DSAC - Differentiable RANSAC for Camera Localization

Eric Brachmann\textsuperscript{1}, Alexander Krull\textsuperscript{1}, Sebastian Nowozin\textsuperscript{2}, Jamie Shotton\textsuperscript{2},
Frank Michel\textsuperscript{1}, Stefan Gumhold\textsuperscript{1}, Carsten Rother\textsuperscript{1}
\textsuperscript{1} TU Dresden, \textsuperscript{2} Microsoft

CSE 291D
Bekhzod Soliev
Background

• RANSAC (RAandom Sample Consensus) is an iterative algorithm for robust estimation that deals with a large number of outliers in the data.

• Used in fundamental matrix estimation, object detection, pose estimation, SLAM, etc.

Brachman et al., CVPR 2017
RANSAC

• Generate a set of model hypotheses by sampling minimal subsets of the data.
• Score hypotheses using all data points based on some measure of consensus, e.g. by counting inliers.
• Select the best scoring hypothesis – the one that has the highest consensus.
• Refine the selected hypothesis using additional data points, e.g. the full set of inliers.

Brachman et al., CVPR 2017
RANSAC

Motivation

- Deep learning has been a huge success in Computer Vision:
  - End-to-end training
  - Task specific loss function
- Integrate RANSAC into a deep learning pipeline.
Motivation

• RANSAC can’t be directly used in end-to-end learning:
  • Choosing the best hypothesis (argmax) is not differentiable

• Create a deep learning pipeline that mimics RANSAC and allows end-to-end learning by softening a non-differentiable operators.
Camera localization

• Given single RGB image in a known static scene.
• Estimate the 6D camera pose (3D translation and 3D rotation) relative to the scene).

• Widely used in robotics (SLAM), virtual and augmented reality, autonomous driving, et al.

Brachman et al., CVPR 2017
Vanilla RANSAC

\[ h_{AM} = \arg\max_h s_j \]

\[ h_j := H(Y_j) \quad s_j := s(h_j, Y) \]

- non-differentiable, vanilla RANSAC with hard, deterministic argmax selection.
Vanilla RANSAC

- Correspondence prediction done via CNN
- Minimal Set Sampling size is 4
- Hypothesis Generation is PnP algorithm (n = 4)
- Scoring – number of inliers whose reprojection error is less than a given threshold

\[ h_{AM} = \arg\max_{h_j} s_j \]

\[ h_j := H(Y_j) \quad s_j := s(h_j, Y) \]
Learning in a RANSAC pipeline

a) Vanilla RANSAC

\[ h_{AM} = \arg \max_{h_j} s_j \]

\[ h_j := H(Y_j) \quad s_j := s(h_j, Y) \]

\[ \tilde{w}, \tilde{v} = \arg \min_{w,v} \sum_{I \in \mathcal{I}} \ell(R(h_{AM}^{w,v}, Y^w), h^*) \]
Differentiable RANSAC with deterministic, soft argmax selection.

\[
\hat{w}, \hat{v} = \arg\min_{w,v} \sum_{I \in \mathcal{I}} \ell(R(h_{AM}^{w,v}, Y^{w}), h^*)
\]
Soft argmax Selection (SoftAM)

The Soft argmax Selection (SoftAM) is defined as:

$$h_{\text{SoftAM}} = \sum_{j} \frac{\exp(s_j)h_j}{\sum_{j'} \exp(s_{j'})}$$

This is weighted average of hypothesis $h_j$.

The probability of choosing hypothesis $J$ given weights $v, w$ is:

$$P(J|v, w) = \frac{\exp(s(h_j^w, Y^w; v))}{\sum_{j'} \exp(s(h_{j'}^w, Y^w; v))}$$

This is predicted by the scoring function.
Probabilistic Selection (DSAC)

c) Probabilistic Selection (DSAC)

\[ h_{DSAC} = h_j, j \sim \frac{\exp(s_j)}{\sum_{j'} \exp(s_{j'})} \]

- differentiable RANSAC with hard, probabilistic selection (named DSAC).

\[ \tilde{w}, \tilde{v} = \text{argmin}_{w,v} \sum_{I \in \mathcal{I}} \ell(R(h_{AM}^{w,v}, Y^w), h^*) \]
Probabilistic Selection (DSAC)

c) Probabilistic Selection (DSAC)

\[ h_{DSAC} = h_J, J \sim \frac{\exp(s_J)}{\sum_{J'} \exp(s_{J'})} \]

\[ h_{DSAC}^{w,v} = h_J^w, \text{ with } J \sim P(J|v, w) \]

\[ P(J|v, w) = \frac{\exp(s(h_J^w, Y^w; v))}{\sum_{J'} \exp(s(h_{J'}^w, Y^w; v))} \]

The hypothesis is chosen probabilistically.

Probability of choosing hypothesis J given weights v, w.

