Techniques for a Failsafe Visual Inertial SLAM System

Ph.D. Defense Presentation
Manohar Kuse <mpkuse@connect.ust.hk>
13th December 2019
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.

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

https://medium.com/jpg-media-lab/apples-arkit-vs-google-s-arccore-e00ff42b0547
https://medium.com/6d-ai/how-is-arcore-better-than-arkit-5223e6b3e79d
Architecture of a SLAM system

Sensor Data Input

- Camera System (Mono, Stereo, RGBD)
- Range Scanner (LIDAR, Laser Range Finder)
- IMU
- GPS, WiFi-based, other global localization systems

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

Tracker

Point Cloud Align

SLAM Frontend aka. Odometry system

Revisit Detection

Relative Pose Computation

Drifting Pose Estimates

Pose Graph Optimization Engine

Mapping System

Drift corrected pose

Sensor Data Input

SLAM Backend
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
Edge Based Visual-Inertial Odometry System

Sensor Data Input
- Camera System (Mono, Stereo, RGBD)
- Range Scanner (LIDAR, Laser Range Finder)
- IMU
- GPS, WiFi-based, other global localization systems

Sensor Fusion
- Tracker
- Point Cloud Align
- Sensor Fusion - Loosely Coupled (KF, EKF)
- Tightly Coupled (Optimization based frameworks)

SLAM Frontend aka. Odometry system

Revisit Detection
- frame\#j <-> frame\#k

Relative Pose Computation
- Drifting Pose Estimates
- Pose Graph Optimization Engine

SLAM Backend
- Mapping System
- Drift corrected pose

Sensor Data Input
CNN Based Visual Place Recognition System

Sensor Data Input

- Camera System (Mono, Stereo, RGBD)
- Range Scanner (LIDAR, Laser Range Finder)
- IMU
- GPS, WiFi-based, other global Localization Systems

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

Tracker
Point Cloud Align

Drift corrected pose

Relative Pose Computation

\[ \mathbf{T}_{jk} \]

Pose Graph Optimization Engine

Mapping System

SLAM Frontend aka. Odometry system

SLAM Backend

Revisit Detection

Sensor Data Input

Drifting Pose Estimates

\[ \mathbf{T}_{jk} \]
Fast and Robust
Large Viewpoint
Pose Computation

Sensor Data Input

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

Relative Pose Computation

Drifting Pose Estimates

Pose Graph Optimization Engine

Mapping System

Drift corrected pose

Sensor Data Input

Revisit Detection

frame#j <-> frame#k
Kidnap-aware Pose Graph Solver

Sensor Data Input
- Camera System (Mono, Stereo, RGBD)
- Range Scanner (LIDAR, Laser Range Finder)
- IMU
- GPS, WiFi-based, other global localization systems

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

Tracker
Point Cloud Align

Relative Pose Computation

Pose Graph Optimization Engine

Mapping System

Revisit Detection
frame\#j <-> frame\#k

Drift corrected pose

SLAM Frontend aka. Odometry 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

Why Edge Align?

- Point-feature based methods
  - Fail in a low texture region (e.g., 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

\[ f(R, T) = \sum_i \min_j D^2(\Pi[R^T(r_P^i - T)], r_{u_j}). \]

\[ \text{minimize } f(R, T) \]
\[ \text{subject to } R \in SO(3) \]
Distance Transform Trick

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

\[
\nu_{e_i}(\xi) = V^{(n)}(\Pi[\tau(\tilde{\Pi}('e_i, Z_r('e_i)), \xi)])
\]

\[
f(\xi) = \sum_{\forall e_i} (\nu_{e_i}(\xi))^2.
\] (3)

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

\[
\xi^* = \arg\min_{\xi} \sum_{\forall e_i} (\nu_{e_i}(\xi))^2
\] (4)

Use Distance Transform as a proxy for, minimum distance between projected point and an edge point.
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.
Alignment with PNP

Edges of im_curr on im_ref

\[ r \hat{T}^c(PNP) \]

Edge Alignment (proposed)

