Lecture 13: Visual Inertial Fusion

1 Introduction

1.1 Pose Graph Optimization

So far we assumed that the transformations are between consecutive frames, but they can be computed between non adjacent frames $T_{ij}$ as well (e.g. when features from previous keyframes are still observed). They can be used as additional constraints to improve cameras poses by minimizing the following error measure:

$$C_k = \arg\min_{C_k} \sum_i \sum_j \|C_i - C_j \cdot T_{ij}\|^2$$  \hspace{1cm} (1.1)

- For efficiency, only the last $m$ keyframes are used.
- Gauss-Newton or Levenber-Marquadt are typically used to minimize it. For large graphs, there are open source tools.

![Figure 1: Pose graph optimization.](image)

1.2 Bundle Adjustment (BA)

This incorporates the knowledge of landmarks (3D points).

$$X^i, C_k = \arg\min_{X^i, C_k} \sum_i \sum_k \rho \left( p^i_k - \pi(X^i, C_k) \right).$$  \hspace{1cm} (1.2)

Outliers represent an issue: how can we penalize them? In order to penalize wrong matches, we can use the Huber or the Turkey costs:

<table>
<thead>
<tr>
<th>Cost</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huber</td>
<td>$\rho(x) = \begin{cases} x^2, &amp; \text{if }</td>
</tr>
<tr>
<td>Tukey</td>
<td>$\rho(x) = \begin{cases} \alpha^2, &amp; \text{if }</td>
</tr>
</tbody>
</table>
1.3 Bundle Adjustment vs Pose-graph Optimization

In generale, one can conclude the following:

- BA is more precise than pose-graph optimization because it adds additional constraints (landmark constraints).

- BA is but more costly: $O((qM + lN)^3)$ with $M$ and $N$ being the number of points and camera poses and $q$ and $l$ the number of parameters for points and camera poses. The Jacobian is cubic in $q$ and $l$. Workarounds are
  
  - A small window size limits the number of parameters for the optimization and thus makes real-time bundle adjustment possible.
  
  - It is possible to reduce the computational complexity by just optimizing the camera parameters and keeping the 3D landmarks fixed, e.g. freeze the 3D points and adjust the poses.
2 IMU and Camera-IMU System

2.1 IMU Definition

Inertial Measurement Unit. Measures angular velocity and linear accelerations. One can find:

- Mechanical: spring/damper system.
- Optical: Phase shift projected laser beams is proportional to angular velocity.
- MEMS (accelerometer): a spring-like structure connects the device to a seismic mass vibrating in a capacitive divider. A capacitive divider converts the displacement of the seismic mass into an electric signal. Damping is created by the gas sealed in the device.
- MEMS (gyroscopes): measure the Coriolis forces acting on MEMS vibrating structures. Their working principle is similar to the haltere of a fly. Have a look!

2.2 Why IMUs?

In the following, we list reasons to use IMUs:

- Monocular vision is scale ambiguous.
- Pure vision is not robust enough (Tesla accident):
  - Low texture.
  - High dynamic range.
  - High speed motion.

2.3 Why not just IMU? Why Vision?

Pure IMU integration will lead to large drift (especially cheap IMUs). Integration of angular velocity to get orientation: error proportional to $t$. Double integration to get position: if there is a bias in acceleration, the error of position is proportional to $t^2$. The actually position error also depends on the error of orientation.

2.4 Why visual inertial fusion?

In the following, we list advantages (+) and disadvantages (-) of cameras and IMUs:

- Cameras
  + Precise in slow motion.
  + Rich information for other purposes
  - Limited output rate ($\sim 100Hz$)
  - Scale ambiguity in monocular setup.
  - Lack of robustness
• IMU
  + Robust.
  + High output rate ($\sim 1000$ Hz).
  + Accurate at high acceleration.
  - Large relative uncertainty when at low acceleration/angular velocity.
  - Ambiguity in gravity/acceleration.

Together, they can work for state estimation: loop detection and loop closure.

2.5 IMU: Measurement Model

\[
\begin{align*}
\omega_{WB}^B(t) &= \omega_{WB}^B(t) + b^\omega(t) + n^\omega(t) \\
\dot{a}_{WB}^B(t) &= R_{BW}(t) \cdot (a_{WB}^W(t) - g^W) + b^a(t) + n^a(t)
\end{align*}
\]

where $g$ stands for gyroscope and $a$ for accelerometer. The noise is additive Gaussian white noise. The bias has own dynamics

\[
\dot{b}(t) = \sigma_b \cdot w(t),
\]

i.e. the derivative of the bias is white Gaussian noise (random walk). In discrete time, one writes

\[
b[k] = b[k-1] + \sigma_{bd} \cdot w[k], \quad w[k] \sim \mathcal{N}(0, 1), \quad \sigma_{bd} = \sigma_b \cdot \sqrt{t}
\]

In general, IMU biases:

• Can be estimated,
• Can change due to temperature change, mechanical pressure,..
• Can change everytime the IMU is started.

Integration leads to

\[
p_{Wt_2} = p_{Wt_1} + (t_2 - t_1)v_{Wt_1} + \int_{t_1}^{t_2} \int_{t_1}^{t_2} R_{Wt}(t) \left( \dot{a}(t) - b^a(t) + g^w \right) dt^2,
\]

which depends on initial position and velocity. The rotation $R(t)$ can be computed with a gyroscope.

2.5.1 Different Paradigms

Loosely Coupled Approach

It treats VO and IMU as two separate (not coupled black boxes). Each block estimates pose and velocity from visual and inertial data (pose and velocity up to a scale and inertial data in absolute scale).

