Dense Tracking and Mapping for Autonomous Quadrocopters

Jürgen Sturm

Joint work with Frank Steinbrücker, Jakob Engel, Christian Kerl, Erik Bylow, and Daniel Cremers
Motivation

- Imagine you have a flying camera
- What would you use it for?
Motivation

- Imagine you have a flying camera
- What would you use it for?
Motivation

- Imagine you have a flying camera
- What would you use it for?
Motivation

- Imagine you have a flying camera
- What would you use it for?
Motivation

- Our research goal: **Enable flying robots to operate autonomously in 3D environments using onboard cameras**
- Use cameras because light weight and rich data
- Navigation, localization, mapping, exploration, people following, ...
Feature-based Visual Navigation
[Engel, Sturm, Cremers, IROS ‘12]
Feature-based Visual Navigation
[Engel, Sturm, Cremers, IROS ‘12]

- **Architecture**

- **Based on PTAM** [Klein and Murray, ISMAR ‘07]
Motivation

- Video feed from quadrocopter
Motivation

- What PTAM actually sees
Motivation

- **Problem:** Most approaches only use a small fraction of the available data
  - Keypoint detection
  - Visual features

- **Question:** How can we use most/all information to maximize the performance?
Outline of the Talk

- Part 1: Dense tracking
- Part 2: Dense reconstruction
- Part 3: Evaluation and benchmarking
Related Work on Dense Tracking

- Lucas and Kanade (IJCAI’81)
- Lovegrove et al. (IV’11)
- Newcombe et al. (ICCV’11)
- Comport/Tykkälä et al. (ICCV’11)
Dense Tracking

- How can we exploit all data of an RGB-D image?
- Idea

- Photo-consistency constraint

\[ I_1(x) = I_2(\pi(g_{\xi}(z \cdot x))) \] for all pixels \( x \)
How to deal with noise?

- Photo-consistency constraint will not perfectly hold
  - Sensor noise
  - Pose error
  - Reflections, specular surfaces
  - Dynamic objects (e.g., walking people)

- Residuals will be non-zero

\[ r = I_1(x) - I_2(\pi(g_{\xi}(z \cdot x))) \]

- Residual distribution \( p(r) \)
Residual Distribution

- Zero-mean, peaked distribution
- Example: Correct camera pose
Residual Distribution

- Zero-mean, peaked distribution
- Example: Wrong camera pose

![Bar graph showing residual distribution with peaks at various residuals and probability values at the y-axis.]](image-url)
**Goal:** Find the camera pose that maximizes the observation likelihood
Motion Estimation

- **Goal:** Find the camera pose that maximizes the observation likelihood

\[ \hat{\xi}^* = \arg \max_\xi \prod_i p(r_i(\xi)) \]

compute over all pixels

- Assume pixel-wise residuals are conditionally independent

- How can we solve this optimization problem?
Approach

- Take negative logarithm

\[ \xi^* = \arg \min_{\xi} \sum_i - \log p(r_i(\xi)) \]

- Set derivative to zero

\[ \sum_i \frac{\partial \log p(r_i(\xi))}{\partial \xi} = \sum_i \frac{\partial \log p(r_i)}{\partial r_i} \frac{\partial r_i(\xi)}{\partial \xi} \overset{!}{=} 0 \]
Approach (cont.d)

- This can be rewritten as a weighted least squares problem

\[ \xi^* = \arg \min_{\xi} \sum_i w(r_i)(r_i(\xi))^2 \]

with weights \[ w(r_i) = \frac{\partial \log p(r_i)}{\partial r_i} \frac{1}{r_i} \]

- \( r_i(\xi) \) is non-linear in \( \xi \)
- Need to linearize, solve, and iterate
Iteratively Reweighted Least Squares

Problem: \[ \xi^* = \arg \min_{\xi} \sum_{i} w(r_i)(r_i(\xi))^2 \]

Algorithm:

1. Compute weights \[ w(r_i) = \frac{\partial \log p(r_i)}{\partial r_i} \frac{1}{r_i} \]
2. Linearize in the camera motion \( \xi \)
   \[ r_{\text{lin}}(\xi_0 + \Delta \xi) = r(\xi_0) + J \Delta \xi \]
3. Build and solve normal equations
   \[ J^T W J \Delta \xi = -J^T W r(\xi_0) \]
4. Repeat until convergence
Example

First input image

Second input image

Residuals

Image Jacobian for camera motion along x axis
What is a Good Model for the Residual Distribution?

