Visual SLAM and VO

Jakob Engel

Surreal Vision, Oculus Research

*Presentation reflects work done in collaboration with
Daniel Cremers at Technical University Munich and
Vladlen Koltun at Intel Research
Live Operation

[1] LSD-SLAM, Engel, Schoeps, Cremers ‘14
Real-Time Camera Tracking in Unknown Scenes

Parallel Tracking and Mapping for Small AR Workspaces
ISMAR 2007 video results
Georg Klein and David Murray
Active Vision Laboratory
University of Oxford

DTAM: Dense Tracking and Mapping in Real-Time

monoSLAM

PATM

MSCKF

Semi-Dense VO

ORB-SLAM2

SVO

LSD-SLAM

OKVIS

DSO

And many, many more!

 Jakob Engel

Visual SLAM
Overview

• Geometric SLAM formulations.
  • Indirect vs. direct error formulation.
  • Geometry parametrizations choices.
  • Sparse vs. dense model.
  • Optimization approach.
  • Concrete example: DSO direct error formulation.

• System architecture of two examples: ORB-SLAM and DSO

• Different sensor combinations (IMU, mono vs. stereo, RGB-D)

• Evaluating SLAM / VO, loopclosure metric.

• Overall analysis & parameter studies:
  Impact of field of view, image resolution, different noise sources, different algorithm and model choices in controlled experiments: Analyzing results from 40’000 tracked trajectories (100 million tracked frames, or 38 days of video at 30fps) on 50 sequences (190k frames).
Everything is based on **maximum likelihood** estimation:
Find model parameters $X$ that maximise the probability of observing $Y$.

**Indirect:** Observations $Y$ are 2D point positions.
=> $P$ models (geometric) noise on point positions, residual unit is pixels.

**Direct:** Observations $Y$ are pixel Intensities.
=> $P$ models (photometric) noise on pixel intensities, residual unit is radiance.

[2] DSO, Engel, Koltun, Cremers ‘17
## Indirect vs. Direct

<table>
<thead>
<tr>
<th>Indirect Model</th>
<th>Direct Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model Assumptions:</strong></td>
<td><strong>Indirect Model Assumptions:</strong></td>
</tr>
<tr>
<td>$p_{i}^{\text{true}} = \pi_{i}(\mathbf{X})$</td>
<td>$I_{\text{ref}}(p_{i}) = I^{\text{true}}(\pi_{i}(\mathbf{X}))$</td>
</tr>
<tr>
<td>(2D keypoint location equals projection of associated 3D point)</td>
<td>(template intensity equals image intensity at projected point)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measurement Noise Model:</th>
<th>Measurement Noise Model:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{i}^{\text{obs}} = p_{i}^{\text{true}} + n_{\text{geo}}$</td>
<td>$I^{\text{obs}} = I^{\text{true}} + n_{\text{photo}}$</td>
</tr>
<tr>
<td>$n_{\text{geo}} \sim \mathcal{N}(0_{2}, \sigma_{2 \times 2}^{2})$</td>
<td>$n_{\text{photo}} \sim \mathcal{N}(0, \sigma^{2})$</td>
</tr>
<tr>
<td>(gaussian white noise 2D keypoint)</td>
<td>(gaussian white noise on image)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Negative log-likelihood (Energy):</th>
<th>Negative log-likelihood (Energy):</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E(\mathbf{X}) = \sum_{p_{i} \in \Omega_{I_{\text{ref}}}} | r_{i}(\mathbf{X}) |_{2}^{2}$</td>
<td>$E(\mathbf{X}) = \sum_{p_{i} \in \Omega_{I_{\text{ref}}}} r_{i}^{2}(\mathbf{X})$</td>
</tr>
<tr>
<td>$r_{i}(\mathbf{X}) := [\pi_{i}(\mathbf{X}) - p_{i}^{\text{obs}}]$</td>
<td>$r_{i}(\mathbf{X}) := [I^{\text{obs}}(\pi_{i}(\mathbf{X})) - I_{\text{ref}}(p_{i})]$</td>
</tr>
<tr>
<td>(summed over a set of model points)</td>
<td>(summed over a set of template pixel)</td>
</tr>
</tbody>
</table>

---

[2] DSO, Engel, Koltun, Cremers ‘17

ORB-SLAM [3], PTAM [4], OKVIS [5], MSCKF [6], SVO [7] (back-end), monoSLAM [8], [11], ...

