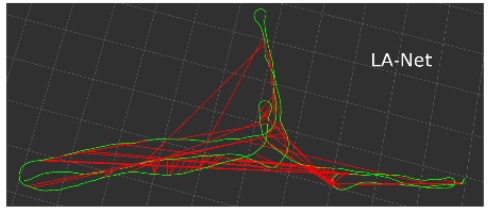


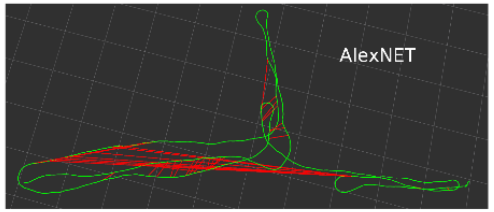


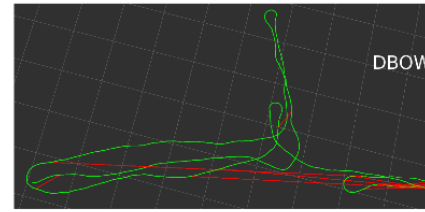




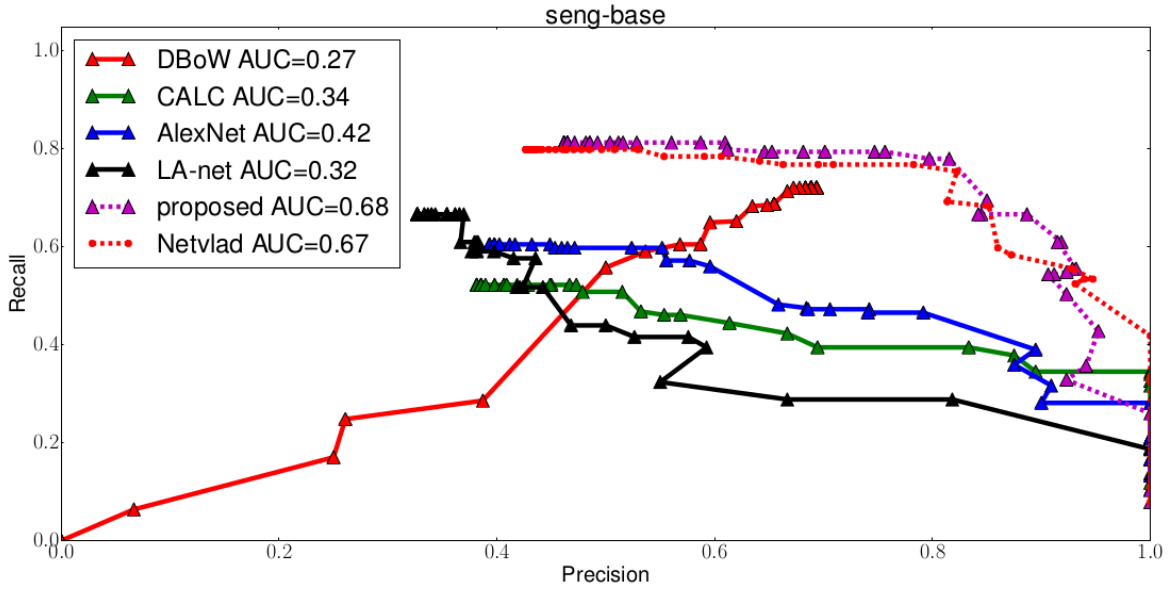


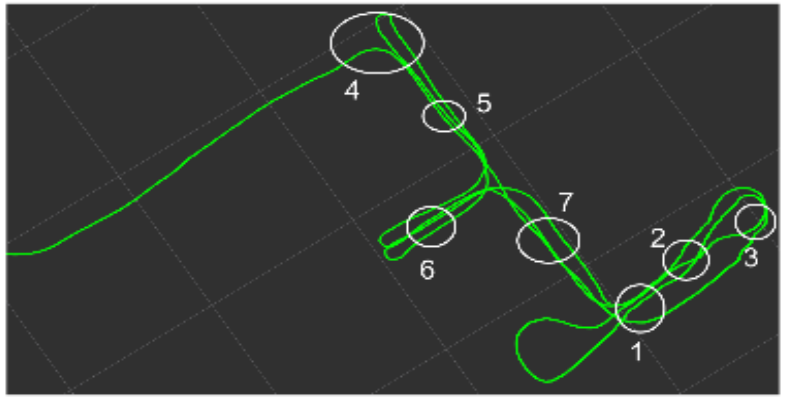


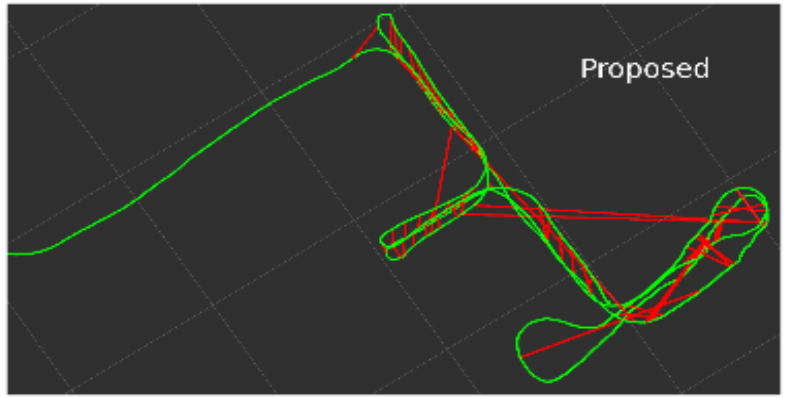


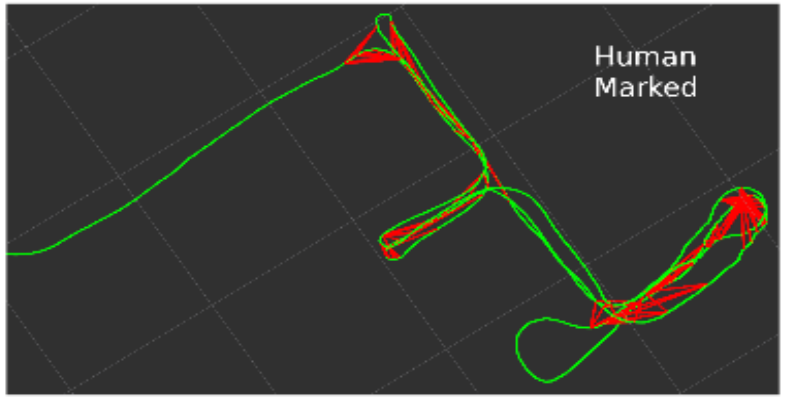


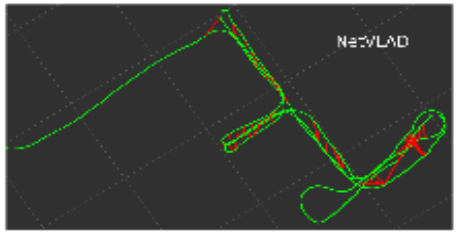
Sequence: Seng-base

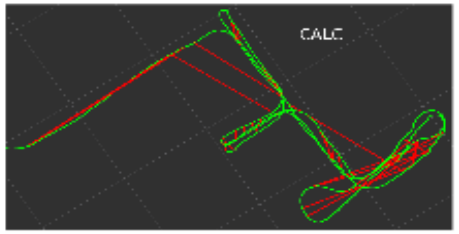


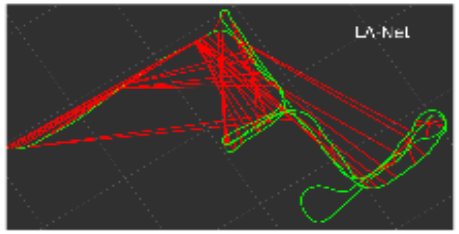


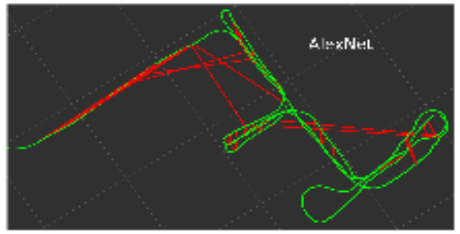


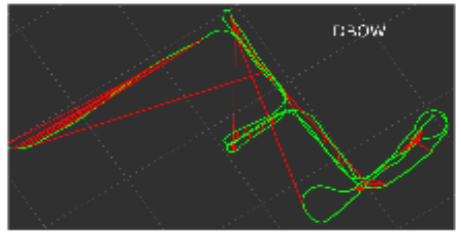




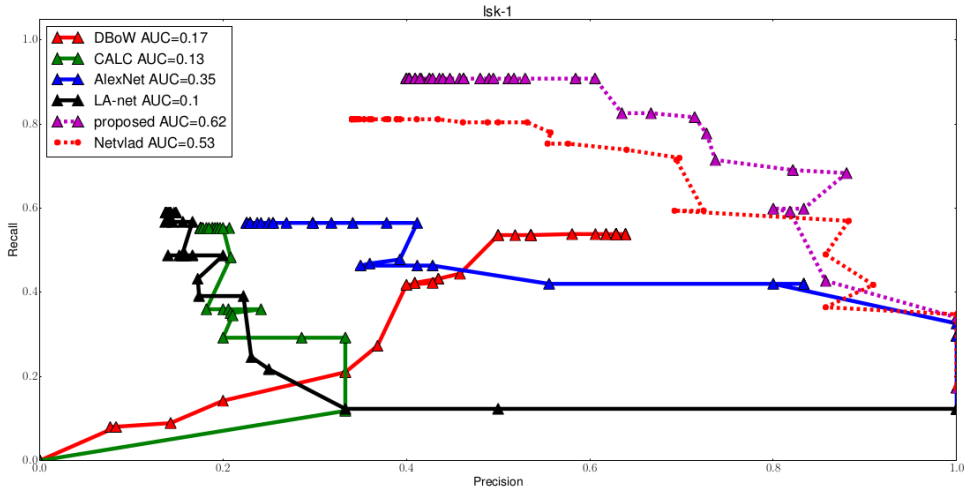


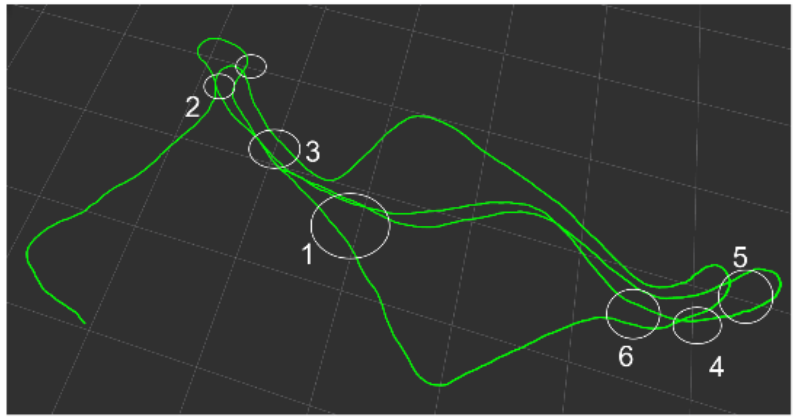


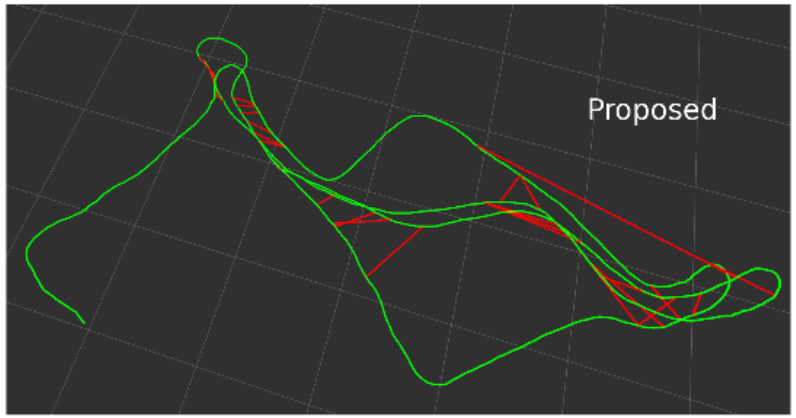


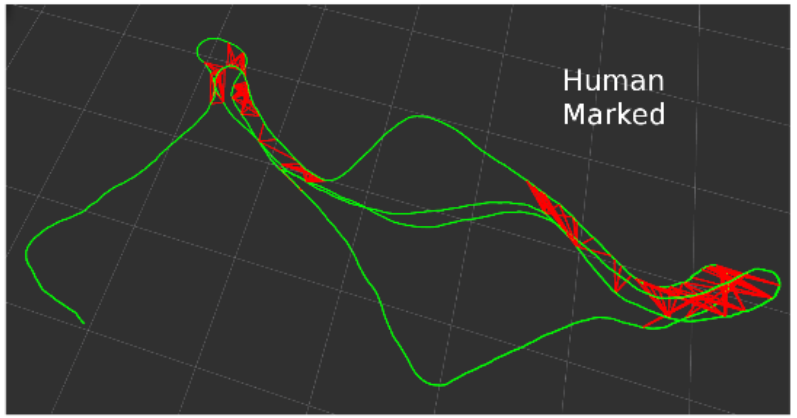


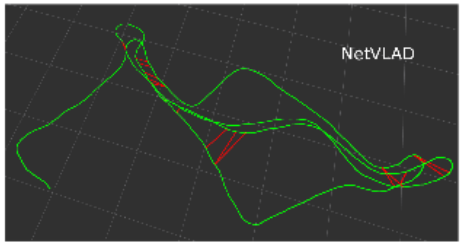
Sequence: LSK-1

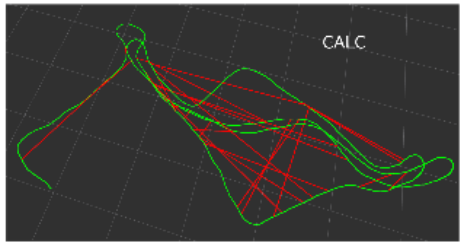


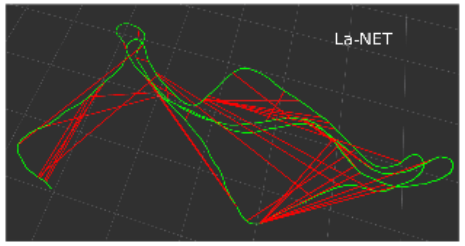


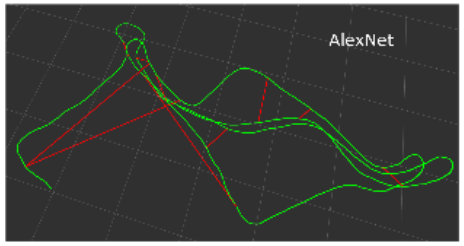


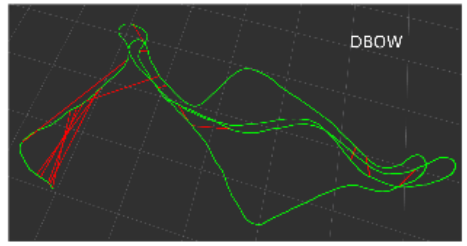