Edges of im_curr on im_ref

\[ r \hat{T}^c(EA) \]
Comparison with Direct Alignment

<table>
<thead>
<tr>
<th>Sequence</th>
<th>D-EA $\delta = 1$</th>
<th>D-EA $\delta = 20$</th>
<th>Kerl et al. [2] $\delta = 1$</th>
<th>Kerl et al. [2] $\delta = 20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>fr2/desk</td>
<td>0.0324</td>
<td>0.1529</td>
<td>0.0333</td>
<td>0.2217</td>
</tr>
<tr>
<td>fr1/desk</td>
<td>0.0289</td>
<td>0.0948</td>
<td>0.0346</td>
<td>0.4286</td>
</tr>
<tr>
<td>fr1/desk2</td>
<td>0.0335</td>
<td>0.1818</td>
<td>0.0343</td>
<td>0.3658</td>
</tr>
<tr>
<td>fr1/door</td>
<td>0.0355</td>
<td>0.1988</td>
<td>0.0330</td>
<td>0.3380</td>
</tr>
<tr>
<td>fr1/room</td>
<td>0.0333</td>
<td>0.2514</td>
<td>0.0307</td>
<td>0.3399</td>
</tr>
<tr>
<td>fr2/desk.with.person</td>
<td>0.0125</td>
<td>0.0994</td>
<td>0.0137</td>
<td>0.1516</td>
</tr>
<tr>
<td>fr3/sitting_hallsphere</td>
<td>0.0208</td>
<td>0.1462</td>
<td>0.0181</td>
<td>0.2599</td>
</tr>
<tr>
<td>fr2/pioneer_slam2</td>
<td>0.0593</td>
<td>0.4447</td>
<td>0.0847</td>
<td>0.4707</td>
</tr>
</tbody>
</table>

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.

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

We use ONLY 25/16 Hz image frequency. The final position drift is 0.08, 0.08, 0.02 meters in X-Y-Z directions. Yaw drift is 0.62 degree.

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

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

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

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

\[ \langle \eta^{(I_q)}, \eta^{(N_j)} \rangle > \langle \eta^{(I_q)}, \eta^{(P_i)} \rangle; \quad \langle \eta^{(I_q)}, \eta^{(N_j)} \rangle < \langle \eta^{(I_q)}, \eta^{(P_i)} \rangle. \]

- 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

\[ \eta \] is the image descriptor, a 1000-Dim float vector
Loss Function / Semi Supervised Learning

Triplet loss, used by NetVLAD’s authors

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

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

The set $\{I_q, \{P_i\}_{i=1,...,m}, \{N_j\}_{j=1,...,n}\}$ is given as, $L_{\text{proposed}}$:

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

I use this because, mine is smoother resulting in stable learning, more separability in +ve and –ve sets.
Effect of Proposed Loss

Triplet loss function, used in original NetVLAD

Green: <▲, △, △>  
Red: <▲, △, △>

Proposed loss function
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

<table>
<thead>
<tr>
<th>CNN Layer</th>
<th># L</th>
<th>D-Size</th>
<th>Model (MB)</th>
<th>Fwd pass memory (MB)</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16_K16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>block5_pool</td>
<td>14.7M</td>
<td>8192</td>
<td>56.19</td>
<td>320x240</td>
<td>640x480</td>
</tr>
<tr>
<td>block4_pool</td>
<td>7.6M</td>
<td>8192</td>
<td>29.19</td>
<td>234.94</td>
<td>767.56</td>
</tr>
<tr>
<td>block3_pool</td>
<td>1.7M</td>
<td>4096</td>
<td>6.65</td>
<td>174.06</td>
<td>725.32</td>
</tr>
<tr>
<td>VGG16_K64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>block5_pool</td>
<td>14.78M</td>
<td>32768</td>
<td>56.38</td>
<td>320x240</td>
<td>640x480</td>
</tr>
<tr>
<td>block4_pool</td>
<td>7.7M</td>
<td>32768</td>
<td>29.38</td>
<td>234.32</td>
<td>711.46</td>
</tr>
<tr>
<td>block3_pool</td>
<td>1.76M</td>
<td>16384</td>
<td>6.75</td>
<td>203.53</td>
<td>696.23</td>
</tr>
<tr>
<td>decoup_K16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pw13</td>
<td>3.2M</td>
<td>8192</td>
<td>12</td>
<td>197.97</td>
<td>792.21</td>
</tr>
<tr>
<td>pw10</td>
<td>1.36M</td>
<td>8192</td>
<td>5</td>
<td>189.1</td>
<td>734.48</td>
</tr>
<tr>
<td>pw7</td>
<td>554K</td>
<td>4096</td>
<td>2</td>
<td>164.9</td>
<td>652.25</td>
</tr>
<tr>
<td>decoup_K16_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pw13</td>
<td>3.5M</td>
<td>512</td>
<td>12</td>
<td>211.58</td>
<td>793.46</td>
</tr>
<tr>
<td>pw10</td>
<td>1.49M</td>
<td>512</td>
<td>5</td>
<td>193.86</td>
<td>739.73</td>
</tr>
<tr>
<td>pw7</td>
<td>686K</td>
<td>512</td>
<td>2</td>
<td>167.67</td>
<td>657.97</td>
</tr>
<tr>
<td>decoup_K64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pw13</td>
<td>3.33M</td>
<td>32768</td>
<td>12</td>
<td>210.98</td>
<td>805.21</td>
</tr>
<tr>
<td>pw10</td>
<td>1.40M</td>
<td>32768</td>
<td>5</td>
<td>180.38</td>
<td>734.58</td>
</tr>
<tr>
<td>pw7</td>
<td>600K</td>
<td>16384</td>
<td>2</td>
<td>162.86</td>
<td>652.35</td>
</tr>
</tbody>
</table>
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 real-world 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 - 1
Example False Positives - 1
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.


