Benchmarking Robust Estimation Methods

Dmytro Mishkin, Czech Technical University in Prague
Deep matchers: RANSAC sucks!

Is that so?
We would like to benchmark the following problems

• Fundamental & essential matrices
• Homography
• PnP
• Rigid point cloud registration
• Multi-instance estimation
• Multi-model estimation
We would like to benchmark the following problems

• Fundamental & essential matrices
• Homography
• PnP
• Rigid point cloud registration
We have the results for

1 day before tutorial…

Fundamental & Essential matrices

Homography

Rigid point cloud registration
Why the robust estimation benchmark is hard?

• There are few datasets for RANSAC-like problems and existing ones are small
  • That is why one cannot train/evaluate deep learning method on existing datasets

• There is no common evaluation protocol
  • What is the metric? Inlier ratio? Number of “correct” inliers? % of successes?

• There is no training/validation/test split.
  • Common practice - tune the methods on the same dataset, or even image pair, it is tested on.

• There are often no fixed set of correspondences to use.
  • One can easily make almost any method to outperform the rest because of differences in preprocessing
RANSAC 2020: benchmark collection

- We haven’t been able to solve all the mentioned problem in the current evaluation.
- But we believe that we did a good step in the right direction.
- Input data and evaluation scripts (not documented well yet) are online:
  https://github.com/ducha-aiki/ransac-tutorial-2020-data
- Let’s start!
Fundamental and essential estimators
Modular WBS pipeline to benchmark everything

- 15 training image sets
- 10 test image sets, 100 images each
- Metric: pose accuracy

We benchmark this two

Task 1: Stereo
RANSAC
Pose Error Computation

Not used in current work

The Phototourism Dataset

- 30k images from YCC100M dataset, in 26 scenes
- “Ground truth” established by COLMAP reconstruction
- The basis of Image Matching Competitions 2019 & 2020
The Phototourism Dataset

### Training sets

<table>
<thead>
<tr>
<th>Name</th>
<th>Images</th>
<th>3D points</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Brandenburg Gate”</td>
<td>1363</td>
<td>100040</td>
</tr>
<tr>
<td>“Buckingham Palace”</td>
<td>1676</td>
<td>234052</td>
</tr>
<tr>
<td>“Colosseum Exterior”</td>
<td>2063</td>
<td>259807</td>
</tr>
<tr>
<td>“Grand Place Brussels”</td>
<td>1083</td>
<td>229788</td>
</tr>
<tr>
<td>“Hagia Sophia Interior”</td>
<td>888</td>
<td>235541</td>
</tr>
<tr>
<td>“Notre Dame Front Facade”</td>
<td>3765</td>
<td>488895</td>
</tr>
<tr>
<td>“Palace of Westminster”</td>
<td>983</td>
<td>115868</td>
</tr>
<tr>
<td>“Pantheon Exterior”</td>
<td>1401</td>
<td>166923</td>
</tr>
<tr>
<td>“Prague Old Town Square”</td>
<td>2316</td>
<td>558600</td>
</tr>
<tr>
<td>“Reichstag”</td>
<td>75</td>
<td>17823</td>
</tr>
<tr>
<td>“Sacre Coeur” (SC)</td>
<td>1179</td>
<td>140659</td>
</tr>
<tr>
<td>“St. Peter’s Square” (SPS)</td>
<td>2504</td>
<td>232329</td>
</tr>
<tr>
<td>“Taj Mahal”</td>
<td>1312</td>
<td>94121</td>
</tr>
<tr>
<td>“Temple Nara Japan”</td>
<td>904</td>
<td>92131</td>
</tr>
<tr>
<td>“Trevi Fountain”</td>
<td>3191</td>
<td>580673</td>
</tr>
<tr>
<td>“Westminster Abbey”</td>
<td>1061</td>
<td>198222</td>
</tr>
</tbody>
</table>