LSD-SLAM [1], DSO [2], DTAM [9], REMODE [10], SVO [7] (front-end), [12], [13], [15] ...
Geometry Parametrization

There are many ways to represent geometry: pointcloud, depth maps, TSDF, Mesh, occupancy grid, ....

For real-time visual SLAM most important options (non-exhaustive list):

• 3D euclidean points (world coordinates)

\[
\begin{bmatrix}
  x_w, y_w, z_w
\end{bmatrix}, \text{ with } P_w = \begin{bmatrix}
  x_w, y_w, z_w
\end{bmatrix}^T
\]

• 3D euclidean points (local coordinates in „host“ frame h)

\[
\begin{bmatrix}
  x_h, y_h, z_h
\end{bmatrix}, \text{ with } P_w = T_h^w \begin{bmatrix}
  x_h, y_h, z_h
\end{bmatrix}^T
\]

• 3D inverse points (local coordinates in „host“ frame h)

\[
\begin{bmatrix}
  u, v, d
\end{bmatrix}, \text{ with } P_w = T_h^w \begin{bmatrix}
  u/d, v/d, 1/d
\end{bmatrix}^T
\]

• 1D inverse depth (local coordinates in „host“ frame h)

\[
\begin{bmatrix}
  d
\end{bmatrix}, \text{ with } P_w = T_h^w \begin{bmatrix}
  u/d, v/d, 1/d
\end{bmatrix}^T
\]

• 1D inverse distance, 3D inverse (distance + 2 angles), .....
Local vs. World representation of points
Local representation is more flexible as points move with host frame (good when fixing values or linearization points). However, residuals now depend on two camera-to-world poses. Requires to define a „host“ frame.

Reduces geometry-camera pose correlation

Inverse vs. Euclidean
Inverse representation more linear wrt. epipolar geometry, almost always preferable.

1D vs 3D representation
1D obtained by fixing (u,v) in host frame. Correct for direct model when using host as template image. Strictly speaking incorrect for indirect model (see [2]).

[2] DSO, Engel, Koltun, Cremers ‘17
Not synonymous with *direct vs. indirect*!

*Dense + Indirect:* Compute dense optical flow, then estimate (dense) geometry from flow-correspondences.

*Sparse + Direct:* Just use a selected subset of pixels & group them.

- For **dense**, we need to integrate a **prior**, since (from passive vision alone), dense geometry is not observable (white walls).
- Typically, this prior is some version of „*the world is smooth*“.
- Causes correlations between geometry parameters (and significantly increases their number), which is bad for GN-like optimization methods.

[2] DSO, Engel, Koltun, Cremers ‘17
Dense vs. Sparse

Poses: ~5k Parameters.

Inverse Depth Values: ~100M Parameters.

Pose-Pose block (sparse block-matrix)

Pose-Depth correlations. (sparse block-matrix)

Depth-Depth block: very large, very sparse, but irregular (depth residuals + smoothness prior terms)

( theoretical ) Hessian Structure of LSD-SLAM [1]

[2] DSO, Engel, Koltun, Cremers ‘17
Dense vs. Sparse

• As a result, real-time Dense or Semi-Dense approaches refrain from jointly optimizing the complete model.

• They use alternating optimization (LSD-SLAM) or other optimization method like Primal-Dual (REMODE, DTAM).

• They favor large amounts of data to constrain the large number of unknowns, ignoring or approximating correlations.

