# TurboReg: TurboClique for Robust and Efficient Point Cloud Registration

Shaocheng Yan  Pengcheng Shi*  Zhenjun Zhao  Kaixin Wang  Kuang Cao  Ji Wu  Jiayuan Li*

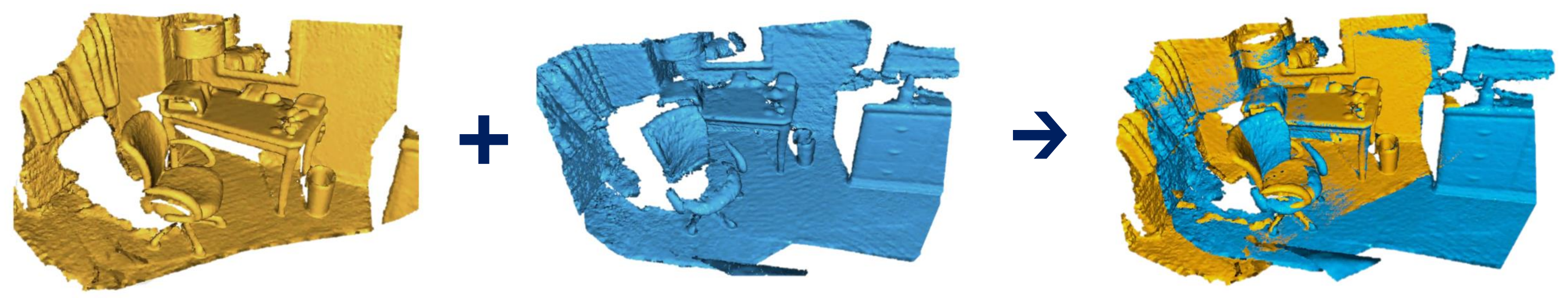
https://github.com/Laka-3DV/TurboReg

Training-free  SOTA  60+FPS  Cross-platform

ICCV OCT 19-23, 2025 HONOLULU HAWAII

## Motivation & Problem

### Point Cloud Registration (PCR)

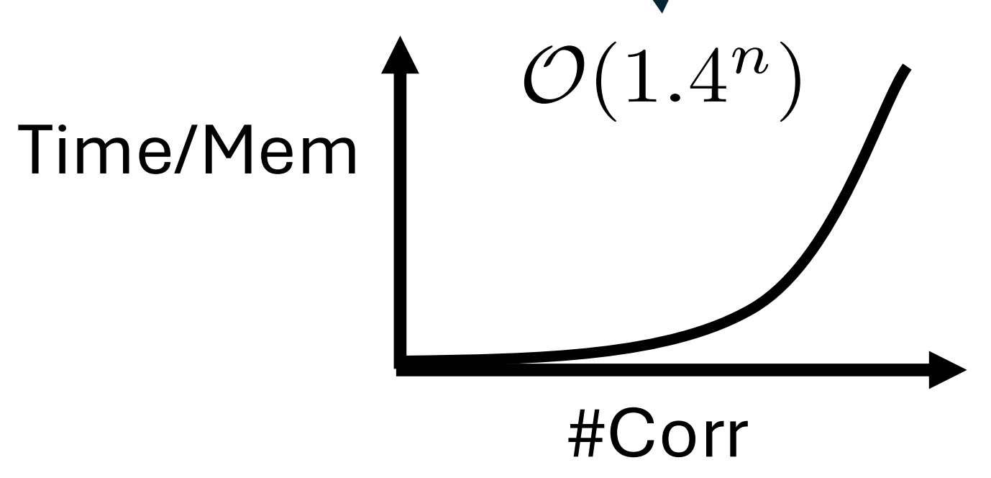


Align two 3D scans captured from different viewpoints.

### Challenges

3DMAC (CVPR-23 Student Best Paper)

Core: Enumerate maximal cliques on a compatibility graph.