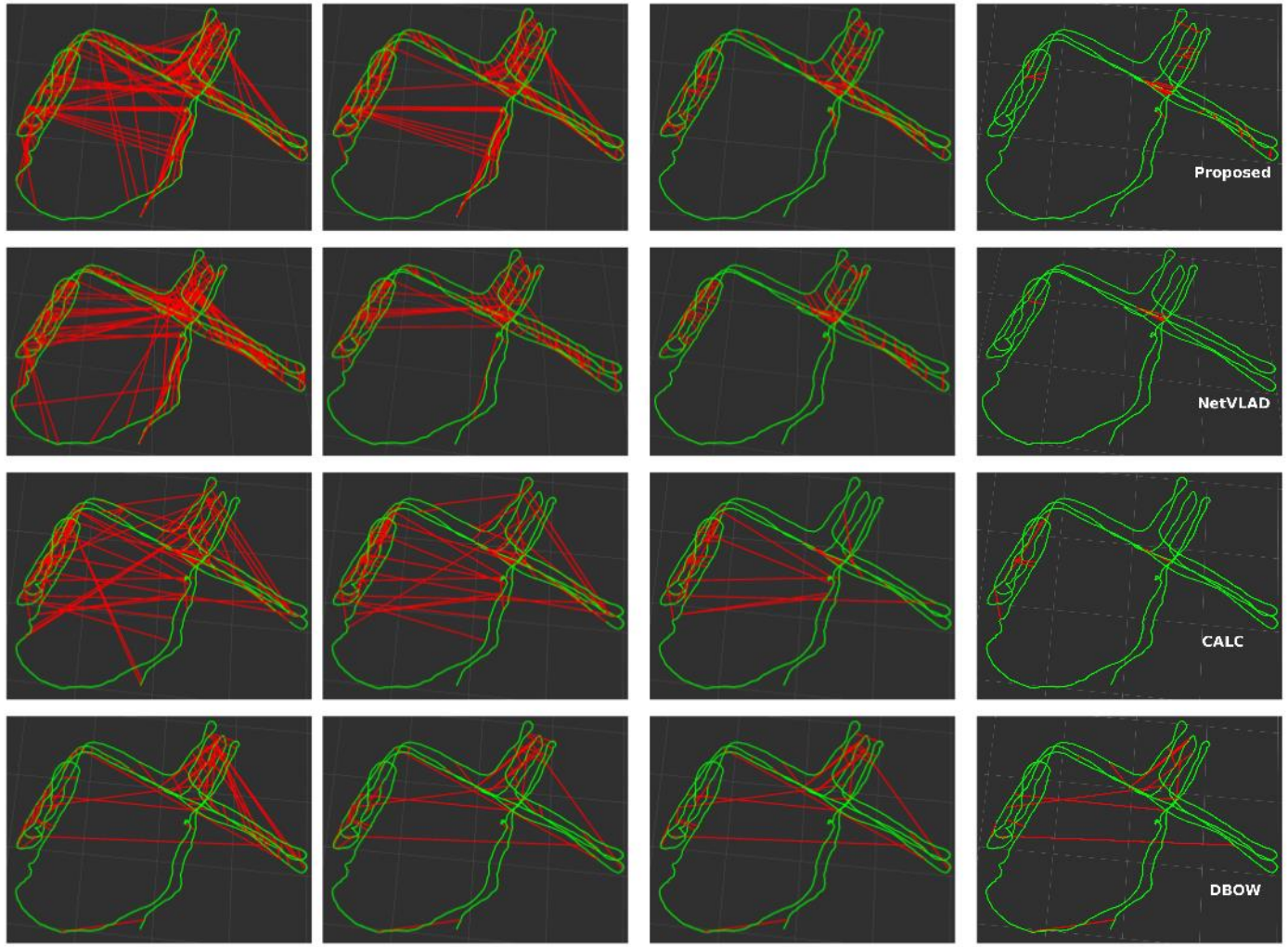






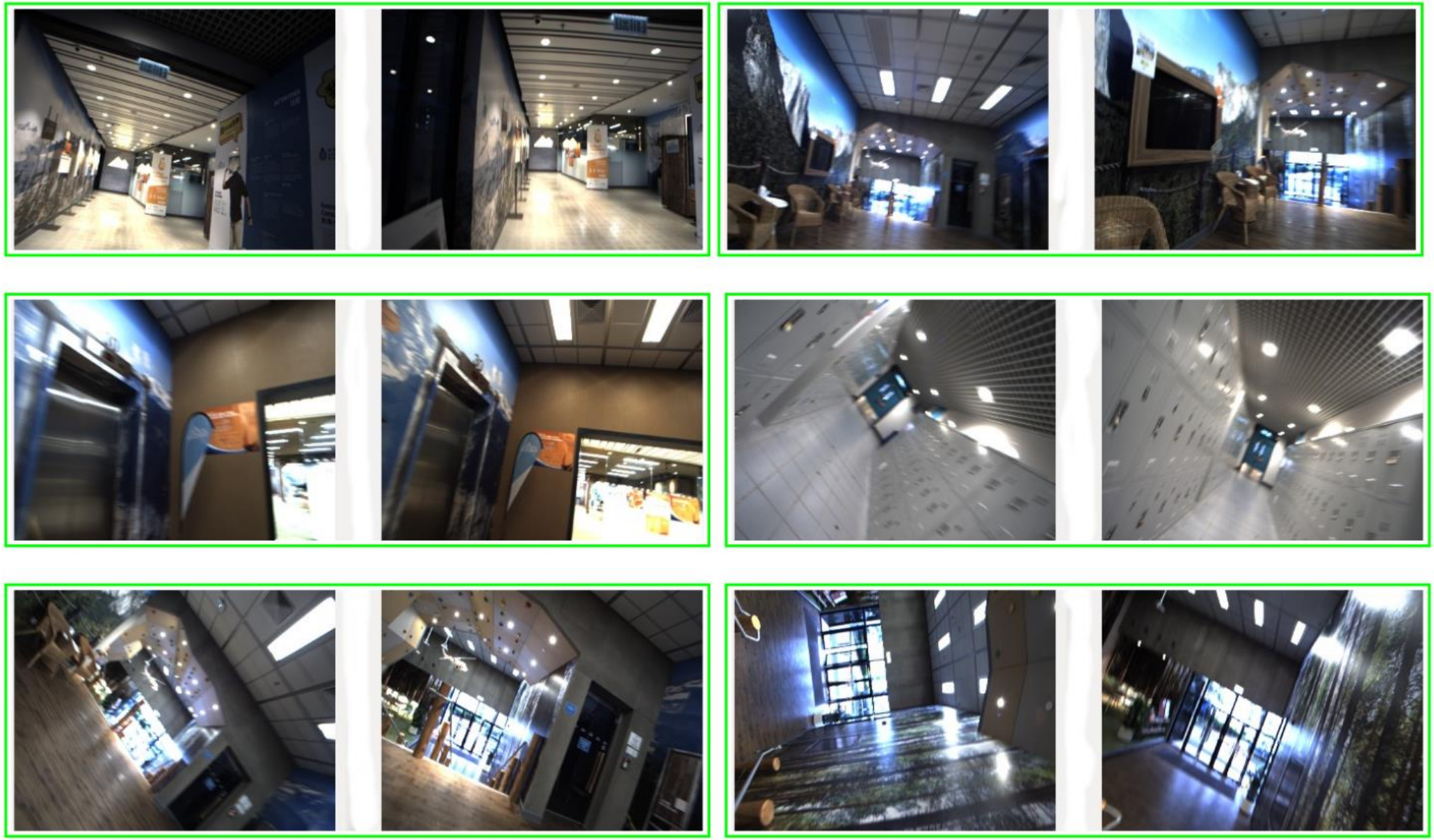


Changing the Threshold

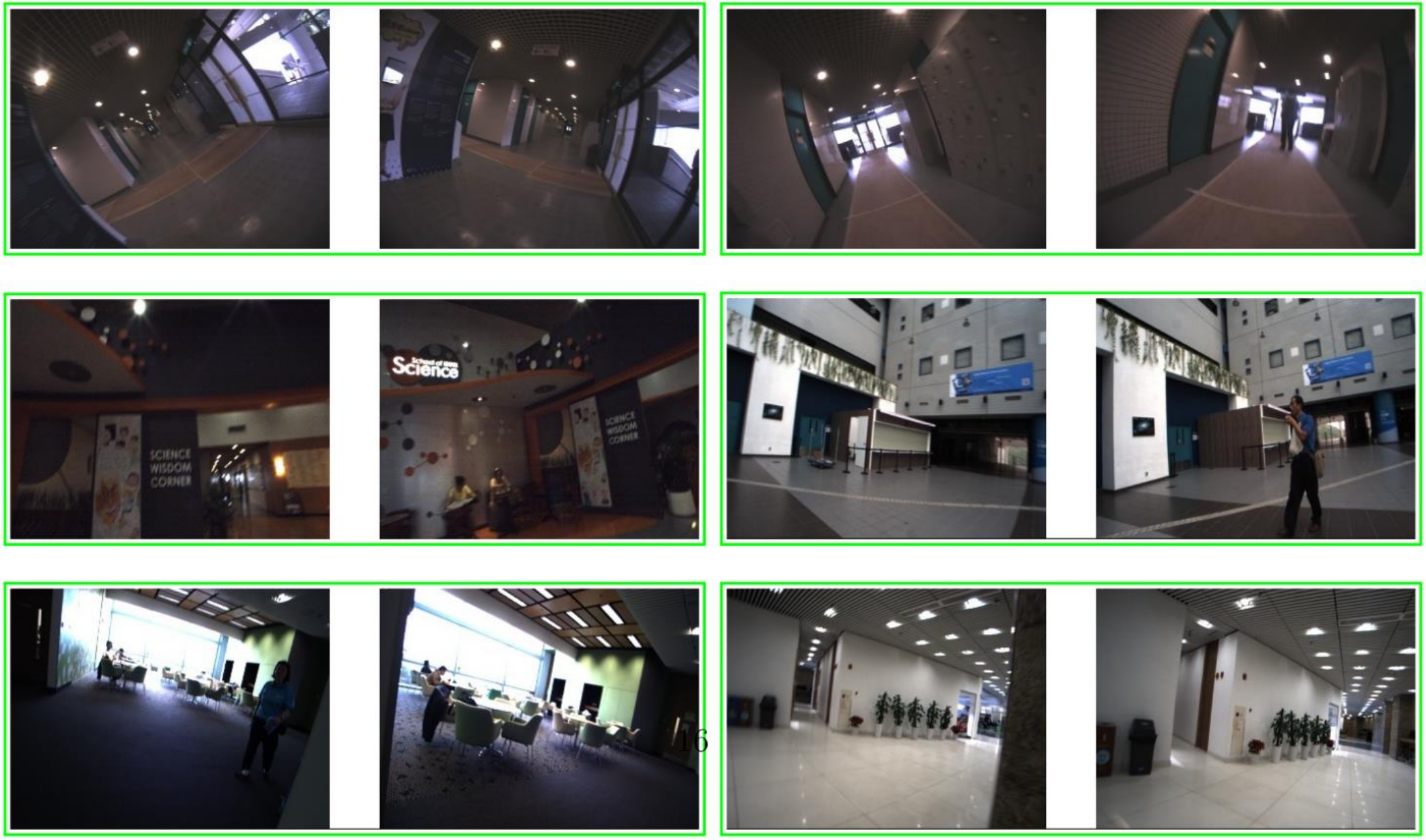


Looser Threshold → Stricter Threshold

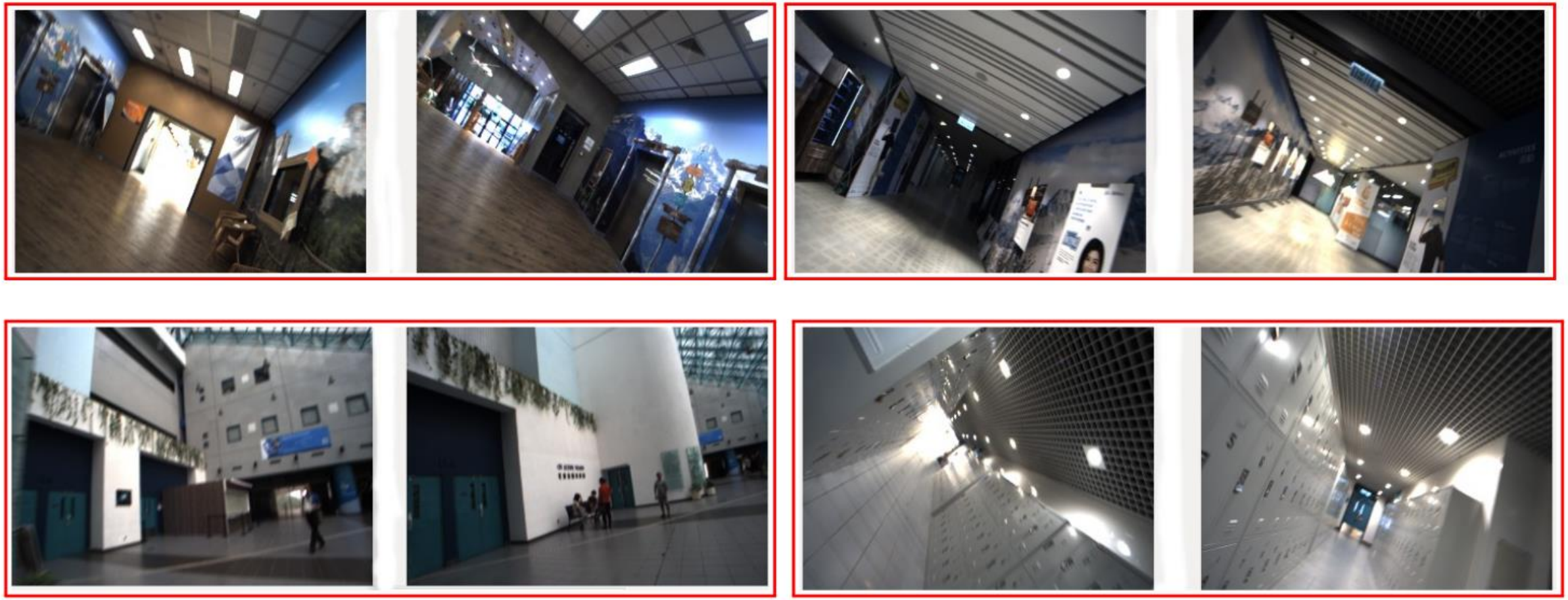
Example True Positives - I



Example True Positives - I



Example False Positives - I

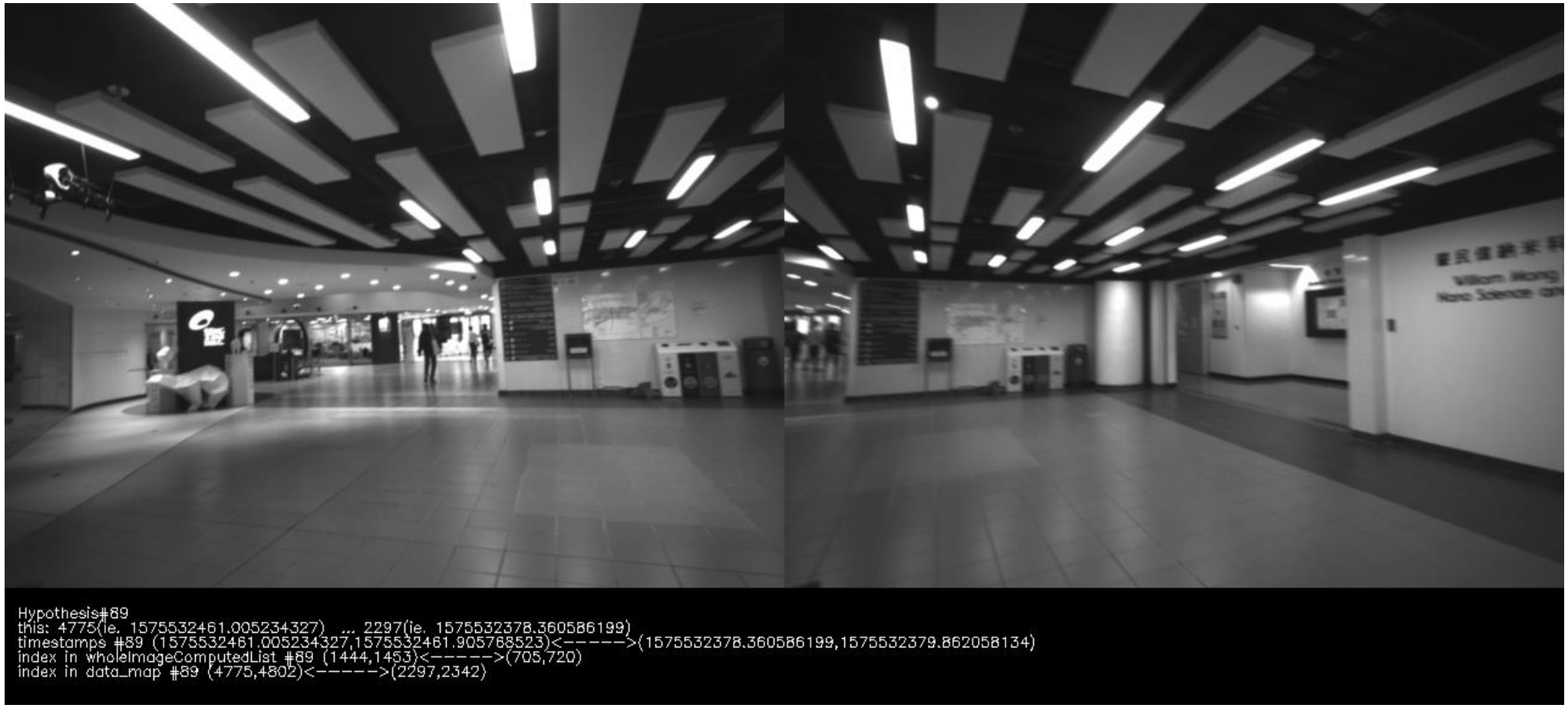


Walking Around in HKUST: True Positives

Walking Around in HKUST: True Positives

Walking Around in HKUST: True Positives

True Positive and Pose Computed



False Positive

False Positives

Conclusion from CNN Based Image Descriptor

- Higher recall rates compared to loop-detection system in state-of-the-art SLAM system
- An order of magnitude faster than NetVLAD; comparable precision-recall performance

Contributions

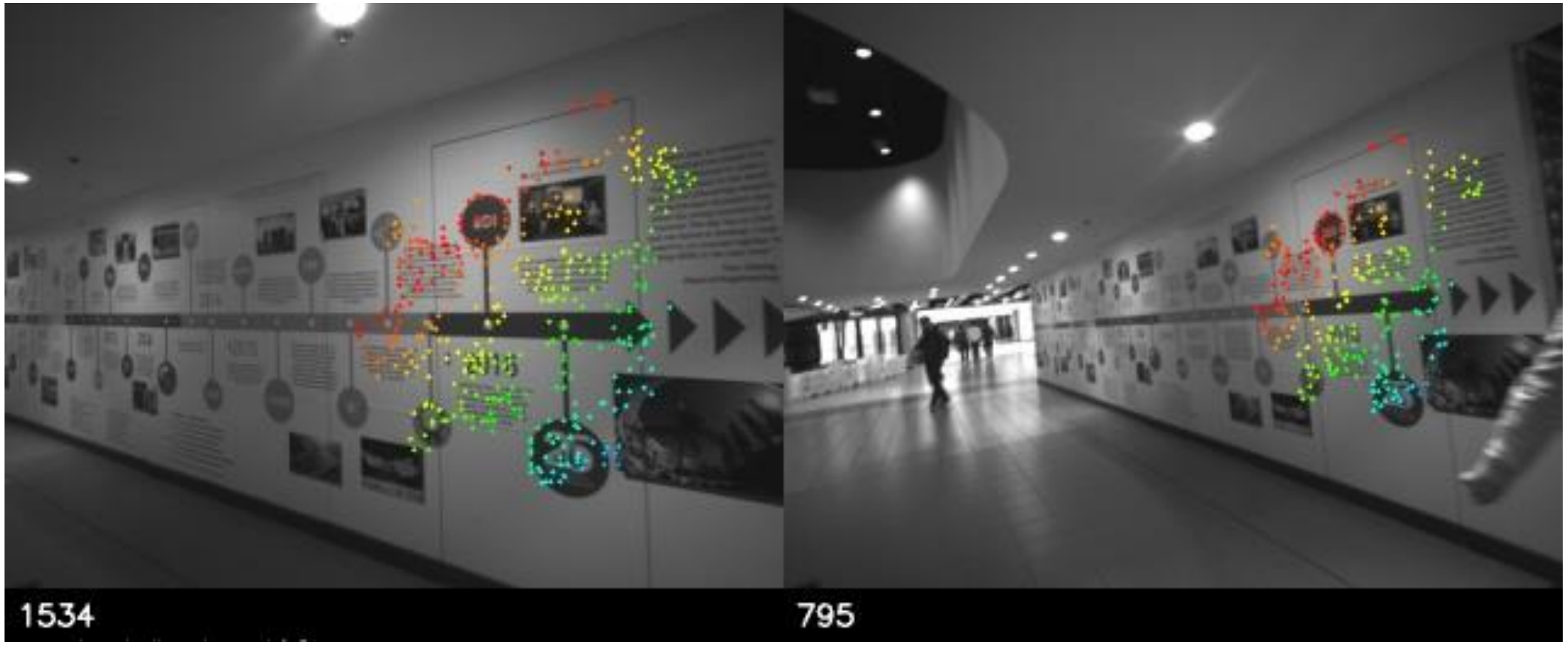
A: Edge Based Visual Odometry System

B: CNN Based Place Recognition System

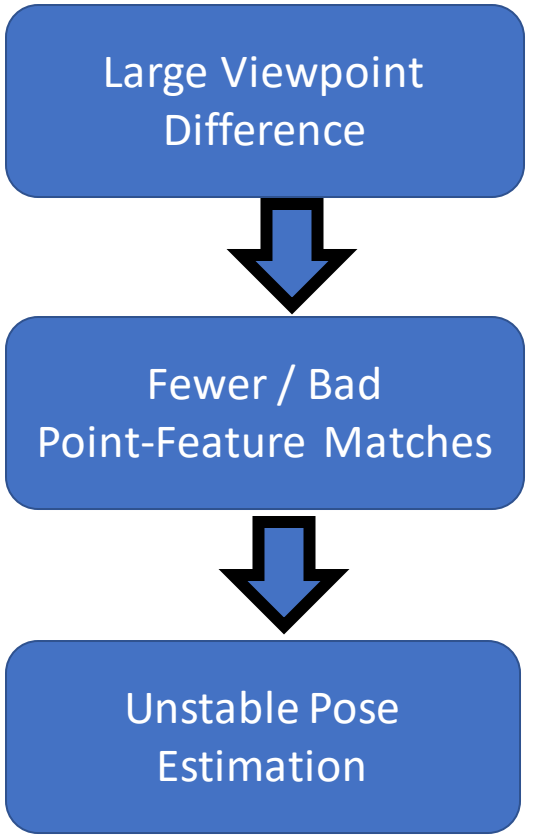
C: Large Viewpoint Pose Computation