![Graph showing different probability distributions for residuals](image)

- **normal distribution**
- **robust normal dist.**
- **t-distribution**
Weighted Error

![Graph showing weighted error](image)

- Red dashed line: normal distribution
- Green dashed line: Tukey weights
- Blue line: t-distribution

Weighted error $w(r)r^2$

Residual $r$

Jürgen Sturm, Computer Vision Group, TUM
Example Weights

- Robust sensor model allows to down-weight outliers (dynamic objects, motion blur, reflections, ...)

Scene

Residuals

Weights
Coarse-to-Fine

- Linearization only holds for small motions
- Coarse-to-fine scheme
- Image pyramids
Dense Tracking: Results
Summary: Dense tracking

- Direct matching of consecutive RGB-D images

Pro
- Super fast, highly accurate (30 Hz on CPU)
- Low, constant memory consumption

Con
- Accumulates drift over time, sometimes diverges

- How can we reduce/eliminate drift?
Dense Tracking and Mapping

- **Idea:** Instead of tracking from frame-to-frame, track frame-to-model to eliminate drift

- **Question:** Where do we get the model from?
Dense Tracking and Mapping

- **Idea:** Compute an iterative solution
  1. Reconstruct model with known poses
  2. Track camera with respect to known model

- **Next question:** How to represent the model?
**Representation of the 3D Model**

- **Idea:** Instead of representing the cell occupancy, represent the distance of each cell to the surface.

- **Occupancy grid maps**

- **Signed distance function (SDF)**

  - zero = free space
  - negative = outside obj.
  - positive = inside obj.

  ![Diagram](attachment:diagram.png)

  

  

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.3</td>
<td>-0.3</td>
<td>0.7</td>
<td>1.7</td>
</tr>
</tbody>
</table>
Dense Mapping: 2D Example

- Camera with known pose
Dense Mapping: 2D Example

- Camera with known pose
- Grid with signed distance function
Dense Mapping: 2D Example

- For each grid cell, compute its projective distance to the surface

projective distance
Dense Mapping: 3D Example

- Generalizes directly to 3D
- But: memory usage is cubic in side length
Dense Tracking

- **Given:**
  - SDF with projective distance to surface
  - New depth image with unknown camera pose

- **Wanted:** Camera pose

- **Our approach:**
  Optimize camera pose so that the observed depth image minimizes the distance in the SDF
Dense Tracking: 2D Example

- 3D model built from the first k frames
Dense Tracking: 2D Example

- Minimize distance between depth image and SDF
Dense Tracking: 2D Example

- Minimize distance between depth image and SDF
Dense Tracking: 2D Example

- Minimize distance between depth image and SDF
3D Example

- **Given**: A sequence of depth images
- **Wanted**: Accurate and dense 3D model
Resulting 3D Model
Can We Print These Models in 3D?
Summary: Dense 3D Reconstruction

- Frame-to-model tracking
- Pro:
  - Reduces drift significantly
  - Outputs nice 3D models
- Con:
  - Memory intensive (~2 GB for $256^3$)
  - Computationally more expensive (~60 Hz on GPU)
Evaluation and Benchmarking

- How can we evaluate such methods?
- What are good evaluation criteria?
Existing Benchmarks

- Intel dataset: laser + odometry [Haehnel et al., 2004]
- New College dataset: stereo + omni-directional vision + laser + IMU [Smith et al., IJRR’2009]
- KITTI Vision benchmark: stereo [Geiger et al., CVPR’12]
- Our contribution: Dataset for RGB-D evaluation
Recorded Scenes