Predicted by the scoring function.
Loss function

\[ \ell_{\text{pose}}(h, h^*) = \max\left(\angle(\theta, \theta^*), \|t - t^*\|\right) \]

- \( h \) – predicted hypothesis
- \( h^* \) – ground-truth hypothesis
- \((\theta, t)\) – predicted axis angle rotation and translation of the camera (degree, cm)
- \((\theta^*, t^*)\) – ground-truth axis angle rotation and translation of the camera
Differentiable Camera Localization Pipeline

• Two CNNs:
  • Coord CNN: predicting 2D-3D image correspondences
  • Score CNN: predicting scores for hypothesis (learning consensus)

Brachman et al., CVPR 2017
Componentwise Training

- Coord CNN and Score CNN

\[ l_{\text{coord}}(y, y^*) = \| y - y^* \| \]

\[ l_{\text{score}}(s, s^*) = | s - s^* |, \text{ where: } s^* = -\beta l_{\text{pose}}(h, h^*) \]

Brachman et al., CVPR 2017
End-to-End Training

\[ \ell_{\text{pose}}(h, h^*) = \max(\angle(\theta, \theta^*), \|t - t^*\|) \]

We measure angle \( \angle(\theta, \theta^*) \) between estimated and ground truth rotation in degree, and distance \( \|t - t^*\| \) between estimated and ground truth translation in cm.
End-to-End Training

- The derivatives for hypothesis selection (PNP) and the refinement are calculated with central differences.
- Initialization: componentwise training
- Optimization: stochastic gradient descent with momentum

Brachman et al., CVPR 2017
Table 1. Accuracy measured as the percentage of test images where the pose error is below 5cm and 5°. *Complete* denotes the combined set of frames (17000) of all scenes. Numbers in **green** denote improved accuracy after end-to-end training for SoftAM resp. DSAC compared to componentwise training. Similarly, **red** numbers denote decreased accuracy. **Bold** numbers indicate the best result for each scene.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RANSAC</td>
<td>SoftAM</td>
<td>DSAC</td>
</tr>
<tr>
<td>Chess</td>
<td>70.7%</td>
<td><strong>94.9%</strong></td>
<td>94.9%</td>
</tr>
<tr>
<td>Fire</td>
<td>49.9%</td>
<td>73.5%</td>
<td>75.1%</td>
</tr>
<tr>
<td>Heads</td>
<td>67.6%</td>
<td>48.1%</td>
<td>72.5%</td>
</tr>
<tr>
<td>Office</td>
<td>36.6%</td>
<td>53.2%</td>
<td>70.4%</td>
</tr>
<tr>
<td>Pumpkin</td>
<td>21.3%</td>
<td><strong>54.5%</strong></td>
<td>50.7%</td>
</tr>
<tr>
<td>Kitchen</td>
<td>29.8%</td>
<td>42.2%</td>
<td>47.1%</td>
</tr>
<tr>
<td>Stairs</td>
<td>9.2%</td>
<td><strong>20.1%</strong></td>
<td>6.2%</td>
</tr>
<tr>
<td>Average</td>
<td>40.7%</td>
<td>55.2%</td>
<td>59.5%</td>
</tr>
<tr>
<td>Complete</td>
<td>38.6%</td>
<td>55.2%</td>
<td>61.0%</td>
</tr>
</tbody>
</table>

Brachman et al., CVPR 2017
Table 2. **Median pose errors** of the complete 7-Scenes dataset (17000 frames). Most accurate results marked **bold**.

<table>
<thead>
<tr>
<th>Method</th>
<th>Technique</th>
<th>Median Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brachmann et al. [5]</td>
<td></td>
<td><strong>4.5cm, 2.0°</strong></td>
</tr>
<tr>
<td>Ours, Trained Componentwise</td>
<td>RANSAC</td>
<td>4.0cm, 1.6°</td>
</tr>
<tr>
<td></td>
<td>SoftAM</td>
<td><strong>3.9cm, 1.6°</strong></td>
</tr>
<tr>
<td></td>
<td>DSAC</td>
<td>4.0cm, 1.6°</td>
</tr>
<tr>
<td>Ours, Trained End-To-End</td>
<td>SoftAM</td>
<td>4.0cm, 1.6°</td>
</tr>
<tr>
<td></td>
<td>DSAC</td>
<td><strong>3.9cm, 1.6°</strong></td>
</tr>
</tbody>
</table>
Results

Figure 3. (a) Effect of end-to-end learning on pose accuracy w.r.t. individual components. (b) Effect of end-to-end training on the average entropy of the score distribution. Set text for details.

Brachman et al., CVPR 2017
Figure 4. **Prediction quality.** We analyze scene coordinate prediction quality on an Office test image (a) with ground truth scene coordinates (b) (XYZ mapped to RGB). The prediction after componentwise training can be seen in (c). We visualize the relative change of prediction error w.r.t. componentwise training in (d) for SoftAM, resp. in (e) for DSAC. We observe an aggressive strategy of SoftAM which focuses large improvements on small areas (14% of predictions improve). DSAC shows small improvements but on large areas (38% of predictions improve). Note that DSAC achieves superior pose accuracy on this scene.
Conclusion

• Two ways of differentiating RANSAC
• Tested on camera localization problem
• Probabilistic selection (DSAC) was superior
• Can be deployed in any deep learning pipeline

Brachman et al., CVPR 2017
Conclusion

• Strength:
  • Differentiable
  • Novel
  • End-to-end learning

• Weaknesses:
  • Refinement differentiation may not be accurate
  • Overfitting for score prediction
Conclusion

• Possible improvements:
  • Making the refinement differentiable
  • Choosing a different way for computing consensus
  • Decrease the number of learning components to one

• Possible extensions:
  • Different applications, e.g. fundamental matrix prediction
  • Integrate it to a SFM pipeline with deep learning (end-to-end learning)
Thank you!

Questions? 😊