D: Kidnap Aware Pose Graph Solver

Contribution-C:
Large Viewpoint Pose Computation



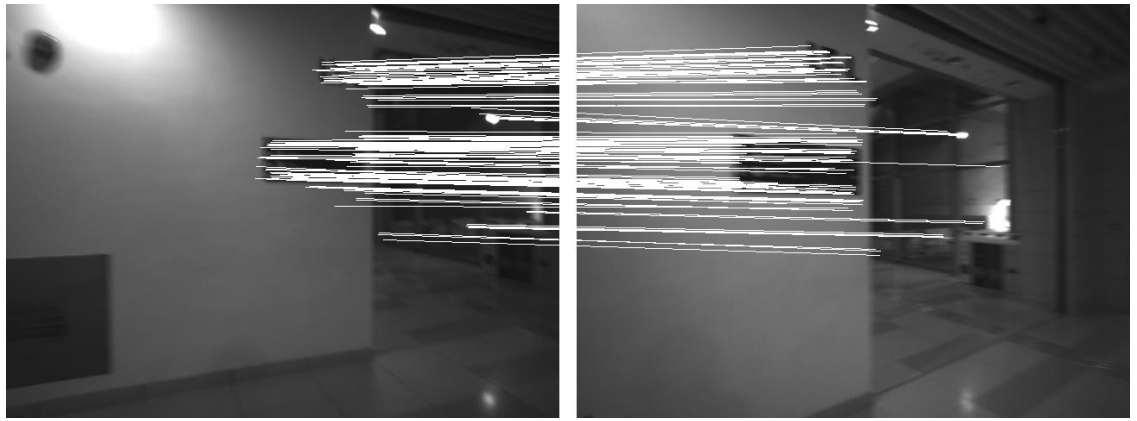
Challenges and Motivation



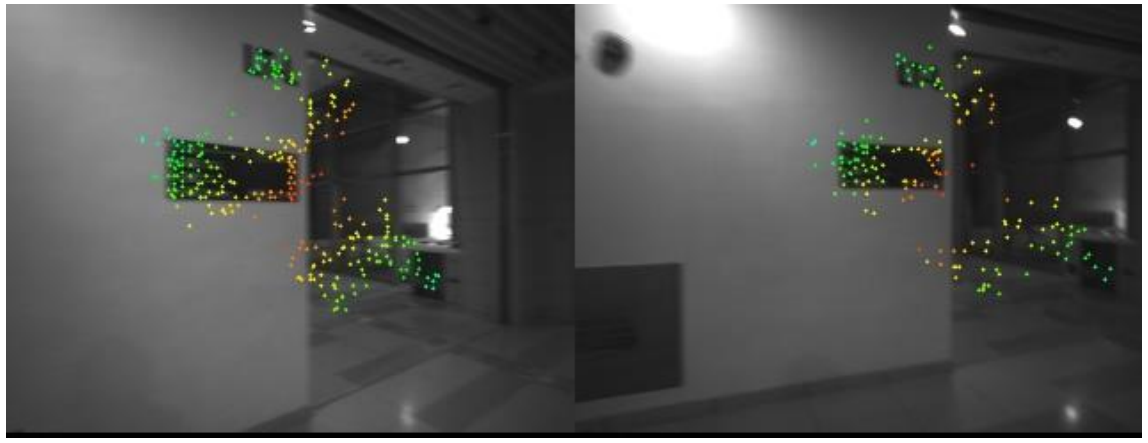
- If relying on standard real-time pipelines
  - ORB features
  - Perspective-N-Points (PnP)
- Unstable relative pose estimation
- Effect of a better place recognition is nullified if using standard real-time pose computation pipeline
- Challenges:
  - True positive revisit candidate AND no point feature matches
  - True positive revisit candidate AND bad (noisy + spurious matches)
  - Accuracy of Pose Computation in presence of: noisy + spurious matches

GMS Matcher vs SIFT vs ASIFT vs ORB

SIFT, ORB Results in 0 Matches



ASIFT : Results in 140 feature matches in 4 seconds of processing, too slow for realtime pipelines

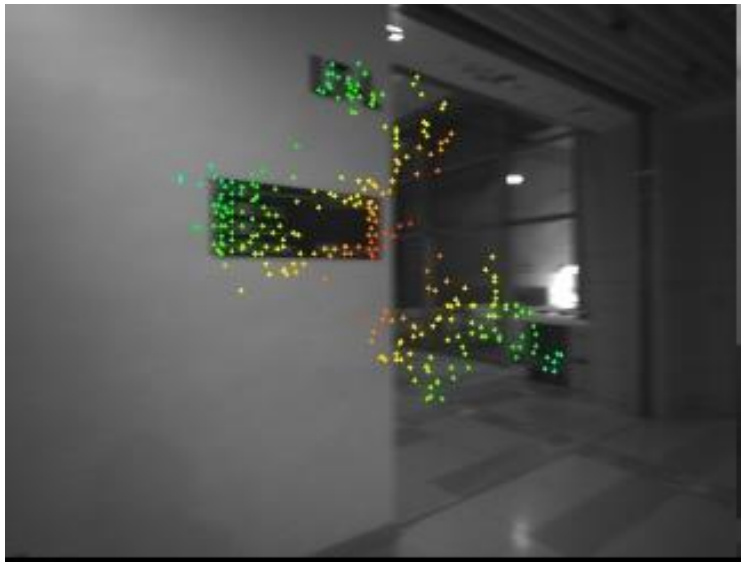


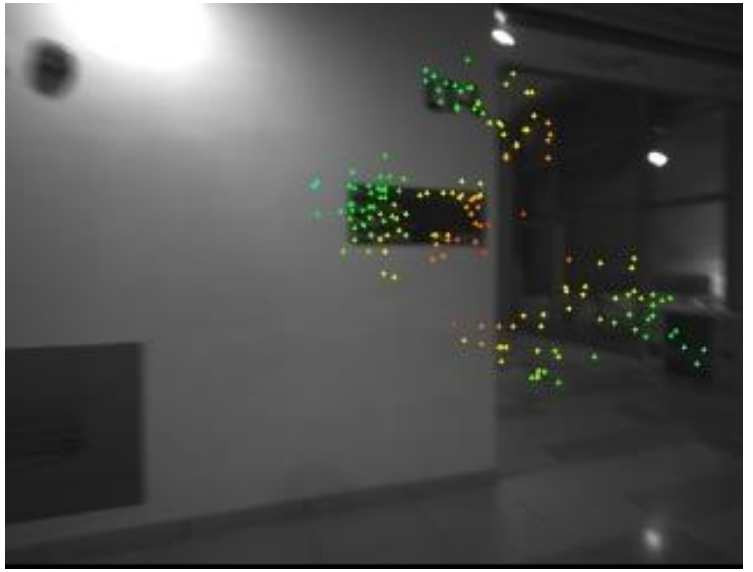
GMS-Matcher: 289 point feature matching.