- Different environments (office, industrial hall, ...)
- Large variations in camera speed, camera motion, illumination, number of features, dynamic objects, ...
- Handheld and robot-mounted sensor
Dataset Acquisition

- Motion capture system
  - Camera pose (100 Hz)
- Microsoft Kinect (later: Asus Xtion Pro Live)
  - Color images (30 Hz)
  - Depth images (30 Hz)
- External video camera (for documentation)
Motion Capture System

- 9 high-speed cameras mounted in room
- Cameras have active illumination and pre-process image (thresholding)
- Cameras track positions of retro-reflective markers
Calibration of the overall system is not trivial:
1. Intrinsic calibration (Mocap + Kinect)
2. Extrinsic calibration (Kinect vs. Mocap)
3. Time synchronization (Kinect vs. Mocap)
RGB-D SLAM Dataset and Benchmark

Contact: Jürgen Sturm

We provide a large dataset containing RGB-D data and ground-truth data with the goal to establish a novel benchmark for the evaluation of visual odometry and visual SLAM systems. Our dataset contains the color and depth images of a Microsoft Kinect sensor along the ground-truth trajectory of the sensor. The data was recorded at full frame rate (30 Hz) and sensor resolution (640×480). The ground-truth trajectory was obtained from a high-accuracy motion-capture system with eight high-speed tracking cameras (100 Hz). Further, we provide the accelerometer data from the Kinect. Finally, we propose an evaluation criterion for measuring the quality of the estimated camera trajectory of visual SLAM systems.

How can I use the RGB-D Benchmark to evaluate my SLAM system?

1. Download one or more of the RGB-D benchmark sequences (file formats, useful tools)
2. Run your favorite visual odometry/visual SLAM algorithm (for example, RGB-D SLAM)
3. Save the estimated camera trajectory to a file (file formats, example trajectory)
4. Evaluate your algorithm by comparing the estimated trajectory with the ground-truth trajectory. We provide an automated evaluation tool to help you with the evaluation. There is also an online version of the tool.
Dataset Download

We recommend that you use the 'xyz' series for your first experiments. The motion is relatively small, and only a small volume on an office desk is covered. Once this works, you might want to try the 'desk' dataset, which covers four tables and contains several loop closures.

We are happy to share our data with other researchers. Please refer to the respective publication when using this data.

Remarks:
- The file formats are described here.
- The intrinsic camera parameters are here.
- We provide a set of useful tools for working with the dataset.
- The *_validation sequences do not contain ground truth. They can only evaluated using the online tool.

<table>
<thead>
<tr>
<th>Sequence name</th>
<th>Duration</th>
<th>Length</th>
<th>Download</th>
<th>more info</th>
</tr>
</thead>
<tbody>
<tr>
<td>freiburg1_xyz</td>
<td>30.09s</td>
<td>7.112m</td>
<td>tgz (0.47GB)</td>
<td>more info</td>
</tr>
<tr>
<td>freiburg1_rpy</td>
<td>27.67s</td>
<td>1.664m</td>
<td>tgz (0.42GB)</td>
<td>more info</td>
</tr>
<tr>
<td>freiburg2_xyz</td>
<td>122.74s</td>
<td>7.029m</td>
<td>tgz (2.39GB)</td>
<td>more info</td>
</tr>
</tbody>
</table>
Dense Tracking and Mapping for Autonomous Quadrocopters
File Formats

- In total: 69 sequences (33 training, 36 testing)
- One TGZ archive per sequence, containing
  - Color and depth images (PNG)
  - List of color images (timestamp filename)
  - List of depth images (timestamp filename)
  - List of camera poses (timestamp tx ty tz qx qy qz qw)
What Is a Good Evaluation Metric?

- Visual odometry system outputs
  - Camera trajectory (accumulated)
- Visual SLAM system outputs
  - Camera trajectory
  - 3D map
- Ground truth
  - Camera trajectory
What Is a Good Evaluation Metric?