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

\[ a_p X_i^{(a_p)} = \pi^{-1}((u_i, v_i), D_{a_p}(u_i, v_i)) \]

\[ b_q X_i^{(b_q)} = \pi^{-1}((u_i', v_i'), D_{b_q}(u_i', v_i')) \]

\[ a_0 X_i^{(a)} = (w T_{a_0})^{-1} w T_{a_p} a_p X_i^{(a_p)} \]

\[ b_0 X_i^{(b)} = (w' T_{b_0})^{-1} w' T_{b_q} b_q X_i^{(b_q)} \]

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

\[
f(X_j, R, t) = \| x_0^{(a)} - R \cdot x_j^{(b)} - t \|_2^2
\]

\[
\min_{R, t, s, j = 1 \ldots K} F(R, t, s) = \sum_{j=1}^{K} s_j^2 f(X_j, R, t) + \lambda (1 - s_j)^2
\]

s.t. \( R \in SO(3) \)

\[
R^{(n-1)}, t^{(n-1)} = \arg\min_{R, t} F(R, t, s^{(n-1)})
\]

\[
s^{(n)} = \arg\min_{s} F(R^{(n-1)}, t^{(n-1)}, s)
\]

Result with GMS-matcher and this method.
4-DOF Pose Refinement with Edge Alignment

- Iterative method
- Optimization variables: yaw, translation in the IMU frame
- Uses initial 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

\[
as_{imu} \hat{T}_{b imu} = \minimize_{as_{imu} \hat{T}_{b imu}} V^a \left( \Pi \left[ (imuT_{cam})^{-1} a_{imu} \hat{T}_{b imu} \right] \right)
\]

\[
aT_b = (imuT_{cam})^{-1} a_{imu} \hat{T}_{b imu} \ \ \ \ imuT_{cam} \ X
\]

- Distance transform of frame-a
- Perspective projection
- Imu-camera calibration
- Edge points of frame-b
- Optimization variable (yaw, translation)
Result of Alignment with Edge Alignment (EA)
6DOF Edge Alignment

PNP (+ RanSAC) with ORB Features (blue)

PNP (+ RanSAC) with Gms-Matcher

4DOF Edge Alignment
6DOF Edge Alignment

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

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)
  • Descriptor computation only. No full system.

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 $T^m_n$
Figure 4: System Overview
Complicated Kidnap Cases
Demo: Merging 20+ frames live

Learning Whole-Image Descriptors for Real-time Loop Detection and Kidnap Recovery under Large Viewpoint Difference

Manohar Kuse and Shaojie Shen

HKUST Aerial Robotics Group

https://github.com/HKUST-Aerial-Robotics/VINS-Kidnap

kidnap_video.mp4

https://www.youtube.com/watch?v=lDzDHZkInos
Demo
Augmented Reality (AR) under Kidnap

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

• 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.
OpenSource Repositories

- **mpkuse / edge_alignment**
  - Ros package for Edge Alignment with Ceres solver
  - [View on GitHub](https://github.com/HKUST-Aerial-Robotics/VINS-kidnap)

- **mpkuse / cerebro**
  - Intelligent place recognition module for vins-fusion
  - [View on GitHub](https://github.com/HKUST-Aerial-Robotics/VINS-kidnap)

- **mpkuse / cartwheel_train**
  - NetVLAD based weakly supervised learning whole image descriptor with street view data
  - [View on GitHub](https://github.com/HKUST-Aerial-Robotics/VINS-kidnap)

- **mpkuse / solve_keyframe_pose_graph**
  - A kidnap-aware multi-threaded node to solve 6DOF pose graph slam. Needs poses at each node (subscribes to) and relative positions at edges. Maintains an optimized pose graph. Has support for recovery from kidnap
  - [View on GitHub](https://github.com/HKUST-Aerial-Robotics/VINS-kidnap)

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

[Image: Aerial view of a coastal city with buildings and a large body of water in the background.]
Additional Slides for Replies to Questions
Loop Detections in Real SLAM datasets
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"?
- This pose is passed onto the pose-graph-optimization engine.
- Our pose-graph-optimization uses the switching constraint formulation, originally proposed by Sünderhauf.

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