[ASIFT] Guoshen Yu, and Jean-Michel Morel, ASIFT: An Algorithm for Fully Affine Invariant Comparison, Image Processing On Line, 1 (2011), pp. 11–38. https://doi.org/10.5201/ipol.2011.my-asift
[GMS-Matcher] Bian, JiaWang, et al. "Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

GMS-Matcher

- 289 Point feature matches
- Running time:
  - Desktop PC, 640x480 resolution: 55 ms
  - Desktop PC, 320x240 : 20-30ms
  - Intel NUC : 640x480: 120ms
  - Intel NUC, 320x240: 40ms
- Issues:
  - Results in imprecise matching.
  - Around 30-40% False Matches





This result uses 320x240 image

Pose Computation (Proposed)

- Input: Loop Hypothesis (Sequence)
- Output: Relative Pose between the sequence

1. GMS-matcher 30-100ms
2. Sparsify the points and track in adjacent views 5-30ms, proportional to number of adjacent views used)
3. Fast Global pose estimation with Alternating Minimizations: 3D-3D alignment 5ms
4. Four-Degree-of-Freedom (4DOF) Refinement with Edge Alignment. 70-150ms
    - Pitch, Roll from VIO
    - yaw, tx, ty, tz from (3).

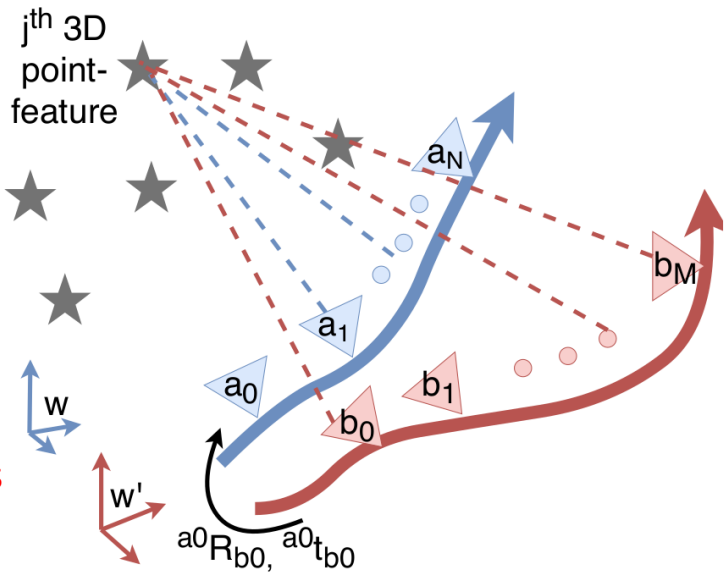


Fig. 1. Notations for the proposed method.

Cumulate Tracked Points

$$
\begin{aligned}
{}^{a_p}X_i^{(a_p)} &= \pi^{-1}((u_i, v_i), D_{a_p}(u_i, v_i)) \\
{}^{b_q}X_i^{(b_q)} &= \pi^{-1}((u'_i, v'_i), D_{b_p}(u'_i, v'_i))
\end{aligned}
$$

$$
\begin{aligned}
{}^{a_0}X_i^{(a)} &= ({}^{w}\mathbf{T}_{a_0})^{-1}\ {}^{w}\mathbf{T}_{a_p}\ {}^{a_p}X_i^{(a_p)} \\
{}^{b_0}X_i^{(b)} &= ({}^{w'}\mathbf{T}_{b_0})^{-1}\ {}^{w'}\mathbf{T}_{b_q}\ {}^{b_q}X_i^{(b_q)}
\end{aligned}
$$

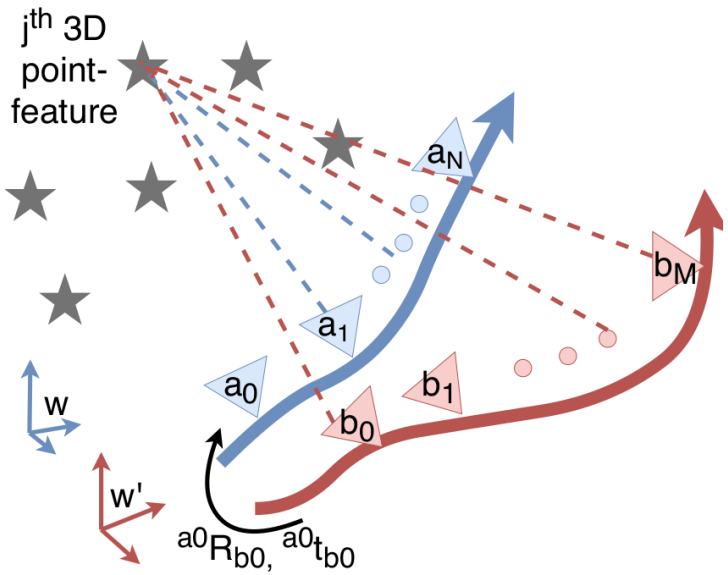


Fig. 1. Notations for the proposed method.

Fast Global pose estimation with Alternating Minimizations

- **Doesn't need an initial guess**
- **Inaccurate estimation**
- **Can identify false matches with the switch constraints**

$$f(X_j, \mathbf{R}, \mathbf{t}) = || \ {}^{a_0}X_j^{(a)} - \mathbf{R} \ {}^{b_0}X_j^{(b)} - \mathbf{t}||_2^2$$

$$\underset{\mathbf{R},\mathbf{t},s_j j=1\ldots K}{\text{minimize}} \quad F(\mathbf{R}, \mathbf{t}, \mathbf{s}) = \sum_{j=1}^{K} s_j^2 f(X_j, \mathbf{R}, \mathbf{t}) + \lambda(1 - s_j)^2$$

$$\text{s.t.} \quad \mathbf{R} \in SO(3)$$

Result with GMS-matcher and this method

$$\mathbf{R}^{(n-1)}, \mathbf{t}^{(n-1)} = \underset{R,t}{\text{argmin}} \ F(\mathbf{R}, \mathbf{t}, \mathbf{s}^{(n-1)})$$

$$s^{(n)} = \underset{s}{\text{argmin}} \ F(\mathbf{R}^{(n-1)}, \mathbf{t}^{(n-1)}, \mathbf{s})$$

${}^{c}X$ ${}^{r}\hat{T}_c \ {}^{c}X$

4-DOF Pose Refinement with Edge Alignment

- Iterative method
- Optimization variables: yaw, translation in the IMU frame
- Uses intial guess for pose
  - Yaw, translation from the global method converted to imu frame, optimization variables
  - Pitch and roll from VIO (in IMU frame), fixed constants

Distance transform of frame-a

Perspective projection

Imu-camera calibration

Edge points of frame-b

$$ {}^{a_{imu}}\hat{T}_{b_{imu}} = \underset{{}^{a_{imu}}T_{b_{imu}}}{\text{minimize}}\ V^{a}\left(\Pi[({}^{imu}T_{cam})^{-1}\ {}^{a_{imu}}T_{b_{imu}}\ {}^{imu}T_{cam}\ {}^{b}X] \right) $$

$$ {}^{a}T_{b} = ({}^{imu}T_{cam})^{-1}\ {}^{a_{imu}}\hat{T}_{b_{imu}}\ {}^{imu}T_{cam} $$

Optimization variable (yaw, translation)

Result of Alignment with Edge Alignment (EA)

^{c}X

Current Image

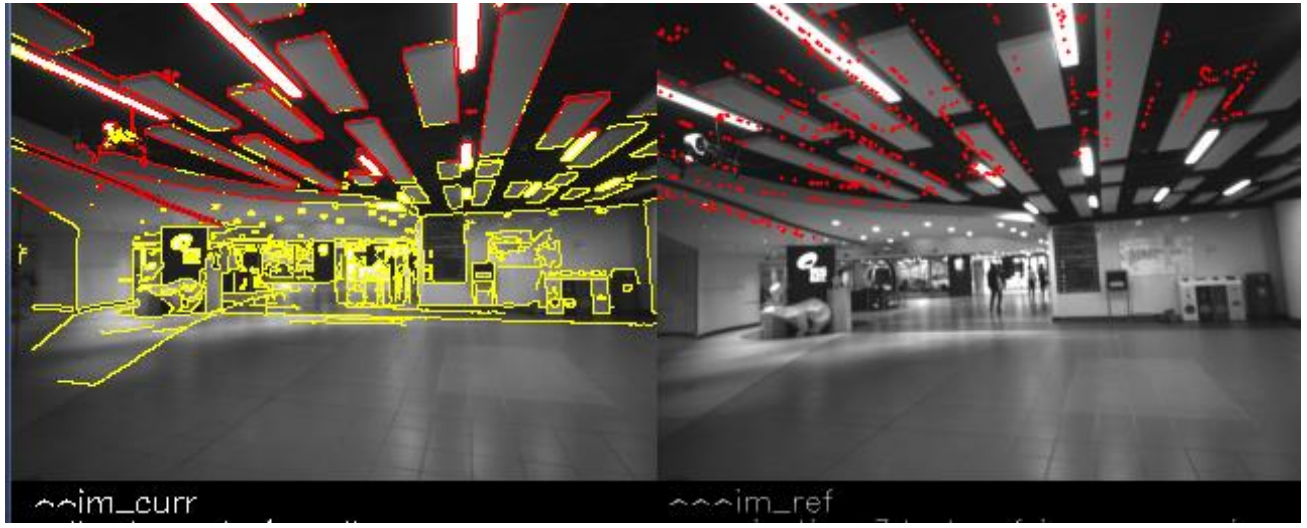
$^{r}\hat{T}_{c}\ ^{c}X$

Reference Image

$^{r}\hat{T}'_{c}\ ^{c}X$

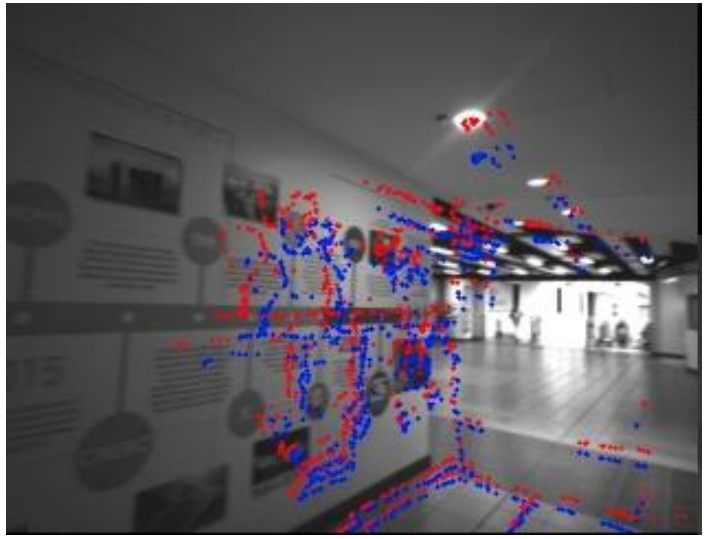
Reference Image
After EA refinement

PNP (+ RanSAC) with ORB Features (blue)
PNP (+ RanSAC) with Gms-Matcher



6DOF Edge Alignment

4DOF Edge Alignment



PNP (+RanSAC) with ORB Features (blue)
PNP (+RanSAC) with Gms-Matcher

6DOF Edge Alignment

4DOF Edge Alignment

Conclusion: Pose Computation

- Relative pose computation in the wild is hard!
- With direct edge-based refinement tighter pose estimates
- 4-DOF direct refinement (edge-alignment) help us narrow the gap