• This **works well if the data is good**. It **fails if the configuration is close-to-degenerate**, since we may accumulate wrong linearizations, which will not be corrected as we ignored correlations & don’t re-linearize.
Kalman Filter  
*MSCKF [6] (indirect VIO), monoSLAM* [8] (indirect VO), Soatto et.al [11,12]*

Kausal integration of information over time, representing the current state as mean and covariance and marginalizing old information. Today mostly used for lightweight visual-inertial odometry.  
*Real-time by early fixation of linearizations and marginalization of old states*

Information Filter  
*OKVIS* [5] (indirect VIO), *DSO* [2] (direct VO)

Similar to EKF, but representing the state as minimum of a quadratic energy function. Today mostly used for visual-inertial odometry (slightly more accurate and expensive than EKF).  
*Real-time by early (but slightly later than EKF) fixation of linearizations and marginalization of old states*

Graph-Optimization  
*ORB-SLAM* [3], *PTAM* [4] (indirect SLAM), *LSD-SLAM* [1] (direct SLAM), SVO*

Maintains full map as grasp, typically optimizing in multiple layers: local active window (front-end) and full map (back-end). Allows re-using old geometry and loop-closures (drift-free scene browsing), rather than marginalizing everything outside the field of view.  
*Real-time by sub-sampling residuals / states (only optimize Keyframes)*

Primal-dual and others  
*REMODE* [10], *DTAM* [9] (direct dense reconstruction)

Allows to efficiently solve regularized dense systems wrt. depth, which are intractible for GN-based optimization. Typically poses are estimated with a separate optimization (sequential / alternating).  
*Real-time by GPU acceleration & fixing poses.*

* open-source code by authors available
Two Examples

ORB-SLAM 2 [3]
- **Fully indirect, sparse model** (FAST corners + ORB descriptors)
- Points represented as **3D euclidean points in world coordinates**
- Optimized using **graph-optimization (g2o)** with **local and global window**.

**Open-source code:**
- monocular, stereo- and RGB-D visual SLAM
- Implicit and explicit loop-closure, re-localization and map re-use.

Direct Sparse Odometry (DSO) [2]
- **Fully direct, sparse model** (plus photometric calibration)
- Points represented as **1D inverse depth in local frames**
- Optimized using **information filtering** (sliding window of keyframes)

**Open-source code:**
- monocular visual odometry
Direct Sparse Odometry (DSO):
Use photometric error over a small neighbourhood (8 pixel, ).

\[
E_{pj} := \sum_{p \in \mathcal{N}_p} w_p \left\| (I_j[p'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[p] - b_i) \right\|_\gamma
\]

- Warped point: \( p' = \Pi_c (R \Pi_c^{-1}(p, d_p) + t) \)
- Weights: \( w_p := \frac{c^2}{c^2 + \|\nabla I_i(p)\|^2_2} \)
- Exposure times: \( t_i, t_j \)
- Photometrically corrected Image: \( I'_i(x) := t_i B_i(x) = \frac{G^{-1}(I_i(x))}{V(x)} \)
- Per-Frame affine lighting parameters \( (a_i, b_i), (a_j, b_j) \)

[2] DSO, Engel, Koltun, Cremers '17
**Indirect model:** Sensor „measures“ 2D point positions.

**Direct model:** Sensor measures pixel radiance.

**Geometric Calibration:** 3D point -> 2D pixel coordinate. Well understood, and always done (often refined online).

**Photometric Calibration:** Scene irradiance (lux) -> 8-12 bit pixel value. Widely ignored: Features are invariant - brightness constancy is not.

\[
I_i(x) = G(t_i V(x) B_i(x))
\]

- pixel value
- exposure time
- pixel radiance
- camera response function (gamma function)
- attenuation factor (vignetting)

[14] TUM monoVO Dataset, Engel, Usenko, Cremers ‘16
Photometric Calibration

\[ I_i(x) = G(t_iV(x)B_i(x)) \]

