OCTOBER 1 - 5, 2023

IEEE/RSJ International Conference on Intelligent Robots and Systems

Exact Point Cloud Downsampling for Fast and Accurate Global Trajectory Optimization

Kenji Koide, Shuji Oishi, Masashi Yokozuka, and Atsuhiko Banno

National Institute of Advanced Industrial Science and Technology, Japan

AIST

Keypoints

Point cloud downsampling technique that accelerates global registration error minimization

- A sampling result gives the **same quadratic registration error** function as that of the original one at the evaluation point
- **Only 29 residuals** are required to be re-linearized at a minimum
- Drastic reduction of **memory consumption (by 99%)** and **processing time** (**by 87%**) while retaining the accuracy

Problem and Proposal

Global Registration Error Minimization

Directly minimizes registration errors over the entire map (i.e., multi-scan registration)

☺ Accurate loop closing for frames with a very small overlap

☺ Accurate propagation of per-point scan matching uncertainty to pose variables

👍 More accurate than the conventional pose graph optimization

☹ Needs to remember point correspondences for each factor

☹ Needs to re-evaluate registration errors every optimization iteration

👎 Large memory consumption and computation cost

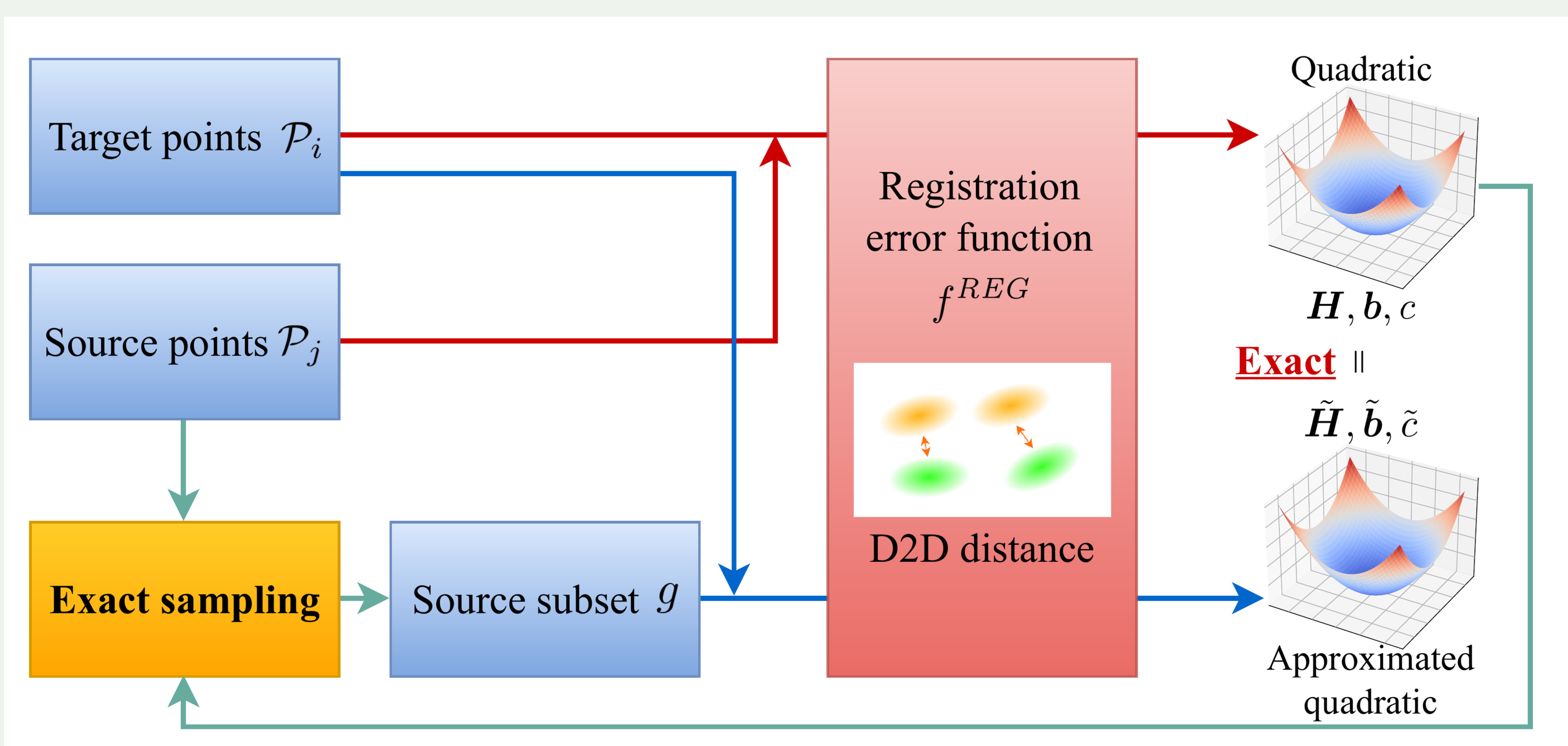
Point Cloud Downsampling

Conventional methods (e.g., random and geometry-aware sampling) introduce approx errors and require a trade-off between processing cost (# of points) and accuracy

[Reijgwart, RA-L 2020] [Wang, RA-L 2022] [Yokozuka, ICRA2021]

Proposal

To reduce points as few as possible while retaining the result as accurate as possible, we find a subset of input points that yields **the same quadratic error function** as that of the original set via **efficient coreset extraction**