Contributions

A: Edge Based Visual Odometry System

B: CNN Based Place Recognition System

C: Large Viewpoint Pose Computation

D: Kidnap Aware Pose Graph Solver

Contribution-D

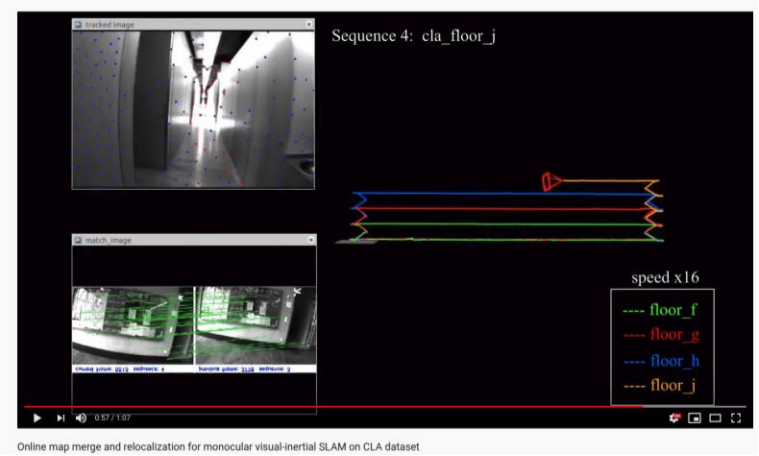
Kidnap Handling for SLAM System

Kidnap

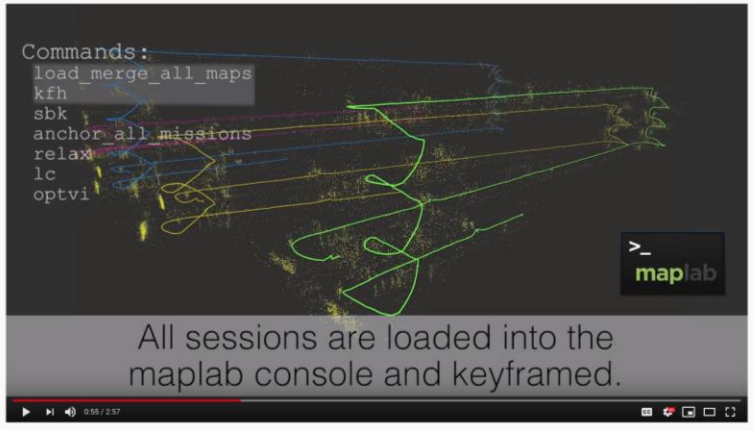
- Blocking out the camera for more than 20-30s.
- Since the vision part doesn't produce credible estimates (during kidnap),
  - it is essentially IMU dead-reckon --> Significant drift
- Current SLAM systems (including VINS-Fusion, Rovio, ORB-SLAM, OKVIS) cannot recover from kidnaps

Literature – Relocalization / Trajectory Merging

- Qin et. al (Previous work from HKUST-UAV)
  - Supports live-merge only if loop connection with 1st world
  - Cannot handle the general case (loop connections between say world#2 and world#3 ignored)
  - No explicit mechanism to detect kidnap and stop/start VINS.
- Maplab (Rovio-li)
  - Offline trajectory merging only
- Google Cartographer
  - Offline trajectory merging only
- HF-NET (CVPR2019)
  - Discriptor computation only. No full system.



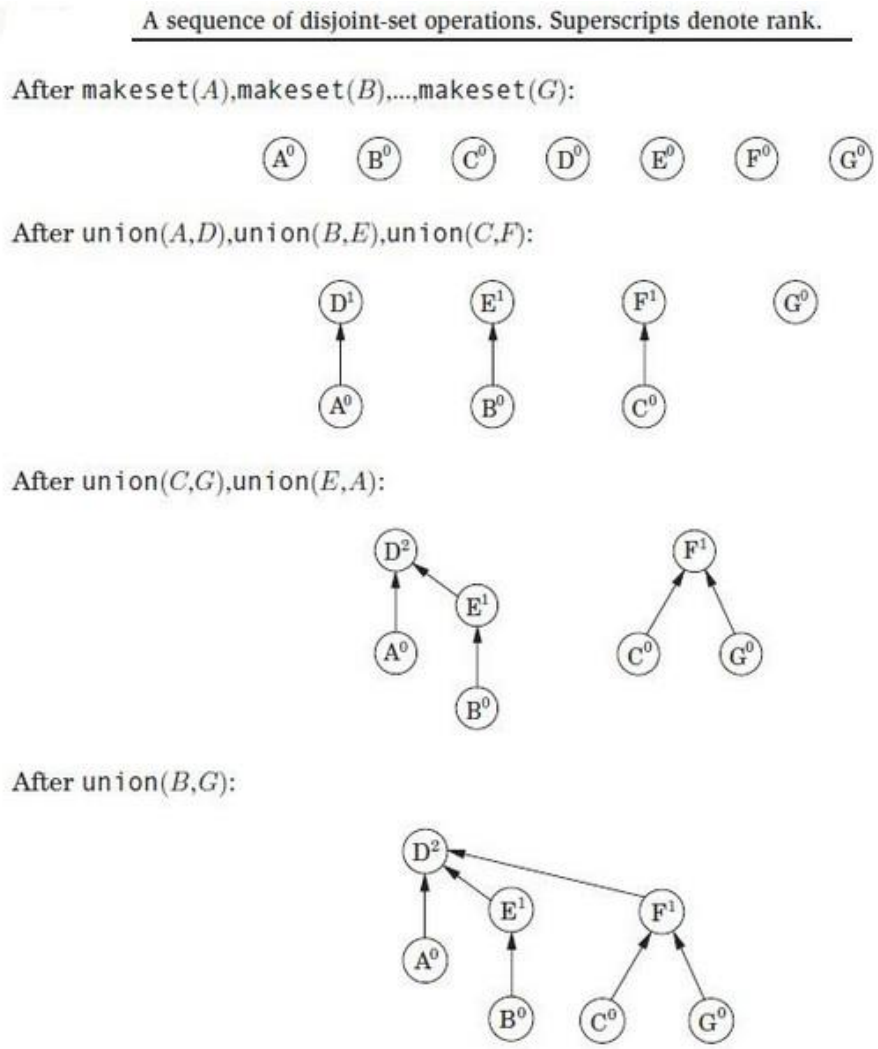
Online map merge and relocalization for monocular visual-inertial SLAM on CLA dataset



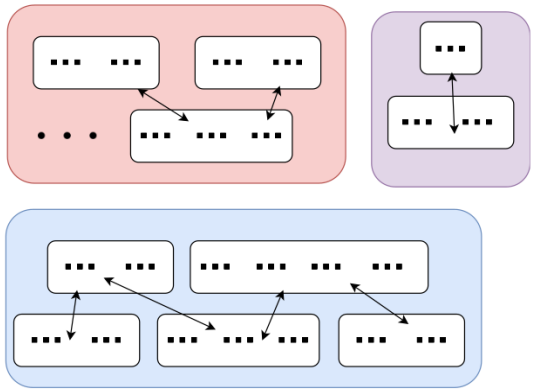
- T. Qin, P. Li and S. Shen. Relocalization, global optimization and map merging for monocular visual-inertial SLAM. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1197-1204, Brisbane, Australia, May 2018.

Our Solution – Data Structure

- Use Disjoint set – tree based data structure with the following operations:
  - Make_set(a)
  - Union(a,b)
  - Find_root(c)



Details



- Each co-ordinate system is a node in disjoint set
  - Each time VINS is restarted (due to kidnap) we do `make_set()`
- Loop connection between different worlds (say world#m and world#n) we do :
  - union(find_root(m), find_root(n))
  - The edge also hold the relative pose between those two co-ordinate systems aka:
- We solve the pose graph optimization for each world-set in co-ordinate frame of root of each set.
  - In general we need to perform BFS (breadth first search) to get the ${}^{root}T_n$

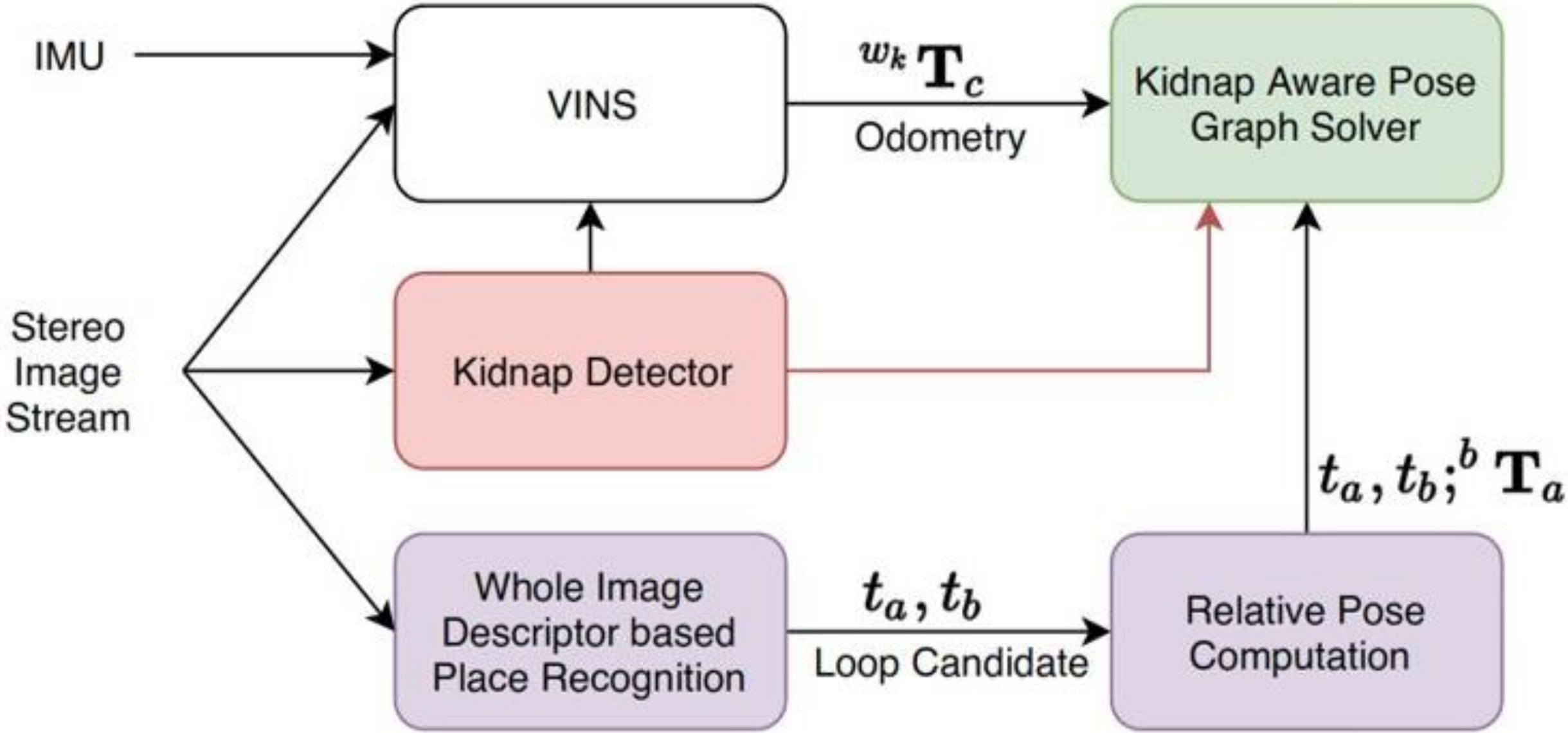
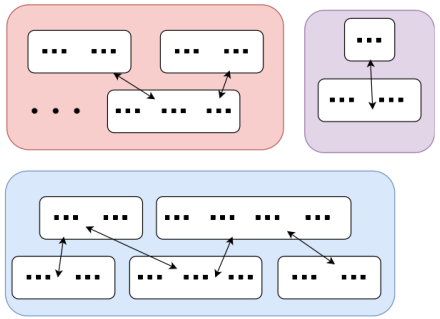
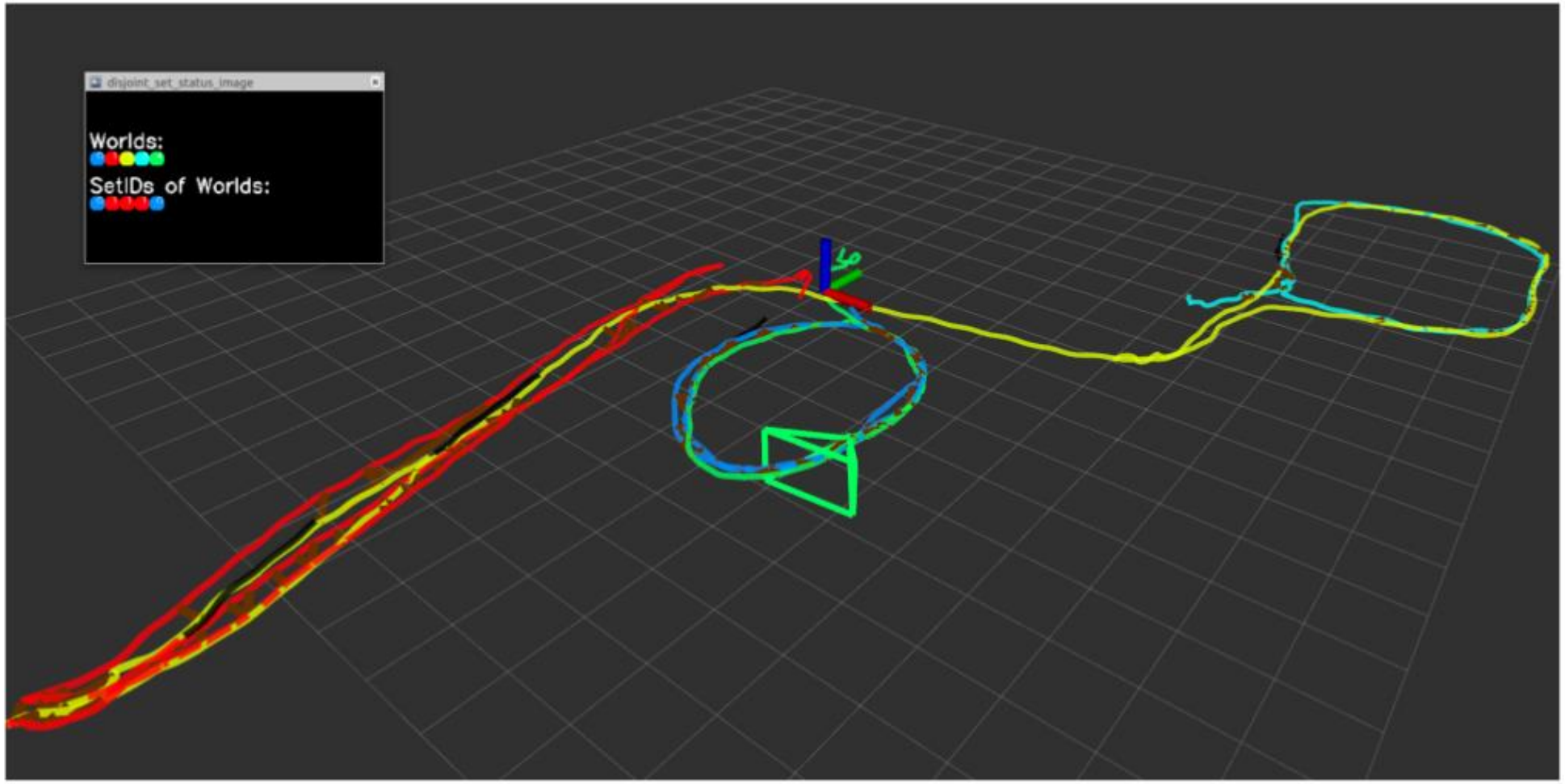


