Visual scene real-time analysis for Intelligent Vehicles: Visual ego-Localization

Pr. Fabien Moutarde
Center for Robotics
MINES ParisTech
PSL Université Paris

Fabien.Moutarde@mines-paristech.fr
http://people.mines-paristech.fr/fabien.moutarde

Acknowledgements

During preparation of these slides, I borrowed significant slide content from several sources, in particular:

- Davide Scaramuzza (University of Zurich, Robotics and Perception Group, rpg.ifi.uzh.ch): slides on « Visual Odometry and SLAM » from his IROS’2016 tutorial
  https://www.rsj.or.jp/databox/international/iros16tutorial_2.pdf
- Juan D. Tardos (Univ. Zaragoza): slides on « Feature-based visual SLAM » from his ICRA’2016 tutorial
  http://www.dis.uniroma1.it/~labrococo/tutorial_icra_2016/
- Akihiko Torii (Tokyo Tech): slides on « Visual place recognition » from his CVPR’2015 tutorial
  https://sites.google.com/site/lsvpr2015/placerecognition
Outline

- Definition, motivations, principle & methods overview
- Visual Odometry
- Tracked Features: visual keypoints
- Visual SLAM
- Place visual recognition
- Visual ego-localization with Deep-Learning
**Visual ego-localization motivations**

- GPS not always available (indoor, tunnels, underground parkings, « urban canyons »)
- GPS precision quite low (up to 10m error! [except for differential GPS])
- GPS directly provides position but NOT the orientation (only the local orientation of TRAJECTORY can be estimated over time)
- Odometry is quite imprecise (cf. wheel slip!), and subject to large rapid cumulative errors
- Inertial Measurement Unit (IMU) expansive if precise, and subject to cumulative errors

**Visual ego-localization approaches**

- Triangulation of visual geo-tagged landmarks
- Visual Odometry
- Visual SLAM
- Place visual recognition
  + Visual ego-localization by keypoints matching with geo-tagged images
But difficult to have enough pre-recorded and geo-tagged landmarks everywhere!

Outline

• Definition, motivations, principle & methods overview
• **Visual Odometry**
• Tracked Features: visual keypoints
• Visual SLAM
• Place visual recognition
• Visual ego-localization *with Deep-Learning*
Visual Odometry (VO) = estimating ego-movement of camera by analysis of the video

• Possible only if:
  – Sufficient overlap between consecutive frames
  – Dominance of static scene over moving objects in the Field of View
  – Enough texture to allow extraction of apparent motion

Visual Odometry working principle

1. Compute the relative motion $T_k$ from images $I_{k-1}$ to image $I_k$
   \[ T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \]

2. Concatenate them to recover the full trajectory
   \[ C_n = C_{n-1}T_n \]

3. An optimization over the last $m$ poses can be done to refine locally the trajectory (Pose-Graph or Bundle Adjustment)
Visual Odometry:
estimation of relative motion

\[ T_k = \arg \min \int \int \rho \left[ l_k \left( \pi \left( T \cdot \pi^{-1}(u), d_u \right) \right) - l_{k-1}(u) \right] du \]

Visual Odometry flow chart

- Image sequence
- Feature detection
- Feature matching (tracking)
- Motion estimation
  - 2D-2D
  - 3D-3D
  - 3D-2D
- Local optimization

Front-end

Back-end

Davide Scaramuzza – University of Zurich – Robotics and Perception Group - rpg.ifi.uzh.ch
Outline

- Definition, motivations, principle & methods overview
- Visual Odometry
- Tracked Features: visual keypoints
- Visual SLAM
- Place visual recognition
- Visual ego-localization with Deep-Learning

Features: visual keypoints

What are Good Features to Track?

Which of the patches below can be matched reliably?

⇒ Choice of keypoints detector is critical!

Corners vs Blob Detectors

- A corner is defined as the intersection of one or more edges
  - A corner has high localization accuracy
  - Corner detectors are good for VO
  - It’s less distinctive than a blob
  - E.g., Harris, Shi-Tomasi, SUSAN, FAST

- A blob is any other image pattern, which is not a corner, that significantly differs from its neighbors in intensity and texture
  - Has less localization accuracy than a corner
  - Blob detectors are better for place recognition
  - It’s more distinctive than a corner
  - E.g., MSER, LOG, DOG (SIFT), SURF, CenSurE

Descriptor: Distinctive feature identifier
- Standard descriptor: squared patch of pixel intensity values
- Gradient or difference-based descriptors: SIFT, SURF, ORB, BRIEF, BRISK

Harris corners detector (1988)

- How do we identify corners?
- We can easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity in at least 2 directions

“flat” region: no intensity change
“edge”: no change along the edge direction
“corner”: significant change in at least 2 directions
FAST corners detector

**FAST = Features from Accelerated Segment Test**

[Rosten et al., PAMI 2010]

- Studies intensity of pixels on circle around candidate pixel $C$
- $C$ is a FAST corner if a set of $N$ contiguous pixels on circle are:
  - all brighter than $\text{intensity}_C + \text{threshold}$, or
  - all darker than $\text{intensity}_C + \text{threshold}$

- Typical FAST mask: test for 9 contiguous pixels in a 16-pixel circle
- Very fast detector - in the order of 100 Mega-pixel/second