✓ Accuracy – SOTA

✗ Efficiency

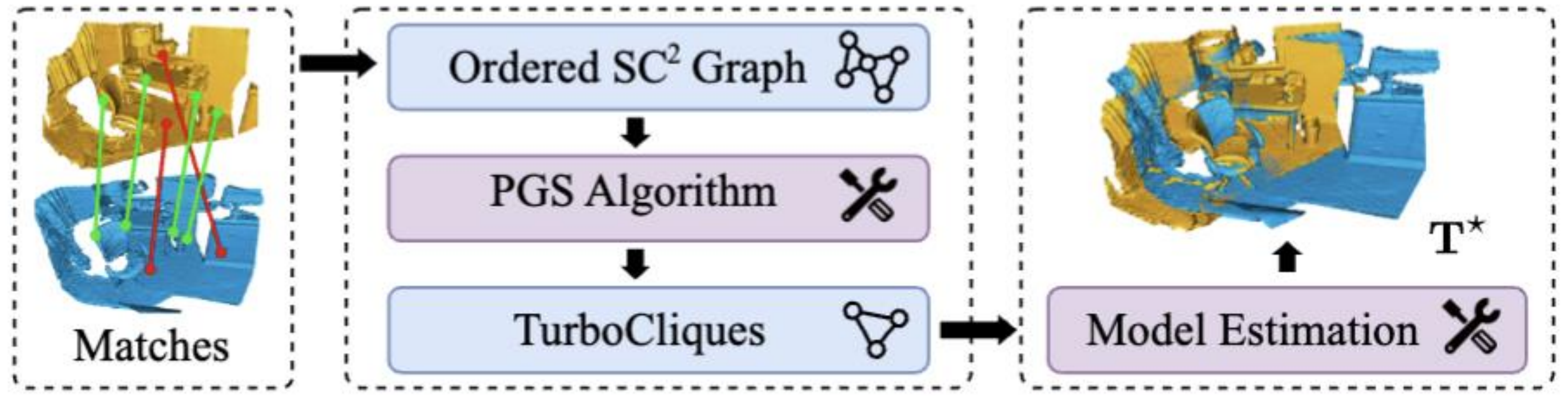
- Slow (< 1FPS)
- Out of memory

***"Can we make Point Cloud Registration both accurate and real-time?"***

Core Idea: No more Maximal Cliques.

- A lightweight, PCR-tailored **TurboClique.**
- A parallel search scheme — **PGS.**

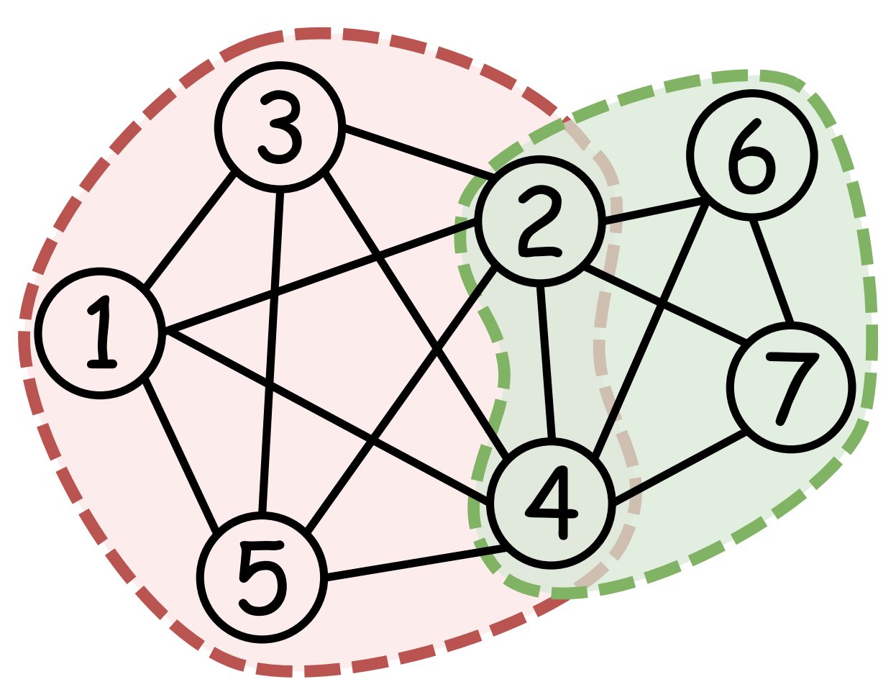
## Method Overview



**Construct** O2Graph → **search** TurboCliques → **estimate** transformations → **select** the one with the highest inlier ratio.