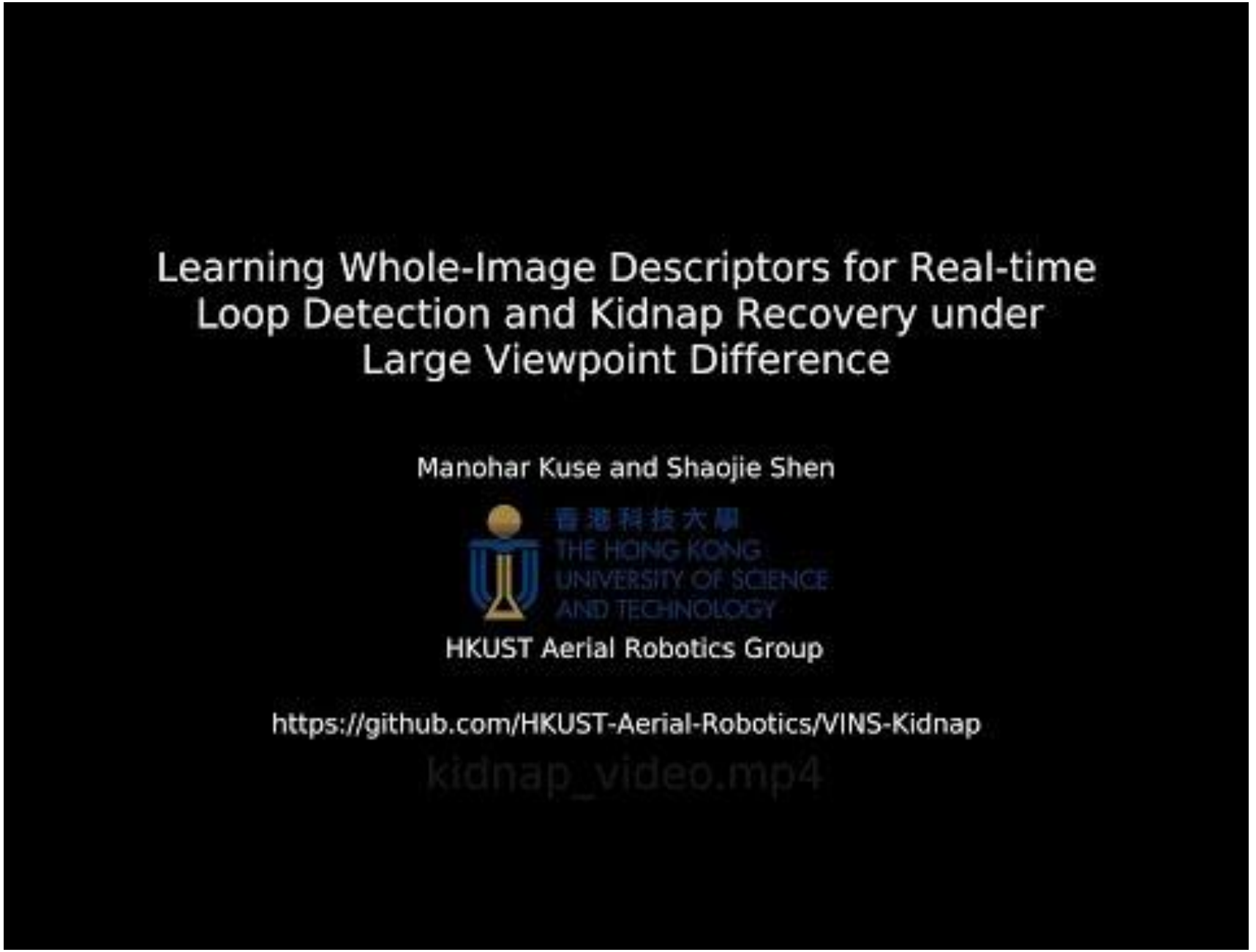
Figure 4: System Overview

Complicated Kidnap Cases



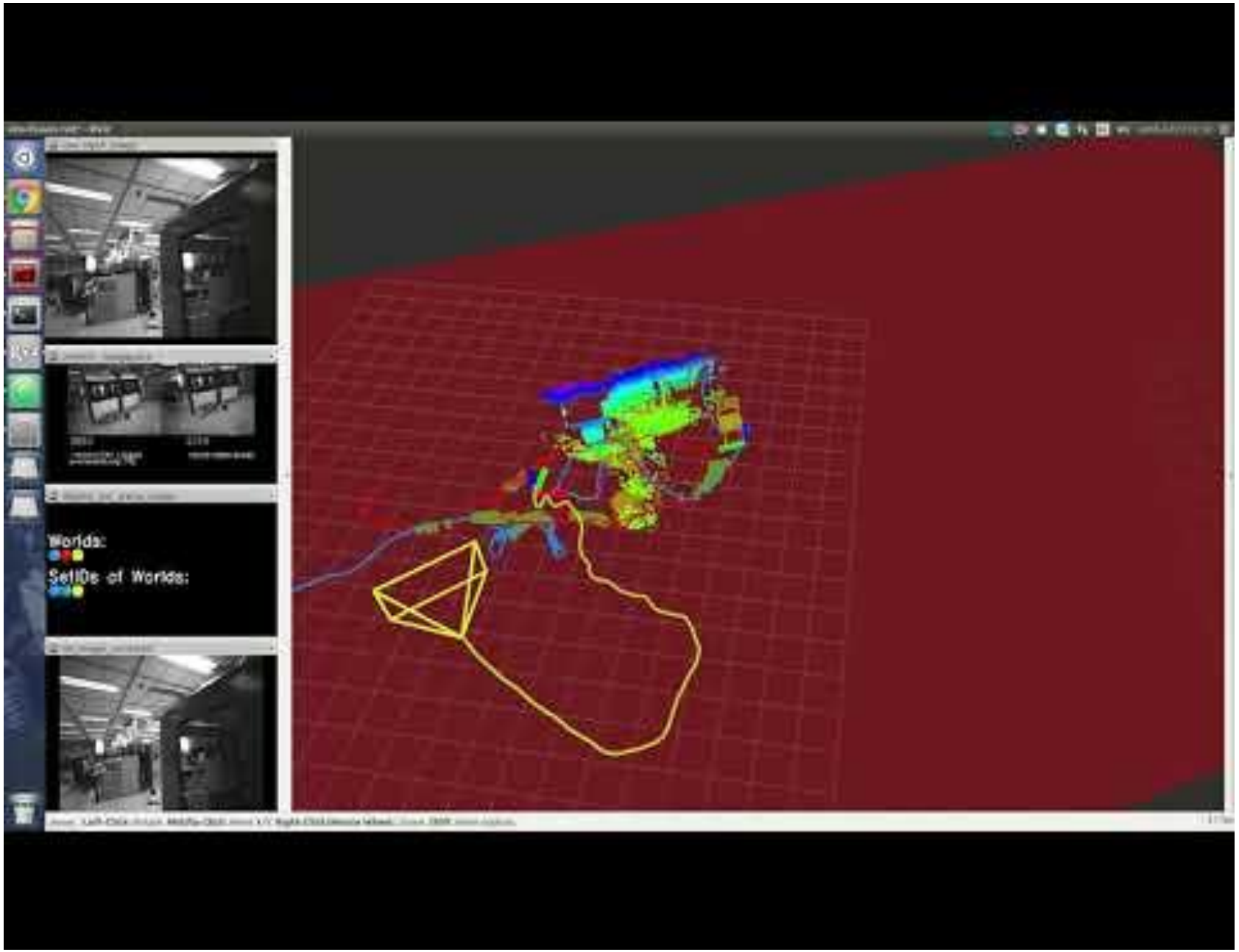


Demo: Merging 20+ frames live



https://www.youtube.com/watch?v=lDzDHZkInos

Demo Augmented Reality (AR) under Kidnap



https://www.youtube.com/watch?v=HL7Nk-fBNqM

Conlcusion: Kidnap Handling

- Use of the 'disjoint-set' data-structure
  - Vastly simplify the handling of multiple co-ordinate system

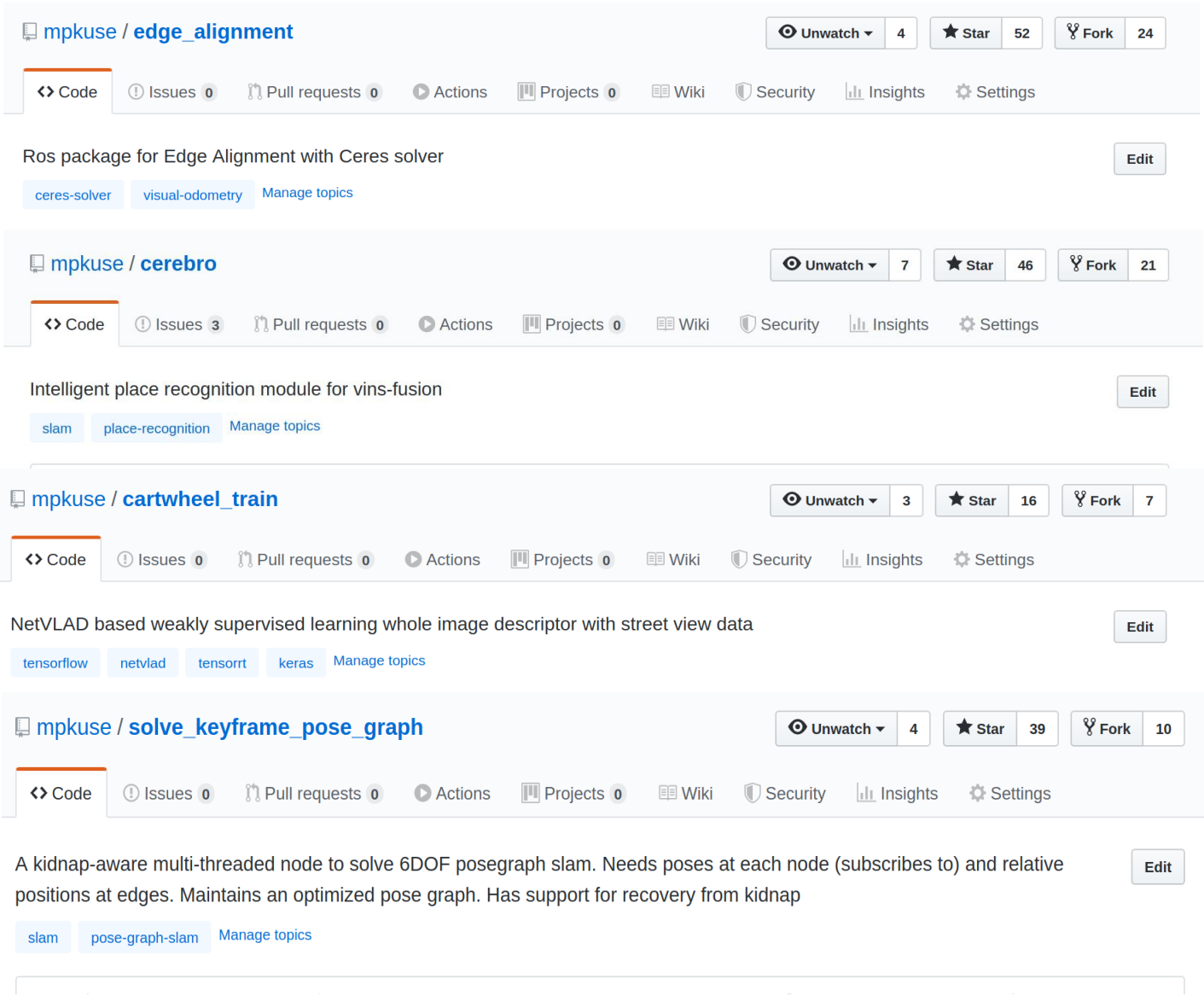
Future Direction

- Real-time semantic scene understanding
- Scene text as a part of place description

Publications

- Journal
  - Manohar Kuse and Shaojie Shen, "**Learning Whole-Image Descriptors for Real-time Loop Detection and Kidnap Recovery under Large Viewpoint Difference**", accepted (Sept 2019) at Robotics and Autonomous Systems (Impact factor 2.167 in 2018).
  - Yonggen Ling, Manohar Kuse, and Shaojie Shen, "**Direct Edge Alignment-Based Visual-Inertial Fusion for Tracking of Aggressive Motions**", accepted *in Springer Autonomous Robots* 2016 (Impact factor 2.706 in 2016).
- Conference
  - Kuse M., Jaiswal SP, Shen S., "**Deep-Mapnets: A Residual Network for 3D Environment Representation**", Accepted in IEEE *Int. Conf. on Image Processing* (ICIP-2017), Beijing, China.
  - Kuse M., Shen S. **Robust Camera Motion Estimation using Direct Edge Alignment and Sub-gradient Method**. Proc. of *International Conference on Robotics and Automation* (ICRA-2016), Stockholm, Sweden.
  - Kuse M., Jaiswal S. P. **Graph Modelling of 3D Geometric Information for Color Consistency of Multiview Images**. *International Conference on Image Processing* (ICIP-2015), Quebec City, Canada.