Total: 25.7k images, 3.6M 3D points

### Test sets

<table>
<thead>
<tr>
<th>Name</th>
<th>Images</th>
<th>3D points</th>
</tr>
</thead>
<tbody>
<tr>
<td>“British Museum” (BM)</td>
<td>660</td>
<td>73569</td>
</tr>
<tr>
<td>“Florence Cathedral Side” (FCS)</td>
<td>108</td>
<td>44143</td>
</tr>
<tr>
<td>“Lincoln Memorial Statue” (LMS)</td>
<td>850</td>
<td>58661</td>
</tr>
<tr>
<td>“London (Tower) Bridge” (LB)</td>
<td>629</td>
<td>72235</td>
</tr>
<tr>
<td>“Milan Cathedral” (MC)</td>
<td>124</td>
<td>33905</td>
</tr>
<tr>
<td>“Mount Rushmore” (MR)</td>
<td>138</td>
<td>45350</td>
</tr>
<tr>
<td>“Piazza San Marco” (PSM)</td>
<td>249</td>
<td>95895</td>
</tr>
<tr>
<td>“Sagrada Familia” (SF)</td>
<td>401</td>
<td>120723</td>
</tr>
<tr>
<td>“St. Paul’s Cathedral” (SPC)</td>
<td>615</td>
<td>98872</td>
</tr>
</tbody>
</table>

Total: 3774 images, 643k 3D points
Can we trust Colmap “Ground truth”? 

Yes, we can!

<table>
<thead>
<tr>
<th>Feature used</th>
<th>Number of images</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100 vs. all</td>
</tr>
<tr>
<td>SIFT [55]</td>
<td>0.46° / 0.13°</td>
</tr>
<tr>
<td>SuperPoint [34]</td>
<td>2.09° / 1.57°</td>
</tr>
<tr>
<td>R2D2 [80]</td>
<td>0.41° / 0.14°</td>
</tr>
</tbody>
</table>

Table 3  Pose convergence in SfM. We report the mean/median of the difference (in degrees) between the poses extracted with the full set of 1179 images for “Sacre Coeur”, and different subsets of it, for three local feature methods – to keep the results comparable we only look at the 100 images in common across all subsets. We report the maximum among the angular difference between rotation matrices and translation vectors. The estimated poses are stable, with as low as 100 images.
Metric computation

1. RANSAC $\rightarrow$ F or E
   a. Essential matrix E from F: $E = K_1^T F K_2$
2. Camera pose (R,t) from OpenCV function cv2.recoverPose(E, inliers)
3. Decompose (R,t) into rotation and translation components, keep only rotation, get the angular error
4. Threshold angular error for set of thresholds and get accuracy per threshold
5. Calculate mAA @ 10
Local features comparison on PhotoTourism

- Local feature: RootSIFT.
  - RootSIFT is a very strong baseline
    #9 among all methods
  - 13.1% relative less than the best

- Matching: mutual nearest neighbour

- Additional info: SNN Lowe ratio

https://vision.uvic.ca/image-matching-challenge/leaderboard/
Training and test correspondences: the same for all methods

• Training data: 100k image pairs per training sequence
  • Total: 1.6M image pairs, 500-3000 correspondences each.
• Test data: 4950 image pairs per test sequence
• The training data and the test data (without GT) was sent to deep learning method authors.
Evaluated methods: classical F

Baselines

- scikit-image: 8pt vanilla RANSAC + final least squares fitting
- OpenCV RANSAC: 7pt vanilla RANSAC + Levenberg-Marquardt final fitting
- OpenCV LMedS: least medians of squares
- LO-RANSAC: 7pt LO-RANSAC + final least squares fitting

Advanced methods

- **DEGENSAC**: Two-view Geometry Estimation Unaffected by a Dominant Plane
- **GC-RANSAC**: Graph-Cut RANSAC
- **MAGSAC**: marginalizing sample consensus
- MAGSAC++, a fast, reliable and accurate robust estimator
- **USAC**: A Universal Framework for Random Sample Consensus
- **AC RANSAC**: Automatic Homographic Registration of a Pair of Images with A Contrario Elimination of Outliers
- GC-RANSAC + DEGENSAC
- GC-RANSAC + DEGENSAC + MAGSAC++
Classical methods, which are not included

- Affine RANSACs (2AC, 3AC)
- Colmap RANSAC
- Guided matching methods
Evaluated methods: learned F

All methods evaluated are using only correspondences \((x,y) \ (x,y)\) and the match confidence.

