Elective in Robotics

coordinator: Prof. Giuseppe Oriolo

Monocular Visual Odometry

(slides prepared by Luca Ricci)
Monocular vs. Stereo: examples from Nature

Predator

• Predators’ eyes face forward.
• The field of vision for each eye overlaps in the front to create binocular vision (stereo vision).

Prey

• A prey animal has eyes that face sideways.
• Only a small area of overlap between the field of vision for each eye (monocular vision).
# Monocular vs. Stereo: examples from Nature

## Monocular vision salient features:
- Increased field of view (FOV)
- Limited depth perception

<table>
<thead>
<tr>
<th>Type of vision</th>
<th>Monocular vision</th>
<th>Stereoscopic vision</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOV (field of vision)</td>
<td>Wide</td>
<td>Narrow</td>
</tr>
<tr>
<td>FOV by right and left eyes</td>
<td>does not overlap</td>
<td>overlapping</td>
</tr>
<tr>
<td>Estimation of distances</td>
<td>Cannot estimate distances accurately</td>
<td>Can estimate distances accurately</td>
</tr>
</tbody>
</table>
Monocular vs. Stereo: examples from Nature

**Monocular vision salient features:**
- Increased field of view (FOV)
- Limited depth perception

The reasons why I won’t ask you to close an eye...

- Accommodation
- Familiar size
- Occlusion
- Motion parallax
- Texture gradient
- Fish-eye effect
- Perspective
- Aerial perspective
Monocular vs. Stereo: recovering depth

<table>
<thead>
<tr>
<th>In Nature</th>
<th>Pigeons have monocular vision rather than binocular vision. Bobbing their heads generates parallax motion for depth perception.</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Geometry</td>
<td>There exists a precise geometric relation between the projections of the same point in 2 different views</td>
</tr>
<tr>
<td>In Computer Vision</td>
<td>Parallel Tracking and Mapping (PTAM), an effective implementation of a monocular visual SLAM algorithm</td>
</tr>
</tbody>
</table>
In Geometry: Epipolar Geometry

\[ \lambda_1 x_1 = X_1 \]

\[ \lambda_2 x_2 = X_2 \]

\[ X_2 = RX_1 + T \quad \text{or} \quad \lambda_2 x_2 = R \lambda_1 x_1 + T \]
In Geometry: Pin-hole Camera Model

Frontal pinhole

\[ X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow x = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \]

\[ Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \]

\[ K_f \quad \Pi_0 \]
In Geometry: The Epipolar Constraint

1. \( \lambda_2 x_2 = R \lambda_1 x_1 + T \)

2. \( \lambda_2 \hat{T} x_2 = \hat{T} R \lambda_1 x_1 \)

Epipolar constraint

\[ x_2^T \hat{T} R x_1 = 0 \]

Essential matrix

\[ E = \hat{T} R \]

Epipolar geometry entities

- \((O_1, O_2, X) \rightarrow\) Epipolar plane
- \(O_1, O_2 \rightarrow\) Baseline
- \(l_1, l_2 \rightarrow\) Epipolar lines
- \(e_1, e_2 \rightarrow\) Epipoles
In Geometry: The Essential Matrix

Essential matrix salient features:

- A special $3 \times 3$ matrix

$$E = \left\{ \hat{T}R \mid R \in SO(3), \ T \in \mathbb{R}^3 \right\}$$

- Apparently 8 dof (9 matrix elements up to scale)
- Practically 5 dof (3 – rotation, 2 – translation up to scale)

How to estimate an essential matrix from a pair of views

(8 – point algorithm | 5 – point algorithm)

1. Rewrite


$$a = [x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2]^T$$

2. Collect constraints from all points

$$\chi E^S = 0$$

$$\chi = [a^1, \ldots, a^n]^T$$

Need parallax motion

$T = 0$ won’t work!
In Geometry: The Essential Matrix

Essential matrix salient features:

- A special 3 x 3 matrix
  \[ E = \left\{ \hat{T}R \mid R \in SO(3), T \in \mathbb{R}^3 \right\} \]
- Apparently 8 dof (9 matrix elements up to scale)
- Practically 5 dof (3 – rotation, 2 – translation up to scale)

How to estimate an essential matrix from a pair of views
(8 – point algorithm | 5 – point algorithm)

Therefore...

Set of feature correspondences \[\rightarrow\] Essential matrix estimation \[\rightarrow\] Camera pose
**In Computer Vision:**

**Monocular Visual Odometry vs Monocular Visual SLAM**

<table>
<thead>
<tr>
<th>Monocular Visual Odometry</th>
<th>Monocular Visual SLAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Feature matching between image frame:</strong></td>
<td><strong>Feature matching between current image frames and a live map:</strong></td>
</tr>
<tr>
<td>• Faster: works in constant time.</td>
<td>• Slower: but accurate.</td>
</tr>
<tr>
<td>• Accumulated small errors will cause drifts.</td>
<td>• Repeated observation of the same features ensures no drifts in trajectory estimate.</td>
</tr>
<tr>
<td>• Cannot maintain a consistent scale from couples of frames (need 3 or more views).</td>
<td>• Scale fixed once set the map.</td>
</tr>
<tr>
<td>• Motion singularities (pure rotations do not constraint enough the motion).</td>
<td>• Extra cost for expanding and maintaining the map.</td>
</tr>
<tr>
<td></td>
<td>• Method based on EKF are limited by the size of the map.</td>
</tr>
</tbody>
</table>
Main features:

- Monocular visual SLAM algorithm
- Intended for small workspace AR (Augmented Reality) applications
- Mapping and Tracking are separated and run in two parallel threads
- Mapping is based on keyframes
- New points are initialized with an epipolar search
- No feature or map uncertainties model (bundle adjustment on a vast number of image features)
- Robust against partial camera occlusions (50% of features available)
PTAM: what is a map

A collection of M map points and N keyframes

- Map point: a 3D point in the world \( p_{jw} = (x_{jw}; y_{jw}; z_{jw}; 1)^T \)
- Keyframe: a pyramid of greyscale 8bpp images (i.e. 640 x 480, ..., 80 x 60) and an associated camera-centred coordinate frame \( (K_i) \)
PTAM: what is a map

Map initialization:

1. Acquire the first keyframe
PTAM: what is a map

Map initialization:

1. Acquire the first keyframe

2. Translate and rotate the camera while tracking the features (hyp. 10 cm translation)
PTAM: what is a map

Map initialization:

1. Acquire the first keyframe

2. Translate and rotate the camera while tracking the features (hyp. 10 cm translation)

3. Acquire the second keyframe and build the map (epipolar search)
PTAM: the tracker

New camera frame

TRACING THREAD

Map available?

TRUE

Apply camera motion model

Track the map

Update camera motion model

AssessTrackingQuality

Get lost?

TRUE

Relocaliser

FALSE

Need New Keyframe?

TRUE

To Map Maker

FALSE

...back to the top
PTAM: more about the tracker

Map point search
Fixed - range image search around the point’s predicted image location

1. Map point in original keyframe (first spotted)

2. Fixed range patch extracted from original image ( 8 x 8 )

3. Affine warp of the patch based on motion model pose estimate

4. Projection on the current view based on the motion model pose estimate

5. Confrontation with the current view (Sum of Squared Differences score)

Examples of affine warping
PTAM: the map maker

...slow but accurate!

MAPPING THREAD
MapMaker::run()

New Keyframe?

Locally converged?

Globally converged?

Add keyframe to the map

Add map points via epipolar search

Set Global and Local convergence to FALSE

Check map points and trash bad ones

Local Bundle adjustment

Global Bundle adjustment

...back to the top
PTAM: monocular SLAM on a quadrotor?

Some basic facts:
- Hummingbird quadrotor equipped with a monocular camera facing downwards
- Multi-Robot-Integration Platform (MIP) implements quadrotor communication, wireless camera sensor interface and some other useful stuff

Passing camera frames from MIP to PTAM will do the trick...

MIP
- WIRELESS CAMERA
- HUMMINGBIRD

PTAM
- MAPPING
- TRACKING

Camera frames

Visual odometry
**MIP: an overview**

**A C++ software aimed to develop control and estimation robotics algorithms**
- Good level of modularity
- Use of abstracted low-level interfaces
- Interface with 3D simulation environment (Player/Gazebo)

### MIP components

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main</strong></td>
<td>main of the program. Here is created and launched the Scheduler</td>
</tr>
<tr>
<td><strong>Baselib</strong></td>
<td>basic library for general purpose and robotics functionalities, e.g. IP communication, pose class, matrix class,...</td>
</tr>
<tr>
<td><strong>Resources</strong></td>
<td>class providing interface modules respect to the hardware or the MIP platform facilities, e.g. camera, quadrotor, keyboard, ...</td>
</tr>
<tr>
<td><strong>Algorithms</strong></td>
<td>class collection of robotics algorithms, e.g. Visual odometry algorithm, estimate (Kalman filtering)</td>
</tr>
<tr>
<td><strong>Tasks</strong></td>
<td>High level robot activities that must be execute in parallel, glueing algorithms and resources, e.g. tracking, deployment, target navigation, mutual localization, entrapment, exploration,...</td>
</tr>
</tbody>
</table>
MIP: an overview

Configuration file
- Resource 1 (options)
  - Task 1 (options)
  - ...

Resourses
- Resource 1 (quadrotor)
- ...
- Resource M (camera)

Tasks
- Task 1
  - Run func.
  - Exe. time
- ...
- Task N
  - Run func.
  - Exe. time

Scheduler
- Task1 run()
- Task2 run()
- TaskN run()

Execution cycle

Algorithms
- Algorithm 1 (Visual SLAM)
- ...
- Algorithm K (EKF)
MIP: use with PTAM and quadrotor

- **QUADROTOR** (simulated)
- **CAMERA** (simulated)
- **CLOCHE KEYBOARD HIT** (pilot quadrotor via keyboard)
- **UAV LOCALIZATION** (through PTAM)
- **VISUAL ODOMETRY** (communicate with PTAM)

Legend:
- **TASKS** = Red
- **RESOURCES** = Blue
- **ALGORITHMS** = Orange

- Sensor data (e.g. IMU)
- Camera frame
- Camera Pose
- Camera frame
- Camera Pose
- Control
MIP: use with PTAM and quadrotor

1. Start 3D simulation environment (Player/Gazebo)
MIP: use with PTAM and quadrotor

1. Start 3D simulation environment (Player/Gazebo)

2. Select the configuration file and run MIP

```
RESOURCES
  Keyboard
  UavPlayerUav
  -usePTAMM false
  CameraPlayer

TASKS
  KeyboardHit
  ClocheTask
  UAVPositioningSystem

  -debugLevel 0
```
MIP: use with PTAM and quadrotor

1. Start 3D simulation environment (Player/Gazebo)

2. Select the configuration file and run MIP

3. Start PTAM and navigate