OpenSource Repositories

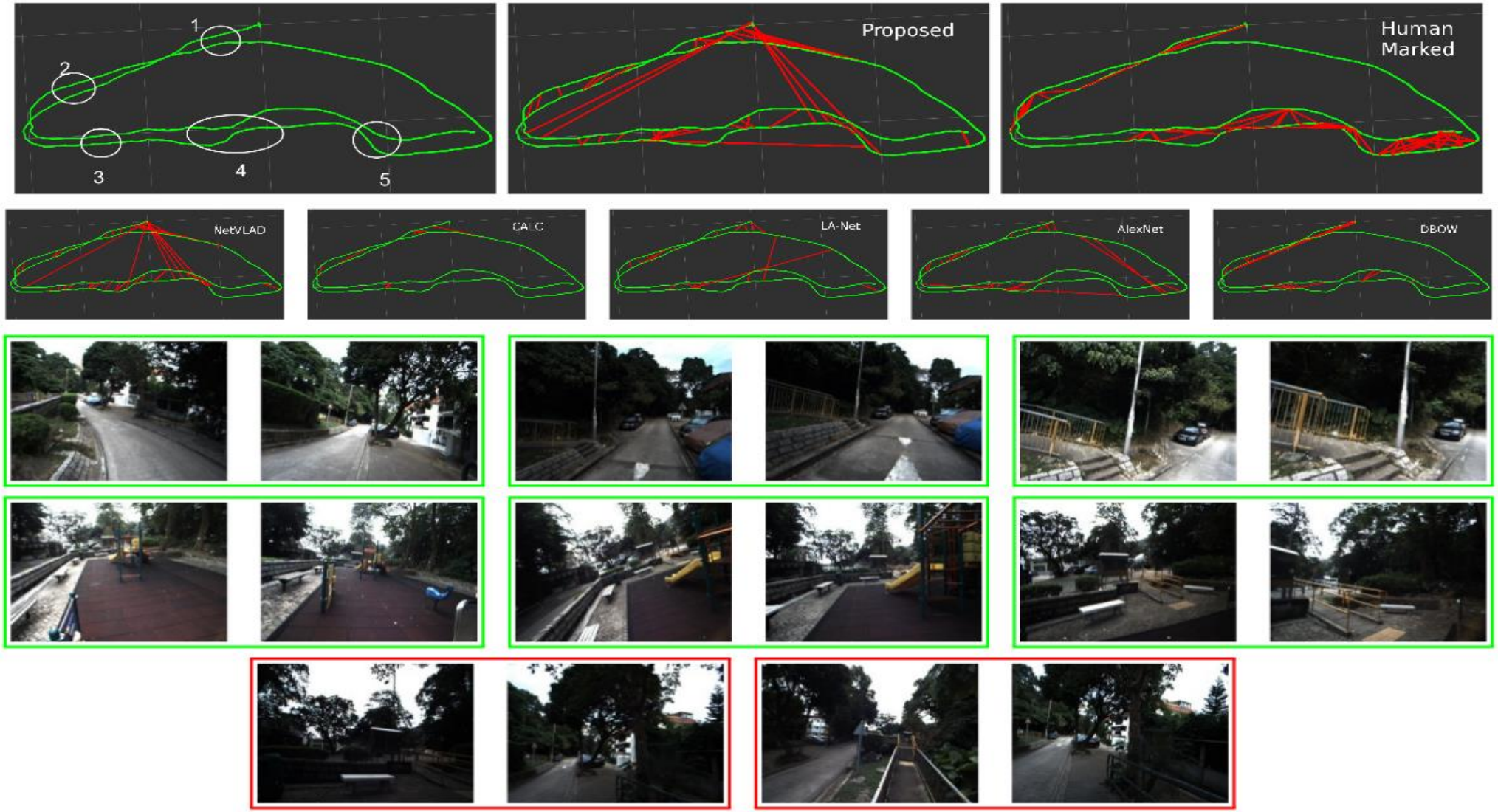


Full system source code is open-sourced and available at: https://github.com/HKUST-Aerial-Robotics/VINS-kidnap


Thanks :)

Additional Slides for Replies to Questions

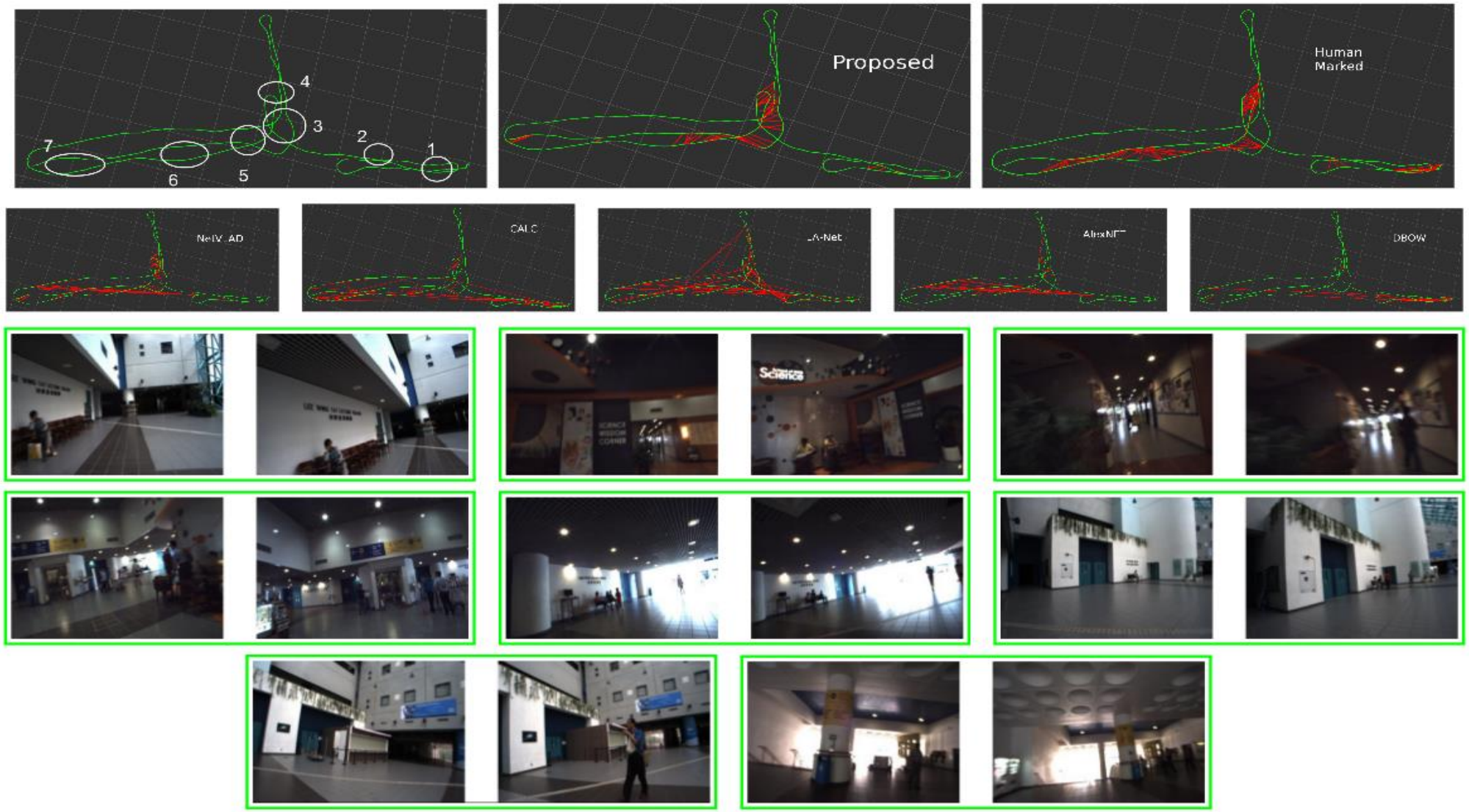
Loop Detections in Real SLAM datasets

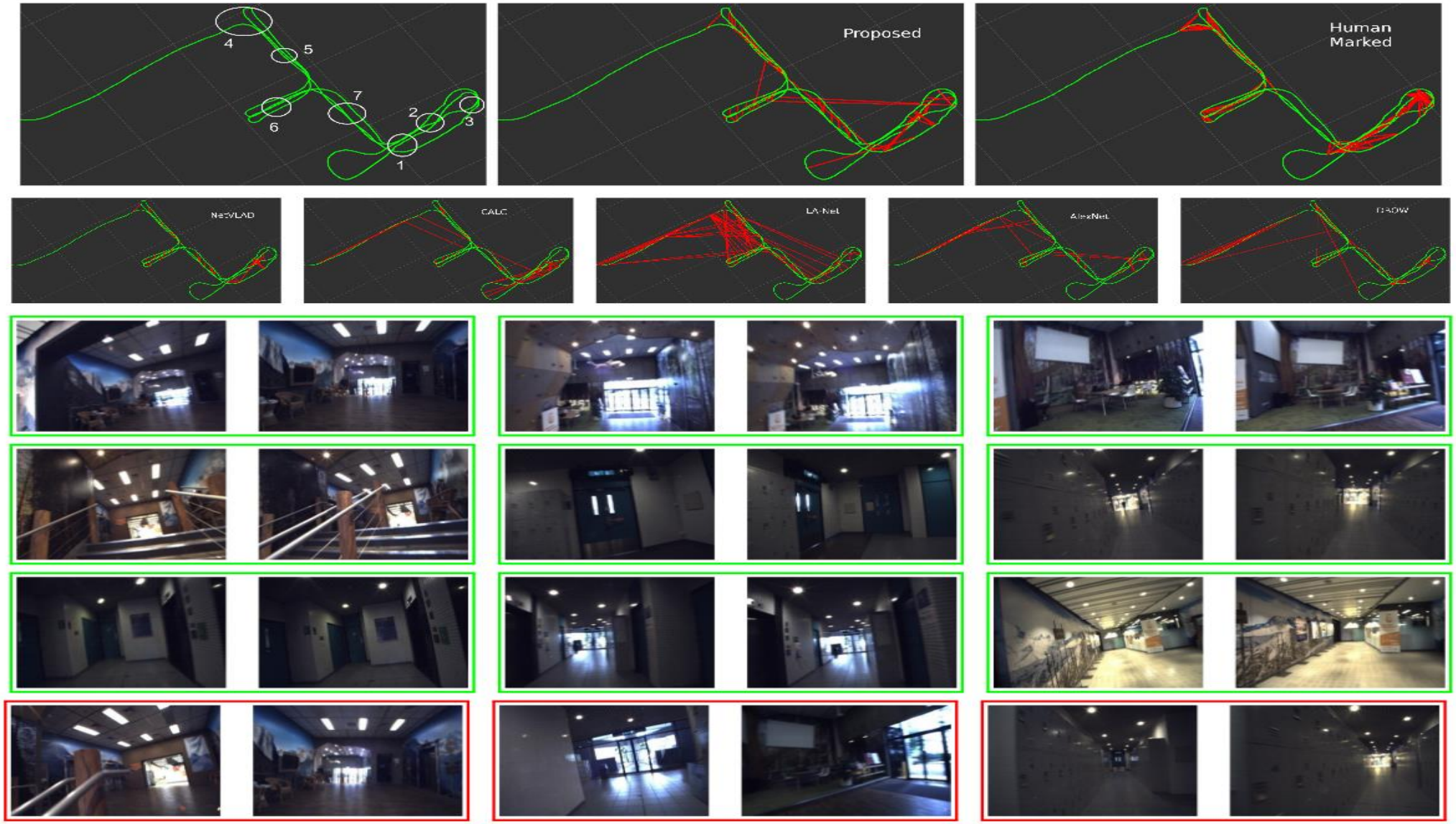


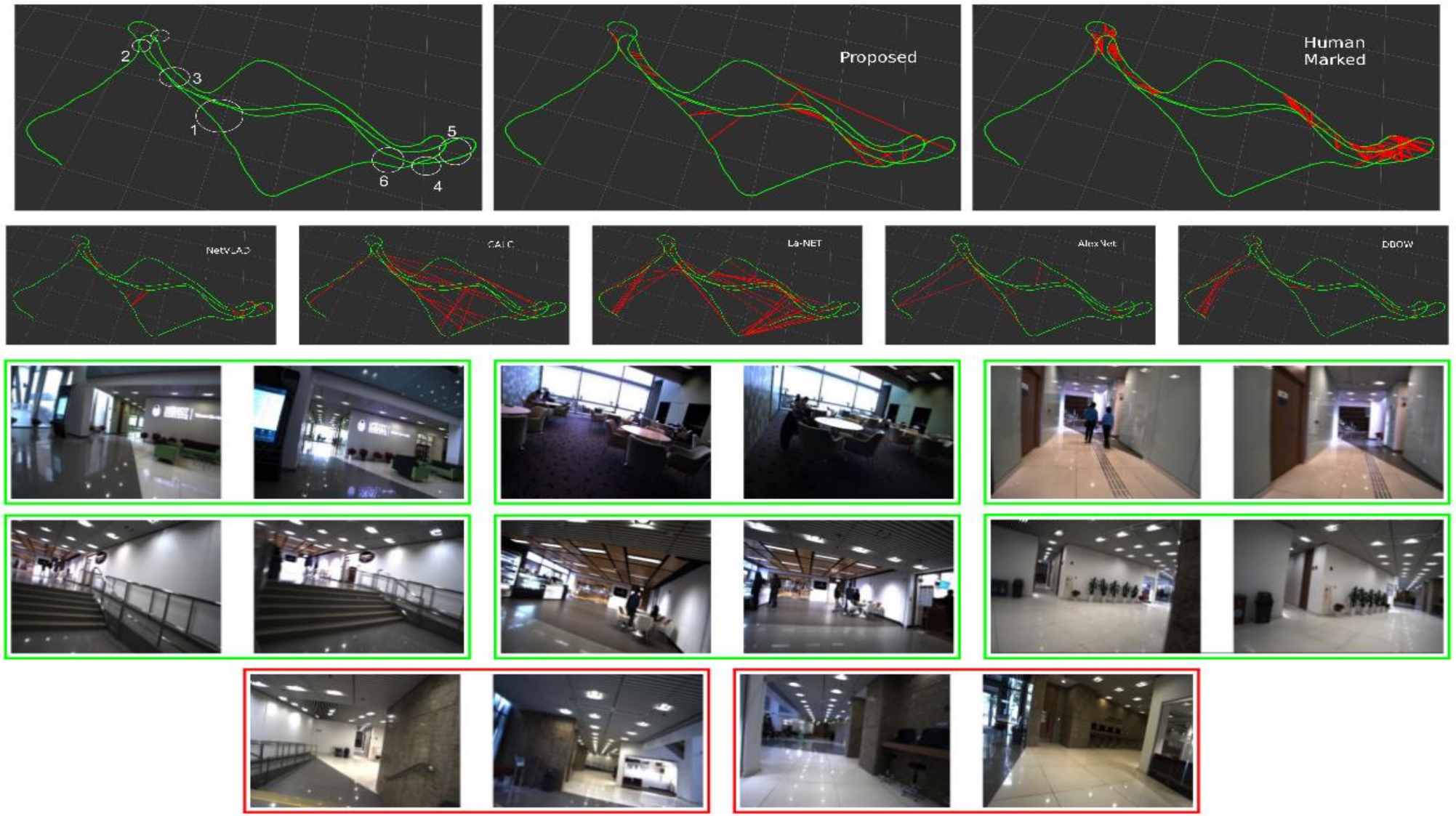
True Positives

False Positive

True Positives




4
5
7
6
2
3
1
Proposed
Human
Marked
NetVLAD
CALC
LA-Net
AlexNet
TRDW
True Positives
False Positive