- **Pixel value**
- **Exposure time**
- **Pixel irradiance**
- **Camera response function** (gamma function)
- **Attenuation factor** (vignetting)

Noise: Irradiance noise is not Gaussian but Poisson-distributed (mean = variance). With 0.5 gamma correction, variance becomes constant across pixel values.
So... what else is there to visual SLAM / Odometry than picking a probabilistic model (direct / indirect, sparse / dense), parameter representation (1D/3D, inverse/euclidean, local/world) and optimization method (EKF / Information Filster / Graph-Optimization / Primal-Dual)?

=> So far, this just gives you an energy function, and tells you how to optimize it (assuming a good initialization).

What will make or break a real-time SLAM / VO system are all the (often heuristic) nitty-gritty details (90% of effort and code, major impact on accuracy / robustness):

• How to select points / pixels to use?
• How to select (key-)frames to use?
• How to select observations / residuals to use (which points are visible where)?
• How to initialize values?
• Which energy terms / parameters are optimized / fixed / marginalized where and when?
Point Selection

**ORB-SLAM2:**
Divide image into coarse grid, **select best FAST-corner(s) from each cell, and across scale-space pyramid.**
Add new points to the optimization such that a certain number of spatially well-distributed points is observed in each keyframe.

**DSO:**
Divide image into coarse grid, **select pixel with largest absolute gradient** in each cell (on finest resolution only).
Add new points to the optimization such that a certain number spatially well-distributed of points is observed in each keyframe.

"Few good, outlier-free and spatially well-distributed points is typically better than many noisy points including a larger fraction of outliers."
Keyframe Selection

**ORB-SLAM2:**
Threshold on successfully tracked points and passed time between keyframes. General strategy: take more keyframes than needed, and sparsify afterwards (survival of the fittest).

Keep window of „active“ keyframes as n frames with largest co-visibility.

**DSO:**
Combined threshold on rotation, translation (measured as optical flow) and exposure change to last keyframe.

Keep sliding window of „active“ keyframes, as mix of short-baseline and large-baseline frames.

[2] DSO, Engel, Koltun, Cremers ’17

Critical for performance in practice. Lots of different approaches in literature.
Residual Selection

**ORB-SLAM2:**
ORB descriptors are used for data association. Accelerate & robustify matching by exploiting geometric constraints.

Attempt to observe each point in as many Keyframes as possible, maximising number of observations per point. Continuously optimize this data-association by merging points and adding new residuals where possible.

**DSO:**
Decide on visibility by thresholding photometric energy per-patch with adaptive threshold.

Only attempt to add residuals within sliding window of active frames, discard everything else to enable efficient optimization.

Determines actual energy terms optimized over.

Parameter Initialization

**ORB-SLAM2:**
- **Poses:** No good initialization needed.
- **Geometry:** Closed-form triangulation from initial matches. Only use points that are sufficiently well localized in euclidean 3D (sufficient observation parallax required).

**DSO:**
- **Poses:** Coarse-to-fine direct image alignment.
- **Geometry:** Pixel-wise, exhaustive search along epipolar line. Start with minimal baseline frames, then increase baseline while using prior from previous search. Use points at any distance (no minimal parallax required).

More difficult for direct approach due to non-linearity of energy.

Critical to avoid adding bad data to system.

[2] DSO, Engel, Koltun, Cremers ’17;
Optimization Strategy

ORB-SLAM2:
Front-End (at framerate):
Optimize newest frame’s pose wrt. fixed map.

Local Window (at keyframe-rate):
Optimize local window of keyframe poses and all observed points. Temporarily fix absolute poses of periphery keyframes.

Loop-Closing (when required):
Pose-graph optimization followed by full map optimization (BA).

DSO:
Front-End (at framerate):
Optimize newest frame’s pose wrt. fixed map.