Tightly Coupled Approach

It makes use of the raw sensors’ measurements: 2D features, IMU readings, more accurate, more implementation effort.
2.5.2 Filtering: Visual Inertial Formulation

System states are:

- **Tightly Coupled**: \( X = (p_W(t); q_{WB}(t); v_W(t); b^a(t); b^g(t); L_{w,1}; \ldots; L_{w,K}) \), with \( L \) Landmarks.

- **Loosely Coupled**: \( X = (p_W(t); q_{WB}(t); v_W(t); b^a(t); b^g(t)) \)

**Closed-form Solution (1D case)**

The absolute pose \( x \) is known up to a scale \( s \), thus

\[
x = s\tilde{x}.
\]  

(2.5)

From the IMU we get

\[
x = x_0 + v_0 \cdot (t_1 - t_0) + \int_{t_0}^{t_1} a(t)dt
\]  

(2.6)

By equating them we get

\[
s\tilde{x} = x_0 + v_0 \cdot (t_1 - t_0) + \int_{t_0}^{t_1} a(t)dt.
\]  

(2.7)
As shown, for 6DOF both $s$ and $v_0$ can be determined from a single feature observation and 3 views. $x_0$ can be set to 0. It holds

$$s x_1 = v_0 \cdot (t_1 - t_0) + \int_{t_0}^{t_1} a(t)dt$$

$$s x_2 = v_0 \cdot (t_2 - t_0) + \int_{t_0}^{t_2} a(t)dt$$

$$\Rightarrow \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix} = \begin{pmatrix} (t_0 - t_1) \\ (t_0 - t_2) \end{pmatrix} \cdot \begin{pmatrix} s \\ v_0 \end{pmatrix} = \begin{pmatrix} \int_{t_0}^{t_1} a(t)dt \\ \int_{t_0}^{t_2} a(t)dt \end{pmatrix}. \tag{2.8}$$

Closed-form Solution (general case)

Consider $N$ feature observations and 6DOF case. Can be used to initialize filter and smoothers. One can show that a linear system of equations can be achieved and solved using the pseudoinverse:

$$AX = S, \tag{2.9}$$

where $X$ is the vector of unknowns (3D point distances, absolute scale, initial velocity, gravity vector, biases). $A$ and $S$ contain 2D feature coordinates, acceleration, and angular velocity measurements.

Different Paradigms

<table>
<thead>
<tr>
<th>Filtering</th>
<th>Fixed-lag Smoothing</th>
<th>Full smoothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only updates the most recent states</td>
<td>Optimizes window of states</td>
<td>Optimize all states</td>
</tr>
<tr>
<td>• (e.g., extended Kalman filter)</td>
<td>• Marginalization</td>
<td>• Nonlinear Least squares optimization</td>
</tr>
<tr>
<td></td>
<td>• Nonlinear least squares optimization</td>
<td></td>
</tr>
</tbody>
</table>

×1 Linearization

×Accumulation of linearization errors

×Gaussian approximation of marginalized states

✓Fastest

✓Re-Linearize

×Accumulation of linearization errors

×Gaussian approximation of marginalized states

✓Fast

✓Re-Linearize

✓Sparse Matrices

✓Highest Accuracy

×Slow (but fast with GTSAM)

E.g. ROVIO, minimizes the photometric error instead of the reprojection error.
2.5.3 Filtering: Problems

- Wrong linearization point: linearization depends on the current estimates of states, which can be wrong.
- Complexity of the EKF grows quadratically in the number of landmarks. Few landmarks are usually tracked to allow real-time operation.
- Alternative: MSCKF: keeps a window of recent states and updates them using EKF. Incorporate visual observation without including point positions into the states.

2.5.4 Maximum A Posteriori (MAP) Estimation

This corresponds to fusion solved as a non-linear optimization problem. Increased accuracy over filtering methods. We have

\[ x_k = f(x_{k-1}), \quad z_k = h(x_{i_k}, l_{i_j}), \]  

(2.10)

where \( X \) are the robot states, \( L \) the 3D points and \( Z \) the features and IMU measurements. It holds

\[ \{X^*, L^*\} = \arg\max_{X,L} P(X,L|Z) \]
\[ = \arg\min_{X,L} \left\{ \sum_{k=1}^{N} ||f(x_{k-1}) - x_{k}||^2_{\Lambda_k} + \sum_{i=1}^{M} ||h(x_{i_k}) - z_i||^2_{\Sigma_i} \right\} \]  

(2.11)

An open problem is consistency:

- Filters: Linearization around different values of the same variable may lead to error.
- Smoothing methods: may get stuck in local minima.

2.6 Camera-IMU calibration

**Goal:** Estimate the rigid body transformation \( T_{BC} \) and delay \( t_d \) between a camera and an IMU rigidly attached. Assume that the camera has already been intrinsically calibrated.

**Data:** Image points of detected calibration pattern and IMU measurements (accelerometer and gyroscope).

**Approach:** Minimize a cost function

\[ J(\theta) = J_{\text{feat}} + J_{\text{acc}} + J_{\text{gyro}} + J_{\text{biasacc}} + J_{\text{biasgyro}}, \]  

(2.12)

using e.g. Levenberg-Marquardt.
2.7 Understanding Check

Are you able to answer the following questions?

- Why should we use an IMU for Visual Odometry?
- Why not just an IMU?
- How does a MEMS IMU work?
- What is the drift of an industrial IMU?
- What is the IMU measurement model?
- What causes the bias in an IMU?
- How do we model the bias?
- How do we integrate the acceleration to get the position formula?
- What is the definition of loosely coupled and tightly coupled visual inertial fusions?
- How can we use non-linear optimization-based approaches to solve for visual inertial fusion?