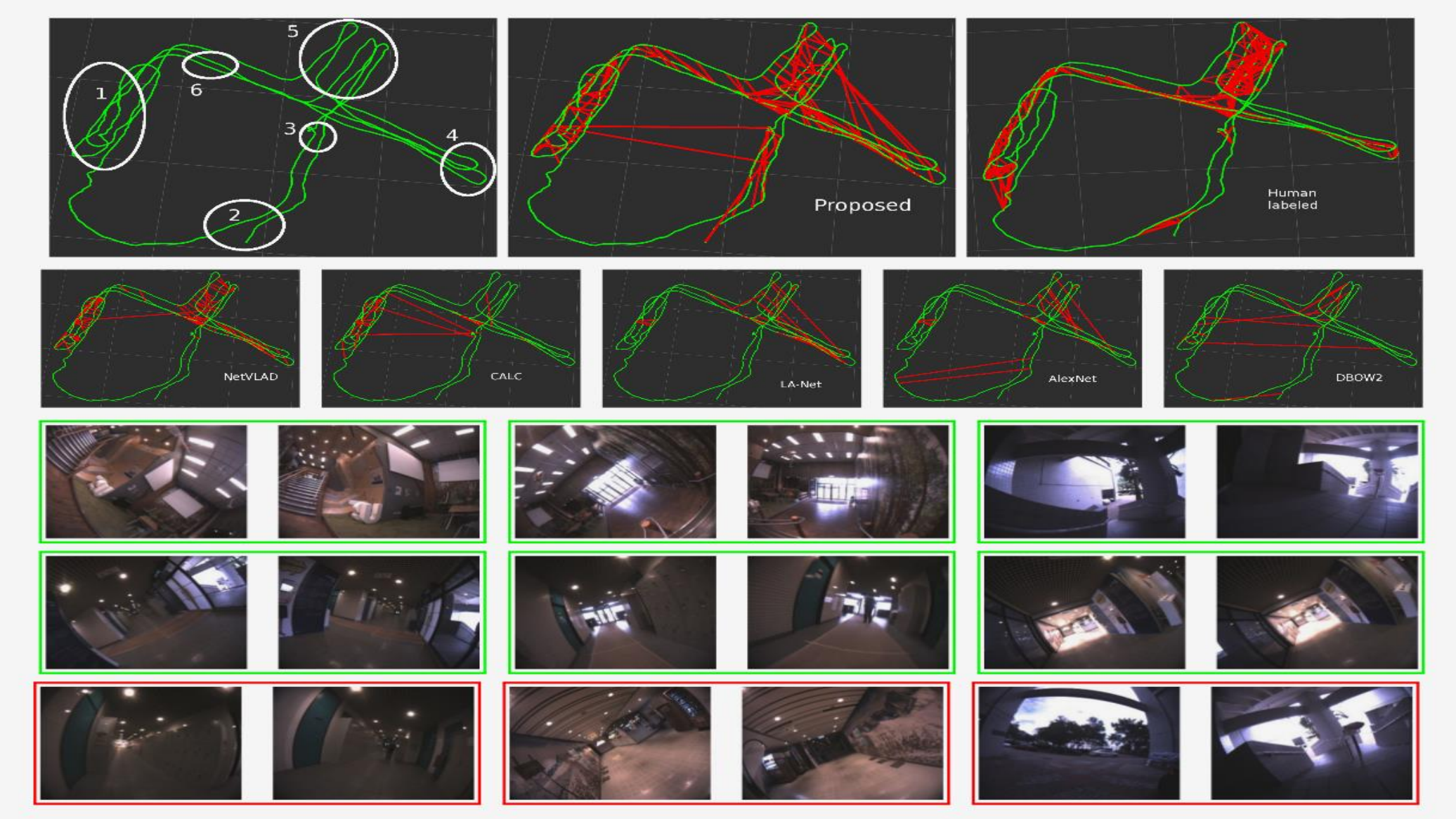


True Positives

False Positive

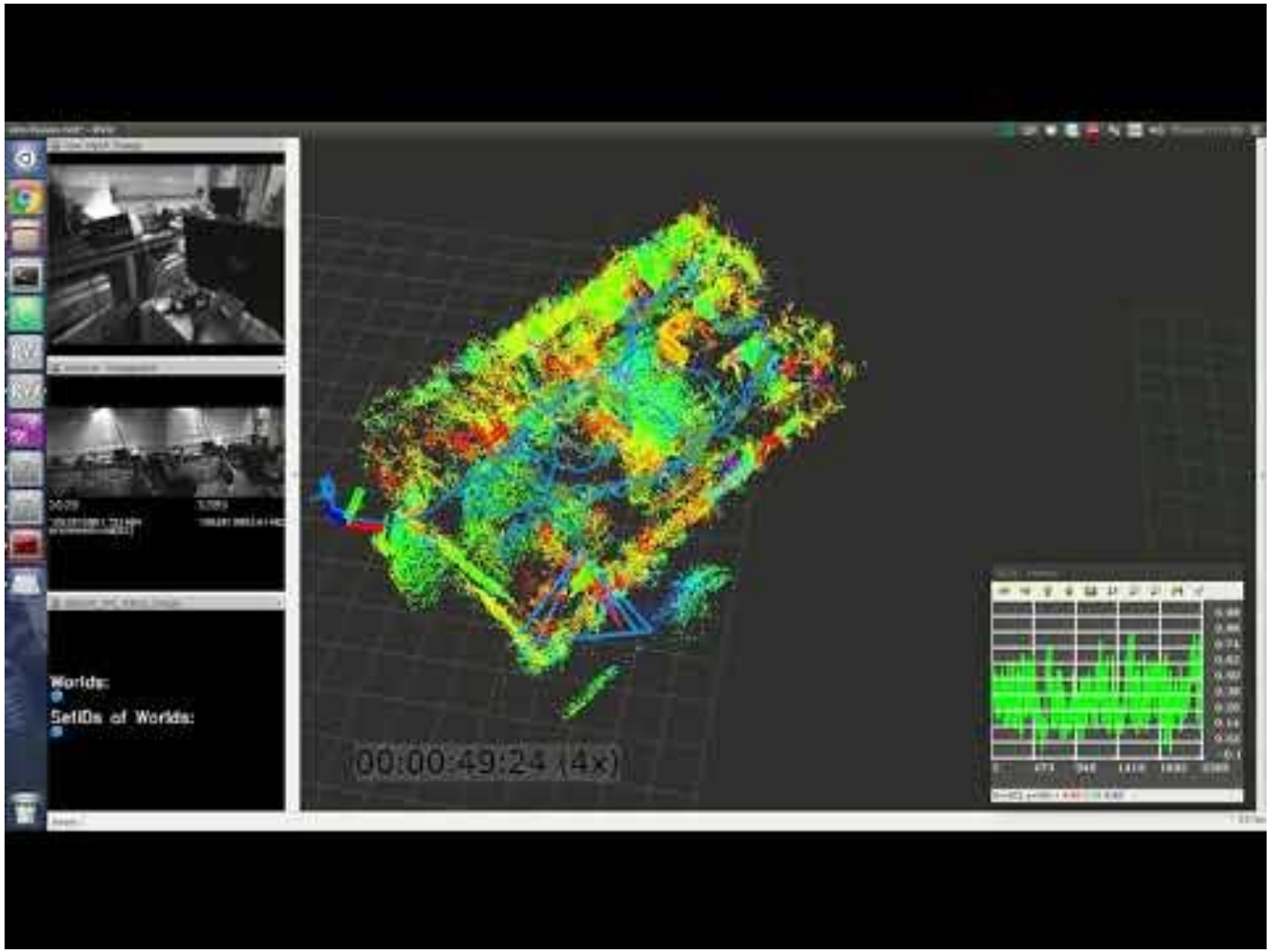
True Positives

False Positive



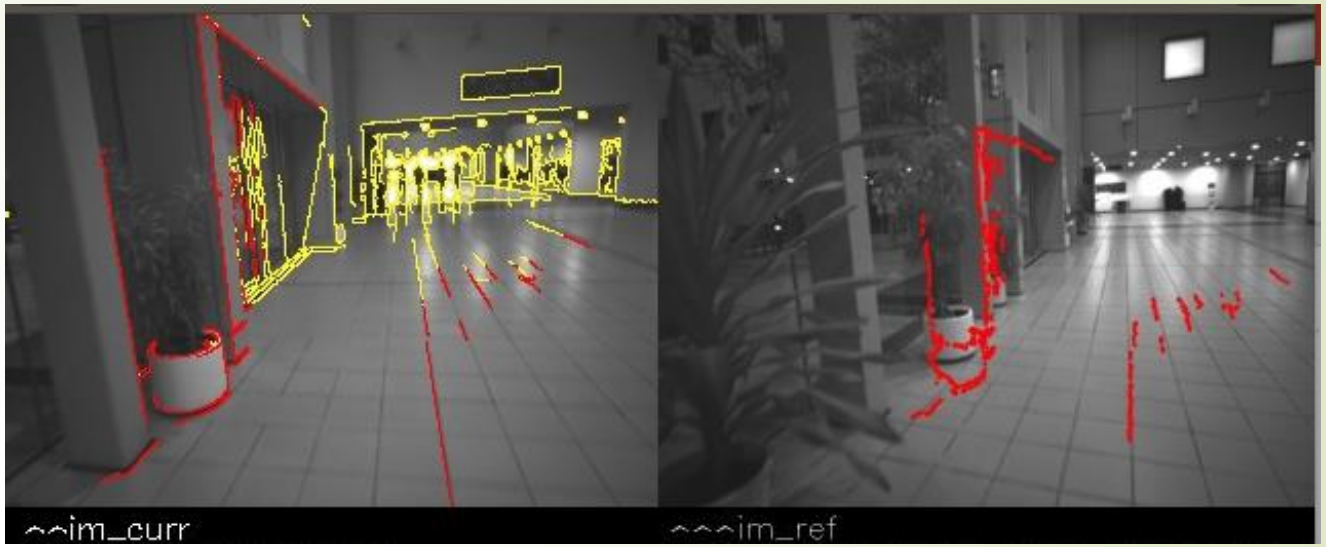
Faster Relocalization from Existing Map

Relocalization from previously built map



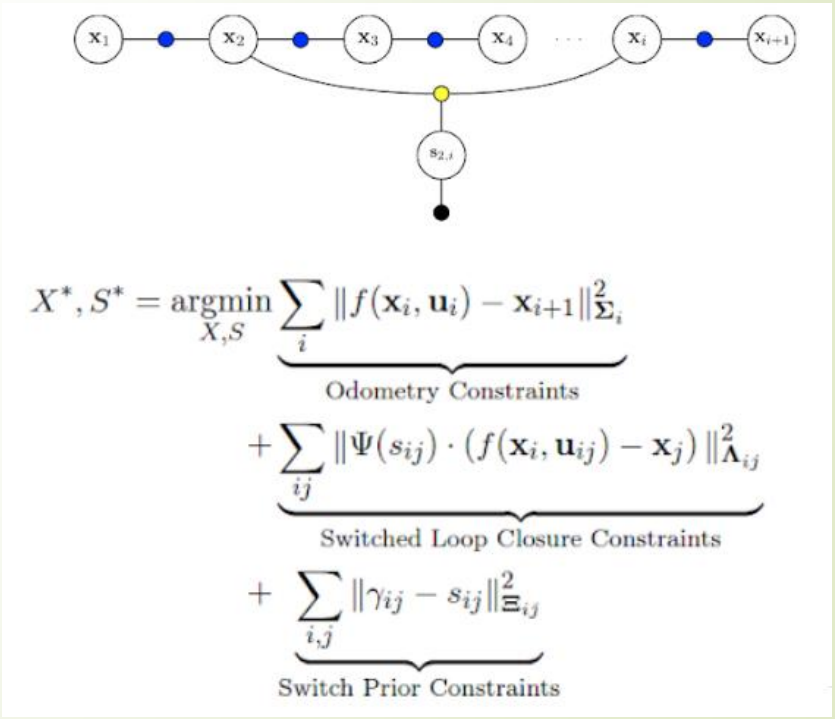
https://www.youtube.com/watch?v=OViEEB3rINo

How to handle "Interesting Failure case"?


im_curr
im_ref


im_ref

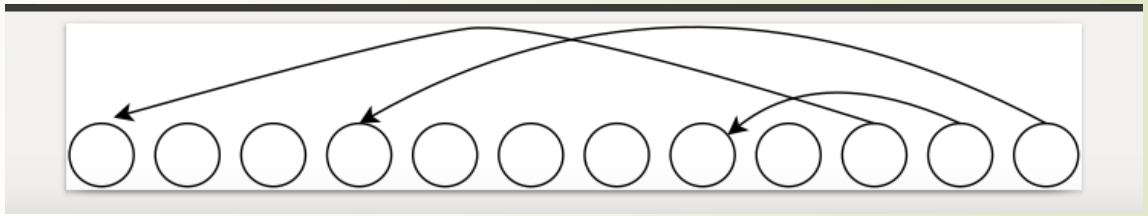
- This pose is passed onto the pose-graph-optimization engine.
- Our pose-graph-optimization uses the switching constraint formulation, originally proposed by [Sunderhauf].

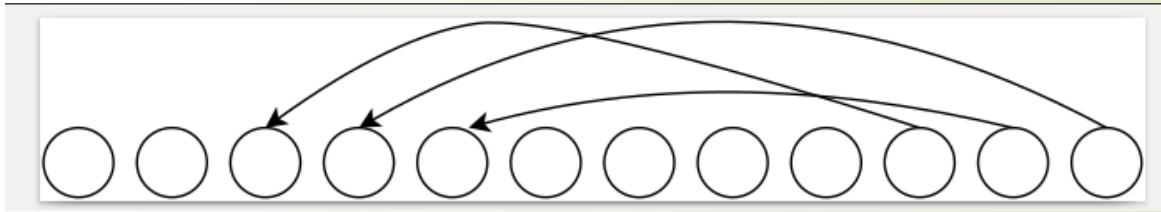


Sünderhauf, Niko, and Peter Protzel. "Switchable constraints for robust pose graph SLAM." Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012.

How do I generate Loop Candidate (Loop Hypothesis)

Coherence Voting





- Grid the sequence
- Maintain a moving window of current few images (say 15)
- Do voting for the grid based on the nearest neighbour from the past
- If a grid gets sufficient votes --> Loop Hypothesis