Sliding Window (at keyframe-rate):
Optimize local window of keyframes and observed points. Permanently marginalize points once they leave the field of view, and keyframes once they drop out of the active window.

Loop-Closing:
Not supported, pure VO.

Here it can get tricky. Avoid adding biases or locking down estimated parameters by premature linearization / fixation of parameters.

MSCKF: one-off geometry triangulation & marginalization, „structureless“

[2] DSO, Engel, Koltun, Cremers ’17;
Sensor Choices

• **Stereo / multi-camera (incl. non-overlapping stereo):**
  Absolute scale observable, increases FOV. Best to use both static constraints (across static baseline), as well as temporal constraints (across time) thus not depend on a fixed stereo-baseline.

• **RBG-D cameras (ToF or structured light):**
  Metric geometry directly observable. Great for dense reconstruction, however commercial sensors are hard to calibrate and have a limited field of view. Can create „dense“ model without depending on regularization / geometry prior.

• **IMU:**
  In many ways complementary to vision. *Tight* IMU integration always is a good idea in practice, and gives impressive results.

• **Rolling shutter camera(s):** global vs. Rolling shutter makes a difference!
Important not just to get your paper accepted! Should guide model selection algorithm design choices.

SLAM / VO, due to its incremental nature, is a highly chaotic process: small perturbations can have big impact.

- Evaluate on many datasets in a wide range of „realistic“ environments (indoor/outdoor, bright/dark, high-frequent/low-frequent, unique / repetitive environments, ....). Avoid manual over-fitting to a few scenes.
- **Augment your data** (run forwards & backwards, different image crops)
- **Exploit non-deterministic** implementation to get a better sampling.

[14] TUM monoVO Dataset, Engel, Usenko, Cremers ‘16
TUM monoVO dataset

[14] TUM monoVO Dataset, Engel, Usenko, Cremers ‘16

- 50 sequences
- 105 minutes
- 190k frames
Impossible to get good ground-truth.
But we can still use it to evaluate tracking accuracy!

⇒ Measure accumulated drift after a large loop.
(obviously before / without closing the loop)

1. Use LSD-SLAM (or any other SLAM / SfM software) to track start- and end-segment.
⇒ they show the same scene with nice, loopy motion
⇒ very precise alignment.
⇒ „ground-truth“ poses for start- and end-segment.

2. Sim (3)-align tracked start- and end-segment to „gt“

\[
T_{s}^{gt} := \arg \min_{T \in \text{Sim}(3)} \sum_{i \in S} (Tp_i - \hat{p}_i)^2
\]

\[
T_{e}^{gt} := \arg \min_{T \in \text{Sim}(3)} \sum_{i \in E} (Tp_i - \hat{p}_i)^2.
\]

3. Explicitly compute accumulated error as Sim(3) transform:

\[
T_{\text{drift}} = T_{e}^{gt} (T_{s}^{gt})^{-1} \in \text{Sim}(3)
\]

[14] TUM monoVO Dataset, Engel, Usenko, Cremers ‘16
How to summarize this SE(3) drift in one number?

Proposed Alignment Error:

\[ e_{\text{align}} := \left( \frac{1}{n} \sum_{i=1}^{n} \| T_{s}^{gt} p_i - T_{e}^{gt} p_i \|_2^2 \right)^{\frac{1}{2}} \]

„RMSE between tracked trajectory (a) aligned to the start segment (blue), and (b) aligned to the end segment (red).”

(all trajectories are normalized to have approximately length 100)

- **Unifies translational, rotational and scale drift** in one number. **Implicitly weighs** rotation & scale by how they would affect translation.
- More meaningful than just translational drift.
- Works for other sensor modalities (VIO, RGB-D, Stereo, ...)
- No set-up required, everyone can do this at home.
- Use in addition to other metrics (e.g. absolute RMSE on datasets with MoCap groundtruth)

[14] TUM monoVO Dataset, Engel, Usenko, Cremers ‘16
[14] TUM monoVO Dataset, Engel, Usenko, Cremers ‘16
How to summarize 50 sequences in one number?