Exact Downsampling

Proposed Approach to Downsampling

Gauss-Newton optimization models an error function in the quadratic form

$$f^{REG}(\mathcal{P}_i, \mathcal{P}_j, \check{\boldsymbol{T}} \boxplus \Delta \boldsymbol{x}) \approx \Delta \boldsymbol{x}^T \boldsymbol{H} \Delta \boldsymbol{x} + 2\boldsymbol{b}^T \Delta \boldsymbol{x} + c,$$

Error function Source / Target points Information matrix / vector Constant

Exact downsampling selects a subset of points that yields the same quadratic function

$$f^{DOWN}(f^{REG}, \mathcal{P}_i, \mathcal{P}_j, \check{\boldsymbol{T}}_i, \check{\boldsymbol{T}}_j) = (\boldsymbol{w}_{ij}, \mathcal{G}_{ij}),$$

Evaluation point Weights / Residual indices

$$\text{s.t.} \quad \boldsymbol{H} = \tilde{\boldsymbol{H}}, \quad \boldsymbol{b} = \tilde{\boldsymbol{b}}, \quad c = \tilde{c},$$

Quadratic function parameters calculated using original set / extracted subset

During optimization, f^{REG} is re-relinearized using the selected subset (i.e., *coreset*)

Coreset Extraction Algorithm

Hessian matrix $\boldsymbol{H}$ can be represented with at most $D^2 + 1$ rows of Jacobian matrix $\boldsymbol{J}$ without approximation via **Caratheodory set extraction**

$$\boldsymbol{H} = \boldsymbol{J}^T \boldsymbol{J} = \sum \boldsymbol{a}_k^T \boldsymbol{a}_k \qquad \frac{1}{N} \sum \text{flatten}(\boldsymbol{a}_k^T \boldsymbol{a}_k) = \frac{1}{N} \sum \boldsymbol{h}_k = \boldsymbol{\mu}^h$$

k-th row of J

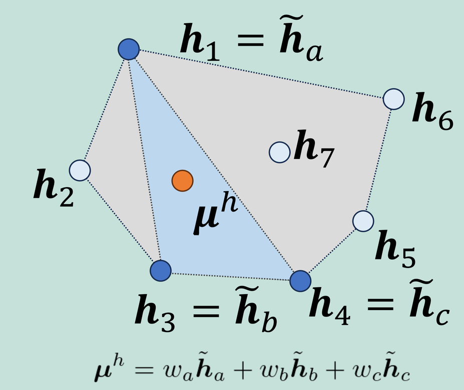
Because $\boldsymbol{\mu}^h$ is in the convex hull of $\boldsymbol{h}_k$, $\boldsymbol{H}$ can be represented by a subset $\tilde{\boldsymbol{J}}$ with at most $D^2 + 1$ rows in $\boldsymbol{J}$ **(Caratheodory set)**