- Trajectory comparison
  - Ground truth trajectory
  - Estimate camera trajectory

- Two evaluation metrics
  - Drift per second
  - Global consistency

Ground truth trajectory $Q_{1:n}$

Evaluation function

Scalar performance index

Estimated camera trajectory $P_{1:n}$

$$Q_1, \ldots, Q_n \in SE(3)$$

$$P_1, \ldots, P_n \in SE(3)$$
Relative Pose Error (RPE)

- Measures the (relative) drift between the i-th frame and the (i+Δ)-th frame

\[
E_i := \left( Q_i^{-1} Q_{i+\Delta} \right)^{-1} \left( P_i^{-1} P_{i+\Delta} \right)
\]

Relative error  True motion  Estimated motion

Ground truth

Relative error

Estimated traj.
Relative Pose Error (RPE)

How to choose the time delta $\Delta$?

- For odometry methods:
  - $\Delta=1$: Drift per frame
  - $\Delta=30$: Drift per second

- For SLAM methods:
  - Average over all possible deltas
  - Measures the global consistency
Absolute Trajectory Error (ATE)

- Alternative method to evaluate SLAM systems
- Requires pre-aligned trajectories

\[ E_i := Q_i^{-1} S P_i \]

Absolute error

Ground truth

Pre-aligned estimated traj.
Evaluation Tools

- Average over all time steps

\[ \text{RMSE}(E_{1:n}) := \left( \frac{1}{m} \sum_{i=1}^{m} \| \text{trans}(E_i) \|^2 \right)^{1/2} \]

- Evaluation scripts for both evaluation metrics available (Python)

- Output: RMSE, median, mean

- Plot to png/pdf (optional)
Comparison of RPE and ATE

- RPE and ATE are strongly related
- RPE considers additionally rotational errors
- $\text{RPE} \geq \text{ATE}$
## Submission form for automatic evaluation of RGB-D SLAM results

<table>
<thead>
<tr>
<th>Groundtruth trajectory</th>
<th>freiburg1/xyz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated trajectory</td>
<td>Datei auswählen (Keine ausgewählt)</td>
</tr>
<tr>
<td>Evaluation options</td>
<td>Offset: 0.00 seconds (add to stamps of estimated traj.)</td>
</tr>
<tr>
<td></td>
<td>Scale: 1.00 (scale estimated traj. by this factor)</td>
</tr>
<tr>
<td>Evaluation mode</td>
<td>absolute trajectory error (recommended for the evaluation of visual SLAM methods)</td>
</tr>
<tr>
<td></td>
<td>relative pose error for pose pairs with a distance of 1 second(s) (recommended for the evaluation of visual odometry methods)</td>
</tr>
<tr>
<td></td>
<td>relative pose error for all pairs (downsampled to 10000 pairs)</td>
</tr>
</tbody>
</table>

**Start evaluation**

Runs the evaluation script on your data and displays the result. No data will be permanently saved on our servers. Alternatively, you can also download the evaluation script and perform the evaluation offline. Additional information about the evaluation options and the file formats is available. We also provide an example trajectory for freiburg1/xyz by RGBD-SLAM as well as instructions how to reproduce these trajectories.
compared_pose_pairs 786 pairs
absolute_translational_error.rmse 0.013473 m
absolute_translational_error.mean 0.012029 m
absolute_translational_error.median 0.011176 m
absolute_translational_error.std 0.006068 m
absolute_translational_error.min 0.000939 m
absolute_translational_error.max 0.034727 m
Summary – TUM RGB-D Benchmark

- Dataset for the evaluation of RGB-D SLAM systems
- Ground-truth camera poses
- Evaluation metrics + tools available
- Since August 2011:
  - >17,000 visitors
  - >4,500 online trajectory evaluations
  - >15 published research papers using the dataset
- Next steps:
  - Possibility to upload own trajectories/publications
  - Results page and automated ranking
Conclusion

- Dense methods bear a large potential
  - Dense camera tracking
  - Dense 3D reconstruction
- Use benchmark for the comparison of alternative approaches
- Release as open-source in preparation
- Please contact us if you are interested in collaboration!