- **DFE: Deep Fundamental Matrix Estimation**
- **CNe: Learning to Find Good Correspondences**
- **ACNe: Attentive Context Normalization for Robust Permutation-Equivariant Learning**
- **NM-Net: Mining Reliable Neighbors for Robust Feature Correspondences** (v2)
- **OANet: Learning Two-View Correspondences and Geometry Using Order-Aware Network** (v2)

**Not included:**
- SuperGlue, as it takes much richer input, that the most of the methods in our study. SuperGlue uses all raw keypoints and descriptors from both images.
- NG-RANSAC (CVPR 2020) - lack of time, hope to add later
- 6D Conv (CVPR 2020) - lack of time, hope to add later
- Eigen-Free training (ECCV 0218) - author doesn’t respond
Is OpenCV RANSAC a way to go?

Finally, which RANSAC do you use for python? F and H cases

OpenCV 69%

USAC 2%

DSAC 5%

Custom (specify) 24%

42 голоси · Остаточні результати

OpenCV functions:

cv2.findHomography()
cv2.findFundamentalMatrix()

https://twitter.com/ducha_aiki/status/1142847831516037120
Classical F methods, 1k iterations

- Methods are sorted by accuracy
- sk-image RANSAC is orders of magnitude slower than the rest
- OpenCV is the least precise RANSAC
**Classical F methods in Jin et.al 2020**

**Performance vs cost: mAP(5°)**

**Performance vs cost: mAP(10°)**

- Feature: SIFT, 8k points
- Vary maxIters, measure time.
- Advanced methods (MAGSAC, GC, DEGENSAC) are better for both per second and per iteration

**Jin et.al 2020:**

**This tutorial:**

- Benchmark was run on 4 different machines.
- Instead we fix the number of iterations for 1k, 10k, 100k, 1M
- We tune all parameters of the methods e.g. “spatial coherence term” for GC-RANSAC, which improves its results significantly

MAGSAC and MAGSAC++ [github.com/danini/magsac](https://github.com/danini/magsac) (CVPR 2019 & CVPR 2020)

DEGENSAC [github.com/ducha-aiki/pydegensac](https://github.com/ducha-aiki/pydegensac) Chum et. al CVPR 2005

GC-RANSAC [github.com/danini/graph-cut-ransac](https://github.com/danini/graph-cut-ransac) Barath and Matas. Graph-cut RANSAC. CVPR 2018
• Method ranking is consistent for different accuracy thresholds.
• In Jin et.al. GC-RANSAC < DEGENSAC, because of lacking of non-standard parameter tuning
• OpenCV E & F had hardcoded. maxIters = 1000. Now fixed, but not for pip install (yet?)
Classical F methods
100k iterations

- A-Contrario RANSAC needs tuning as well as other RANSACs. Without, it performs very poorly.
- The same is true for MAGSAC (untuned is not shown here, but it is bad).
- USAC is worse than standalone DegenSAC, probably due to implementation issues.
Classical F: 1k iterations vs 100k iterations

Classical F methods with 1k vs 100k iterations

mAA

0.0 0.1 0.2 0.3 0.4 0.5

LO-RANSAC 1k  LO-RANSAC 100k  DegenSAC 1k  DegenSAC 100k  MAGSAC++ 1k  MAGSAC++ 100k  GC 1k  GC 100k  GC + DegenSAC 1k  GC + DegenSAC 100k

Learned methods F with DLT and RANSACs

- **Learned + DLT**
- **RANSACs**
- **Learned + RANSACs**

F-summary: general results

• We haven’t exaustively tried to pair all RANSACs with all deep methods
• Least squares (DLT) is not good enough for model estimation after deep methods.
• Degenerate solutions are the big problem
• “We are parameter-free” (AC-RANSAC, MAGSAC) claims are not true
• May be:
  • use iterated re-weighted least squares on top of deep correspondences
  • but also inside (small) RANSAC loop to ensure robustness
F-summary: recommendations