$$\boldsymbol{H} = \tilde{\boldsymbol{J}}^T \tilde{\boldsymbol{W}} \tilde{\boldsymbol{J}}$$

Selected rows (at most $D^2 + 1$ rows) Weights

The classic Caratheodory set extraction algorithm with SVD on hyper-plane takes $O(N^2 D^2)$ time

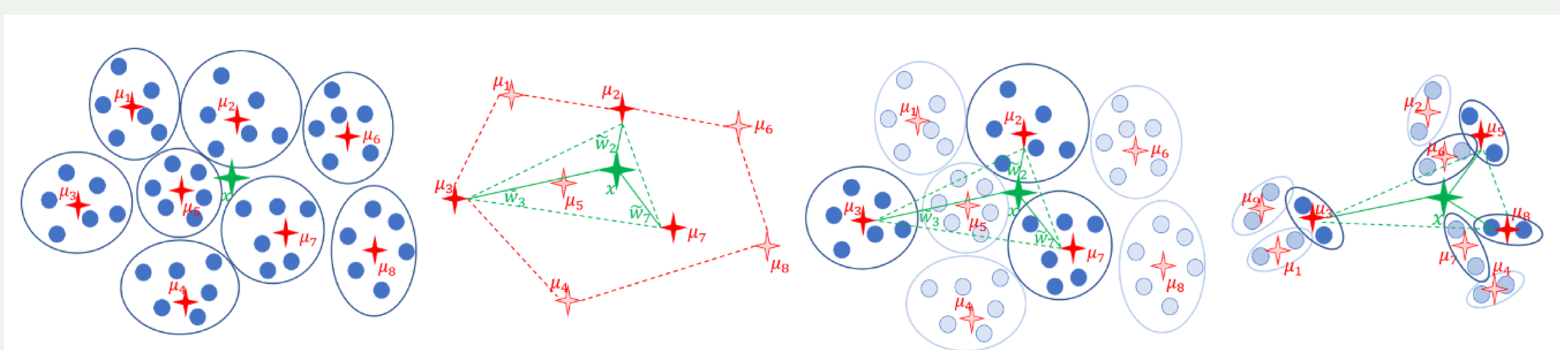
Caratheodory theorem



Every point in the convex hull of a set $S \in \mathbb{R}^N$ can be represented as a combination of $(N+1)$ points in S

We use an efficient Caraheodory set extraction algorithm [Maalouf, 2019], which takes $O(ND)$ time, and extend it to:

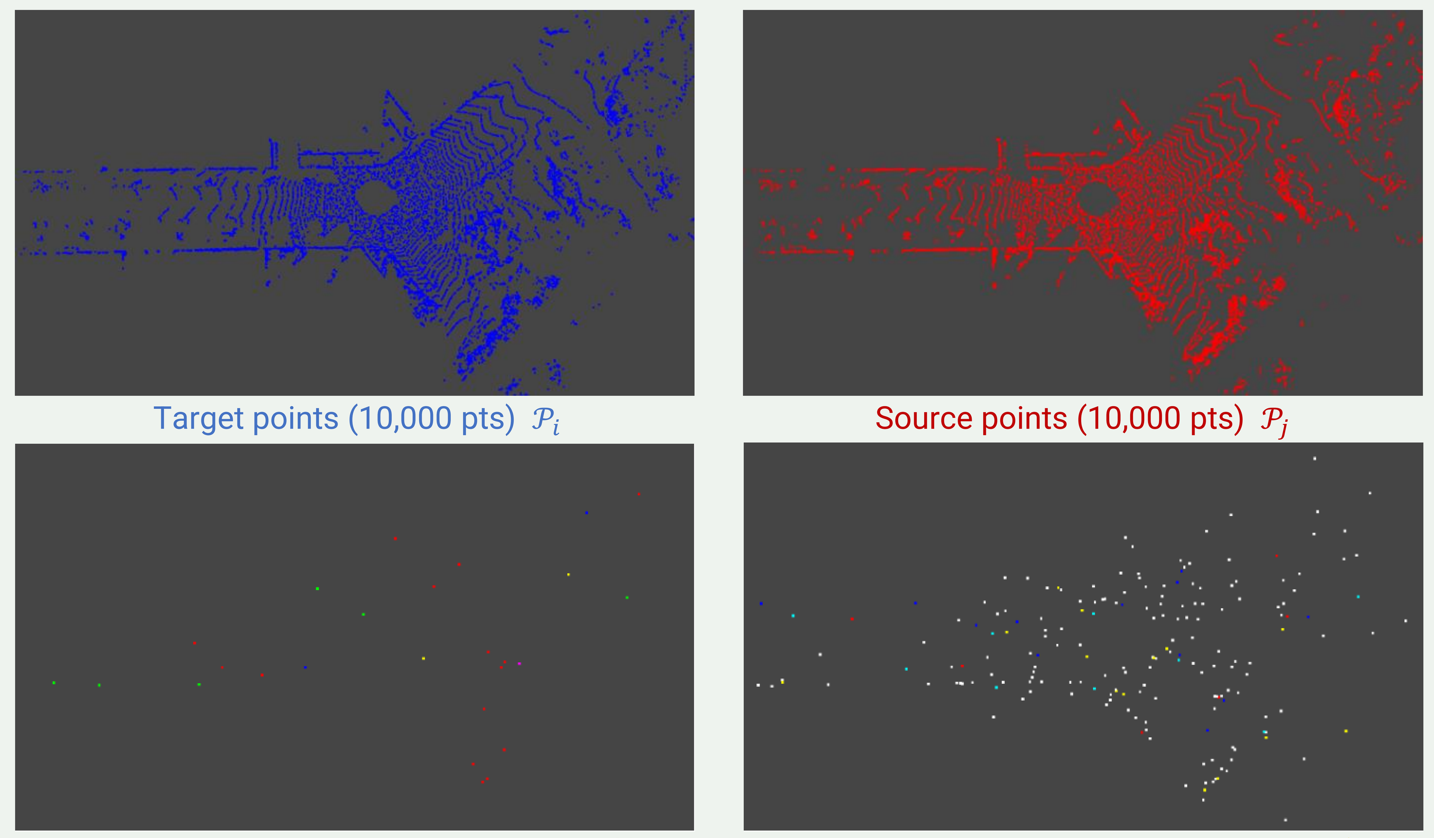
- reconstruct $\boldsymbol{b}$ and c in addition to $\boldsymbol{H}$ to recover the original quadratic function
- control accuracy-vs-speed by changing the number of data to be selected
- speed up the algorithm by eliminating non-upper-triangular elements of $\boldsymbol{H}$



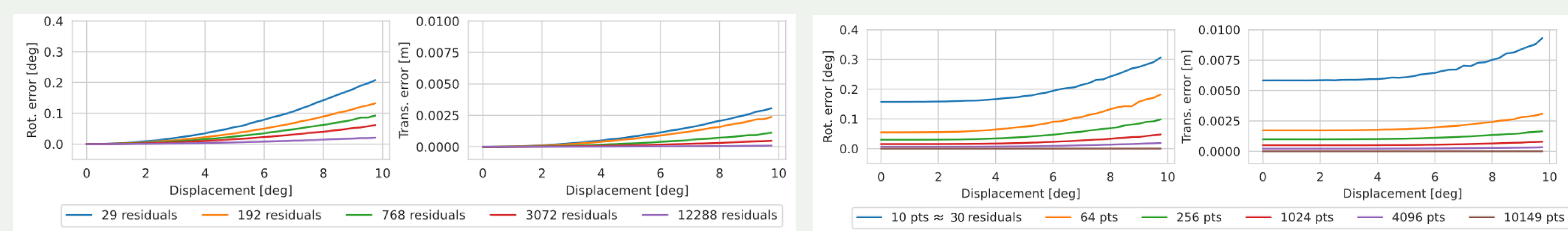
[Maalouf, NeurIPS 2019]