⇒ Average / median don’t work well, in particular if some sequences fail entirely.
⇒ Tables are not practical for 50 sequences.

use *Cumulative Error Plots*!

(„how many sequences Y were tracked more accurately than X?“)

[50 sequences] x [forward, backward] x [5 runs] = 500 run sequences (~20 hours of video) per line.

LSD-SLAM & SVO fail on almost all of the sequences, hence we evaluate only ORB-SLAM and DSO.

[14] TUM monoVO Dataset, Engel, Usenko, Cremers ‘16
Field of View

* Resolution fixed at 640x480.
** Raw (distorted) data at 1240 x 1024

larger FoV is better for both methods.
(wrt. tracking under general motion. Accurate geometry with minimal parallax requires large angular resolution).

[14] TUM monoVO Dataset, Engel, Usenko, Cremers ‘16
General Analysis

Image Resolution

* FoV fixed.
** Ignoring effect on compute requirement

**FOV fixed.**

**Ignoring effect on compute requirement**

**ORB-SLAM**’s accuracy strongly depends on resolution.

**DS-VO**’s accuracy only slightly depends on resolution, no gain above VGA.

[14] TUM monoVO Dataset,
Engel, Usenko, Cremers ‘16
Let’s play around a bit: Forward / Backward

* Sequences mostly contain forward motion.

ORB-SLAM seems to perform better when moving backwards.  
*(Impact of all the nitti-gritty details)*

[14] TUM monoVO Dataset, 
Engel, Usenko, Cremers ‘16
Effect of different noise

Add photometric Noise:
(motion blur, sensor noise / low light, ...)

Add geometric Noise:
(bad calibration, rolling shutter, chromatic aberration, ...)

Corresponds to direct model!

$$I^{\text{obs}} = I^{\text{true}} + n_{\text{photo}}$$

$$n_{\text{photo}} \sim \mathcal{N}(0, \sigma^2)$$

20ms exposure time, 100deg/s rotation,
Kinect camera geometry: delta = 10px

Corresponds to indirect model!

$$p_i^{\text{obs}} = p_i^{\text{true}} + n_{\text{geo}}$$

$$n_{\text{geo}} \sim \mathcal{N}(0, \sigma^2_{2 \times 2})$$

25ms readout time, 100deg/s rotation,
Kinect camera geometry: delta = 6 px

[2] DSO, Engel ‘17
Point Selection

Relatively robust to gradient threshold. Using only corners decreases robustness.

[2] DSO, Engel, Koltun, Cremers ‘17
Number of Points

$\epsilon_{\text{align}}$

number of runs

$N_p = 50$
$N_p = 100$
$N_p = 200$
$N_p = 500$
$N_p = 1000$
$N_p = 2000$
$N_p = 6000$
$N_p = 10000$

1000 points are enough (default: 2k points).

[2] DSO, Engel, Koltun, Cremers ‘17
Some more results

Number of Keyframes taken

More KF -> better robustness (if too little KF, gets lost on strong self-occlusion).

Less KF -> better accuracy (points & frames get marginalized later).

[2] DSO, Engel, Koltun, Cremers ‘17
Some more results

Algorithm Component: Energy Pixel Pattern

1-3 are slightly worse, otherwise no big difference.

[2] DSO, Engel, Koltun, Cremers ‘17
Summary

• Classify different Systems according to
  • Direct vs. Indirect
  • Sparse vs. Dense
  • Geometry Representation
  • Optimization approach

• All the “system-level” details to make a full SLAM / VO system out of it.

• Evaluate on large datasets, use loop-closure metric if groundtruth cannot be obtained.

• Use data to analyze systems and parameters.
References


For a more complete list of references and related work, see e.g. my PhD Thesis.