• Deep learned methods are not replacements for the RANSAC, they are replacement for the hand-crafted correspondence pre-filtering like Lowe SNN ratio.
• Performance of different RANSACs varies significantly, and all the methods have to be tuned to perform well
• Don’t use OpenCV or sk-image F-RANSACs, use GC-RANSAC, MAGSAC or DEGENSAC (all available with python bindings)
• Implementation matters (see USAC fail)
Evaluated methods: classical E

Baselines

- scikit-image: 8pt vanilla RANSAC + final least squares fitting
- OpenCV RANSAC: 5pt vanilla RANSAC + LM final fitting
- OpenCV LMedS: least medians

Advanced methods

- GC-RANSAC: Graph-Cut RANSAC
- MAGSAC: marginalizing sample consensus
- MAGSAC++, a fast, reliable and accurate robust estimator
- USAC: A Universal Framework for Random Sample Consensus
- AC RANSAC: Automatic Homographic Registration of a Pair of Images with A Contrario Elimination of Outliers
- GC-RANSAC + DEGENSAC
- GC-RANSAC + DEGENSAC + MAGSAC++
Evaluated methods: learned E

- **CNe: Learning to Find Good Correspondences**
- **ACNe: Attentive Context Normalization for Robust Permutation-Equivariant Learning**
- **NM-Net: Mining Reliable Neighbors for Robust Feature Correspondences (v2)**
- **OANet: Learning Two-View Correspondences and Geometry Using Order-Aware Network (v2)**
Classical methods, E  1k iterations

<table>
<thead>
<tr>
<th>Method</th>
<th>mAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>skimage</td>
<td>0.0</td>
</tr>
<tr>
<td>OpenCV</td>
<td>0.0</td>
</tr>
<tr>
<td>MAGSAC</td>
<td>0.0</td>
</tr>
<tr>
<td>GC</td>
<td>0.0</td>
</tr>
<tr>
<td>GC + DegenSAC</td>
<td>0.0</td>
</tr>
<tr>
<td>GC + MAGSAC</td>
<td>0.0</td>
</tr>
</tbody>
</table>

- skimage 8pt solver is significantly worse and orders of magnitude slower than 5 pt solvers in the rest of methods
- Unlike F, there is no big difference in performance among methods
- OpenCV E works quite good
A-Contrario RANSAC needs tuning as well as other RANSACs. Without, it performs very poorly.

USAC is worse than even OpenCV.
The benefit of having 100x more iterations is quite small.
Learned methods E

- Learned + DLT
- RANSACs
- Learned + RANSACs
E-summary

• Deep learned methods are not replacements for the RANSAC, they are replacement for the hand-crafted correspondence pre-filtering like Lowe SNN ratio.

• Performance of different RANSACs for E estimation varies, but not as significant, as for F

• Don’t use sk-image E-RANSAC, instead use MAGSAC, GC-RANSAC or OpenCV
Homography estimators
Datasets: Hpatches-Sequences Viewpoints & EVD

• Problems:
  • There is no large-scale and diverse real-world homography dataset
  • If we are going to synthetic training data (by augmentation), then the best method is the one with closest statistics to the test set

• Our (temporary) solution:
  • Use HPatches Sequences (57x5 pairs) and Extreme View Dataset (15 pairs) with random validation/test splits 50/50, without training set
Input data

• HPatches sequences: RootSIFT features
  • Matching: mutual nearest neighbour
  • Additional info: SNN Lowe ratio

• EVD: MSER-RootSIFT and Hessian-Affine-RootSIFT MODS-generated correspondeces with view synthesis.
  • Matching: uni-directional nearest neighbor
  • Additional info: SNN Lowe ratio
Metric: mAA@20 px reprojection error

The image mask is reprojected to the 2nd image and back to get the commonly visible area.