Experimental Results

Approximation Accuracy



$\mathcal{P}_j$, $\tilde{\mathcal{P}}_j^{29}$, $\tilde{\mathcal{P}}_j^{512}$ all yield the **same quadratic registration error function** for $\mathcal{P}_i$ at the evaluation point



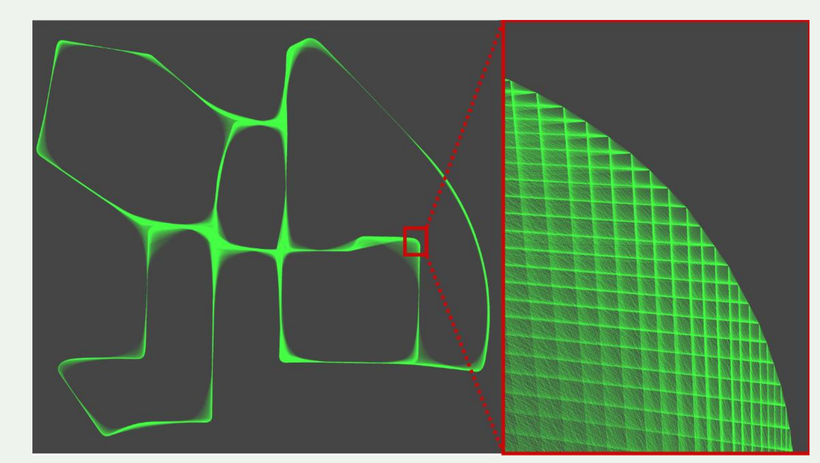
(b) Exact sampling (a) Random sampling

Better **non-linearity approximation accuracy under displacements**

Application to Global Trajectory Optimization

Evaluation on KITTI dataset

- 4,540 pose variables
- 585,417 registration error factors
- Re-linearize registration errors of 10,000 points for each factor at every optimization iteration



Optimization time and memory consumption

- No downsampling **13 hours / 22 GB**
- Exact downsampling **1.7 hours / 0.25 GB**

Method	Sequence 00	02	05	08
Pose graph optimization (Identity)	1.6257 ± 0.7529	23.9856 ± 8.2752	1.5149 ± 0.6760	9.3636 ± 3.2890
Pose graph optimization (Hessian)	1.3777 ± 0.6051	9.3406 ± 3.1490	1.6249 ± 0.8489	5.0532 ± 2.5991
Pose graph optimization (Hessian + Dense)	1.1846 ± 0.5625	9.3393 ± 3.1486	1.6240 ± 0.8488	5.0520 ± 2.5993
Registration error minimization + Exact sampling (M=29)	0.9553 ± 0.4650	8.9679 ± 3.0856	0.2917 ± 0.1060	4.4394 ± 2.5294

Large accuracy gain compared to pose graph optimization

Method	ATE [m]	Optimization time [h]
Exact sampling (M=29)	0.9553 ± 0.4650	1.67
Exact sampling (M=256)	0.9549 ± 0.4646	1.88
Exact sampling (M=1024)	0.9549 ± 0.4647	2.36
Original points (10,000 points)	0.9549 ± 0.4647	13.21

Small accuracy drop with downsampling

More Information

https://github.com/koide3/caratheodory2

https://staff.aist.go.jp/k.koide

https://twitter.com/k_koide3