## Key Techniques [0] ← TurboClique

### [0] A Lightweight Yet Stable Clique for PCR

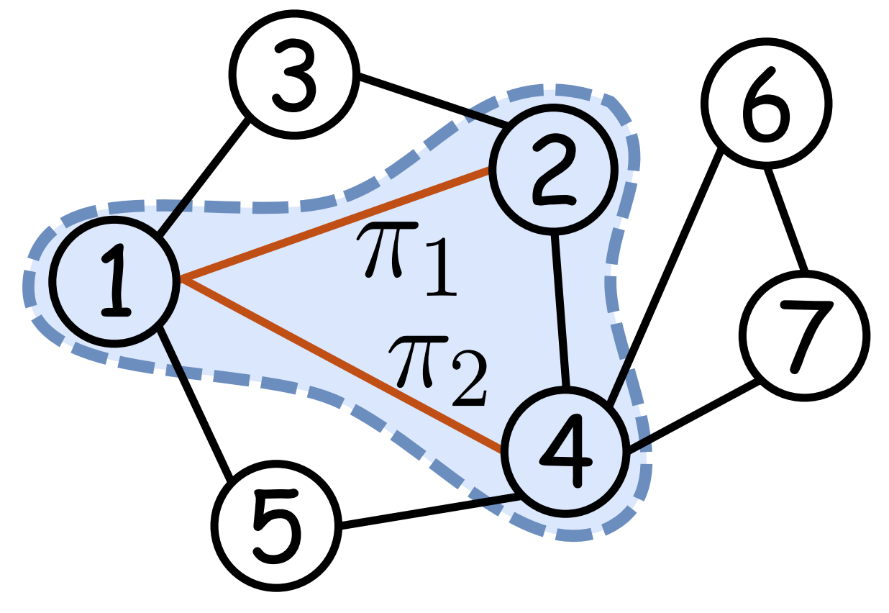


**Maximal Clique**

✓ Stable

✗ Large → Expensive

↓



**TurboClique**

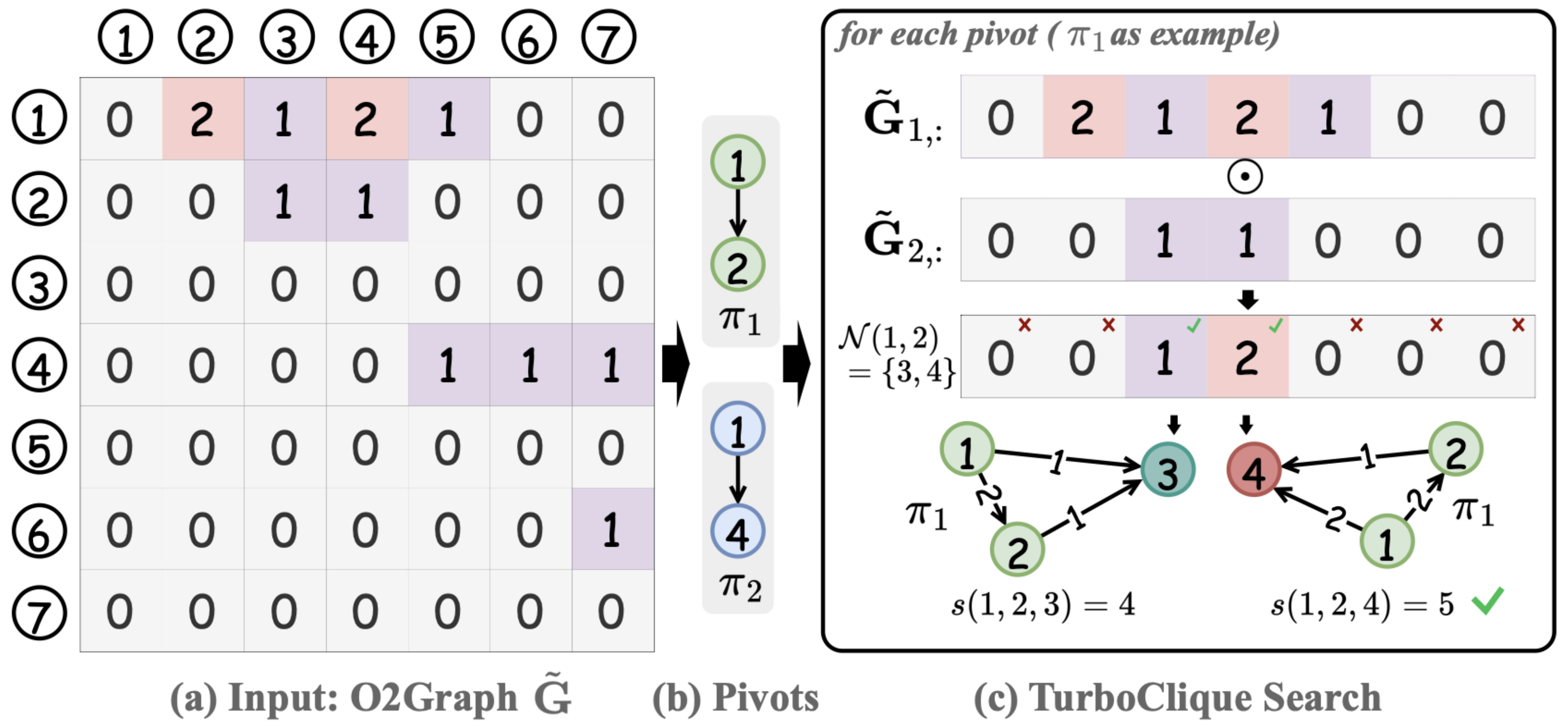
✓ 3-clique → Lightweight

✓ Strict Compatibility → Stable

## Key Techniques [1] ← PGS

### [1] Full Parallelism Search for TurboCliques

- **Core:** Search TurboCliques around pivots defined by high SC2-score edges.
- **Complexity**
  - O(N) on CPU
  - **O(1)** on GPU → **60+FPS** → Speedup **×200+**



|  | ① | ② | ③ | ④ | ⑤ | ⑥ | ⑦ |
|---|---|---|---|---|---|---|---|
| ① | 0 | 2 | 1 | 2 | 1 | 0 | 0 |
| ② | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| ③ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ④ | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| ⑤ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⑥ | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ⑦ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$\pi_1$
$\pi_2$
for each pivot ( $\pi_1$ as example)
$\tilde{\mathbf{G}}_{1,:}$ 0 2 1 2 1 0 0
$\odot$
$\tilde{\mathbf{G}}_{2,:}$ 0 0 1 1 0 0 0
$\mathcal{N}(1,2) = \{3,4\}$ 0 0 1 2 0 0 0
$s(1,2,3) = 4$
$s(1,2,4) = 5$ ✓

(a) Input: O2Graph $\tilde{\mathbf{G}}$ (b) Pivots (c) TurboClique Search

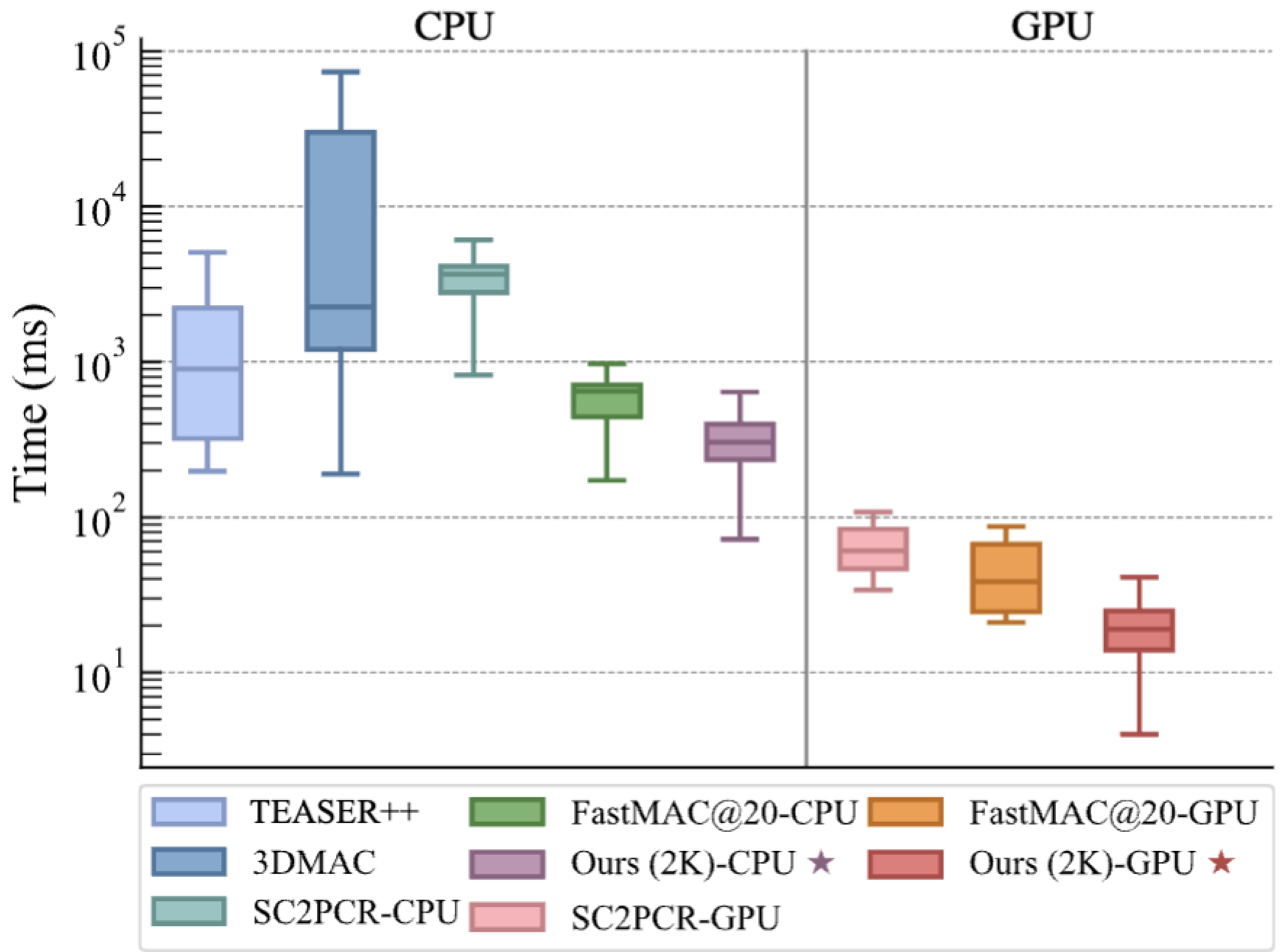
Select top-$K_1$ pivots by SC2 scores → Search all 3-cliques around each pivot → Retain top-$K_2$ TurboCliques ($K_1 \times K_2$ total).