Mean reprojection error is averaged over the visible area and then thresholded to get the mAA. Thresholds are from 1 to 20px in logspace.
Evaluated methods: classical H

Baselines

- **scikit-image**: 4pt vanilla RANSAC + final least squares fitting
- **OpenCV RANSAC**: 4pt vanilla RANSAC + LM final fitting
- OpenCV LMed: least medians
- OpenCV RHO: Fast Target Recognition on Mobile Devices: Revisiting Gaussian Elimination for the the Estimation of Planar Homographies
- **PyRANSAC**: 4pt LO-RANSAC + final least squares fitting

Advanced methods

- **GC-RANSAC**: Graph-Cut RANSAC
- **MAGSAC**: marginalizing sample consensus
- **MAGSAC++**: MAGSAC++, a fast, reliable and accurate robust estimator
- **USAC**: A Universal Framework for Random Sample Consensus
- **AC RANSAC**: Automatic Homographic Registration of a Pair of Images with A Contrario Elimination of Outliers
- **GC-RANSAC + MAGSAC++**
Classical methods, H, 1k iterations
Classical methods, H, 10k iterations
Hpatches-Sequences Viewpoints & EVD
1k vs 10k vs 1M iterations for H

It is better to use more advances RANSAC, than go for the 1M iterations
H-summary

• Performance of different RANSACs varies significantly, and all the methods have to be tuned to perform well
• Don’t use USAC, sk-image H-RANSACs, OpenCV RHO.
• Use GC-RANSAC, LO-RANSAC, MAGSAC or OpenCV H
Rigid PCL registration estimators
Point Cloud Registration. Datasets:

- RGB-D Dataset 7-Scenes + RGBD Object datasets: 4 scenes, median of 10 runs
- ETHZ IGP datasets: 22 scenes, 1 run (large scale, slow)

- Due to the small size of the datasets and their different statistics, there is no val/test split.
- We have run algorithms with “reasonable defaults”, but…take results with a grain of salt.
- If you would like to do a proper benchmark for the 3D point cloud registration, please contact us.
Point Cloud Registration. Metrics

• Mean rotation error [deg]
• Mean translation error [m]
• Time [s]
Point Cloud Registration. Evaluated methods

- **GORE**: Guaranteed Outlier Removal for Point Cloud Registration with Correspondences
- **GC-RANSAC**: Graph-Cut RANSAC
- **MAGSAC**: marginalizing sample consensus
- **MAGSAC++**, a fast, reliable and accurate robust estimator
- **USAC**: A Universal Framework for Random Sample Consensus
- **GC-RANSAC + MAGSAC++**
- **GORE + (*SACs)**
- **TEASER**: Fast and Certifiable Point Cloud Registration
- **Branch and Bound (BnB)**
Point Cloud Registration. Test on MS Scenes

Rigid point cloud registration MS 2 scenes

Point Cloud Registration. Test on MS Scenes: Runtime
Point Cloud Registration. Test on RGBD Objects

Rigid point cloud registration RGBD-Object

Rigid point cloud registration RGBD-Object
Point Cloud Registration. Test on RGBD Objects: runtime
RANSAC does not work here without GORE due to low inlier ratio
Point Cloud Registration. Test on ETHZ IGP: runtime

Rigid point cloud registration ETHZ IGP

Time [s]

0 100 200 300 400 500

GORE + RANSAC TEASER GORE + MAGSAC GORE + GC-RANSAC GORE + MAGSAC++

Translation error [m]

0.00 0.01 0.02 0.03 0.04 0.05

TEASER GORE + RANSAC GORE + GC-RANSAC GORE + MAGSAC++ GORE + MAGSAC

Rigid conclusions

• There is no cheap pre-filtering method like Lowe`s SNN ratio check here
• For some problems you need to use GORE to get the answer at all.
• Good place to put deep learning methods as pre-filters
Overall conclusions

• One still cannot do (yet?) robust estimation without RANSAC
• For further progress of learned methods we need new large scale datasets for training and evaluation
• For any method – we need a protocol to evaluation and train/val/test splits
• Implementation matters a lot.
• Use GC-RANSAC + deep pre-filtering methods, if available.

Thank you for the attention. Questions?