## Results

### Accuracy on 3DMatch Dataset

| Methods | FPFH |  |  | FCGF |  |  | FPS |  |
|---|---|---|---|---|---|---|---|---|
|  | RR(%) | RE(°) | TE(cm) | RR(%) | RE(°) | TE(cm) | CPU | GPU |
| *i) Deep Learned* |  |  |  |  |  |  |  |  |
| DGR [12] | 32.84 | 2.45 | 7.53 | 88.85 | 2.28 | 7.02 | 0.43 | 0.91 |
| PointDSC [5] | 72.95 | 2.18 | **6.45** | 91.87 | 2.10 | 6.54 | 0.20 | 10.74 |
| VBReg [25] | 82.75 | 2.14 | 6.77 | **93.16** | 2.33 | 6.68 | 0.06 | 7.62 |
| *ii) Learning Free* |  |  |  |  |  |  |  |  |
| RANSAC-1M [17] | 64.20 | 4.05 | 11.35 | 88.42 | 3.05 | 9.42 | 0.05 | - |
| RANSAC-4M [17] | 66.10 | 3.95 | 11.03 | 91.44 | 2.69 | 8.38 | 0.01 | - |
| GC-RANSAC [7] | 67.65 | 2.33 | 6.87 | 92.05 | 2.33 | 7.11 | 1.01 | - |
| TEASER++ [51] | 75.48 | 2.48 | 7.31 | 85.77 | 2.73 | 8.66 | 1.12 | - |
| CG-SAC [40] | 78.00 | 2.40 | 6.89 | 87.52 | 2.42 | 7.66 | - | - |
| SC$^2$-PCR [10] | 83.73 | 2.18 | 6.70 | **93.16** | 2.09 | 6.51 | 0.27 | 15.84 |
| 3DMAC [61] | 83.92 | **2.11** | 6.80 | 92.79 | 2.18 | 6.89 | 0.31 | - |
| FastMAC@50 [62] | 82.87 | 2.15 | 6.73 | 92.67 | **2.00** | 6.47 | 1.35 | 4.33 |
| FastMAC@20 [62] | 80.71 | 2.17 | 6.81 | 92.30 | 2.02 | 6.52 | 1.56 | 26.32 |
| Ours (1K) | 83.92 | 2.17 | 6.79 | **93.59** | 2.03 | **6.40** | **2.73** | **61.25** |
| Ours (2K) | **84.10** | 2.19 | 6.81 | **93.59** | 2.04 | 6.42 | 2.46 | 54.04 |

### Runtime Comparison



## Summary & Takeaway

- **Analyze.** Existing graph-based PCR suffers from exponential complexity and poor parallelism.
- **Redesign.** TurboClique & PGS bring lightweight, fully parallel graph reasoning tailored for PCR.
- **Achieve.** Real-time (60+ FPS), training-free registration with SOTA accuracy.