Keypoints DESCRIPTORS for Visual Odometry

**BRIEF descriptor** [Calonder et. al, ECCV 2010]

- **Binary Robust Independent Elementary Features**
- Goal: high speed (in description and matching)

- **Binary** descriptor formation:
  - Smooth image
  - for each detected keypoint (e.g. FAST),
  - sample 256 intensity pairs $p = (p_1, p_2)$ within a squared patch around the keypoint
  - for each pair $p$
    - if $p_1 < p_2$ then set bit $p$ of descriptor to 1
    - else set bit $p$ of descriptor to 0

- The pattern is generated randomly only once; then, the same pattern is used for all patches

- Not scale/rotation invariant
- Allows very fast Hamming Distance matching: count the number of bits that are different in the descriptors matched

**ORB descriptor** [Rublee et al., ICCV 2011]

- Oriented FAST and Rotated BRIEF
- Alternative to SIFT or SURF, designed for fast computation
- Keypoint detector based on FAST
- BRIEF descriptors are steered according to keypoint orientation (to provide rotation invariance)
- Good Binary features are learned by minimizing the correlation on a set of training patches
Outline

• Definition, motivations, principle & methods overview
• Visual Odometry
• Tracked Features: visual keypoints
• Visual SLAM
• Place visual recognition
• Visual ego-localization with Deep-Learning

Visual ego-localization using SLAM

SLAM = Simultaneous Localization and Mapping
• Progressive creation of a « local » map within which the localization is computed
• Can be done with vision, but also with other sensor (in particular laser/LIDAR)
Visual Odometry vs Visual SLAM

- **Visual Odometry**
  - Focus on incremental estimation/local consistency

- **Visual SLAM**: Simultaneous Localization And Mapping
  - Focus on **globally consistent** estimation
  - **Visual SLAM** = visual odometry + **loop detection** + **graph optimization**

- The choice between VO and V-SLAM depends on the tradeoff between performance and consistency, and simplicity in implementation.

- VO trades off consistency for real-time performance, without the need to keep track of all the previous history of the camera.

---

**Feature-based Visual SLAM**

States

- \( x_{wj} \in \mathbb{R}^3 \) Coordinates of point \( j \)
- \( T_{iw} \in \text{SE}(3) \) Pose of camera \( i \)

Measurements

- \( u_{ij} = \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} \) Observation of point \( j \) from camera \( i \)

Reprojection error

- \( e_{ij} = u_{ij} - \pi_i(T_{iw}, x_{wj}) \)

Projection Function

---

From Juan D. Tardos (Univ. Zaragoza) slides
Projection of point $j$ on camera $i$

$$T_{iw} \in SE(3) \left\{ \begin{array}{c}
R_{iw} \in SO(3) \ \\
t_{iw} \in \mathbb{R}^3
\end{array} \right.$$  

**Camera pose**

Coordinates of point $j$ w.r.t. camera $i$

$$x_{ij} = R_{iw}x_{wj} + t_{iw}$$

![Camera projection diagram]

- In summary:

$$\pi_i(T_{iw}, x_{wj}) = \begin{bmatrix} f_i u z_{ij} + c_{i,u} \\ f_i v z_{ij} + c_{i,v} \end{bmatrix}$$

From Juan D. Tardos (Univ. Zaragoza) slides

---

Minimization of reprojection errors

Find the state values minimizing the reprojection errors:

$$e_{i,j} = u_{i,j} - \pi_i(T_{iw}, x_{wj})$$

Bundle Adjustment

$$\{T_{1w} \ldots T_{nw}, x_{w1} \ldots x_{wm}\}^* = \text{arg min}_{T,x} \sum_{i,j} \rho_h(e_{i,j}^T \Sigma_{i,j}^{-1} e_{i,j})$$

$$\Sigma_{i,j} = \sigma_{i,j}^2 I_{2 \times 2} \quad \text{std. dev. typically = 1 pixel * scale}$$

where $\rho_h(\cdot)$ robust cost function (i.e. Huber cost) to downweight wrong matchings

From Juan D. Tardos (Univ. Zaragoza) slides
**Bundle Adjustment (BA)**

\[
\{T_{1w} \ldots T_{nw}, x_{w1}, \ldots, x_{wm}\}^* = \arg \min_{T,x} \sum_{i,j} \rho_h(e_{ij}^T \Sigma_i^{-1} e_{ij})
\]

- The problem is sparse
  - Not all cameras see all points!
- But still not feasible in real time
  - example: 1k images and 100k points → 1s per LM iteration
- Local BA or sliding-window BA
- BA requires very good initial solutions

- **EKF approach**
  - Only keeps the last pose
  - \(O(n^2)\) with the number of features
  - Limited to 200-300 features in real-time
- **Keyframe approach (PTAM)**
  - Uses only a few keyframes for map estimation with non-linear optimization
  - Can handle thousands of points
  - Given the same computational effort is more precise than EKF-SLAM

---

**Choice of features (keypoints type)**

<table>
<thead>
<tr>
<th>Detector</th>
<th>Descriptor</th>
<th>Rotation Invariant</th>
<th>Automatic Scale</th>
<th>Accuracy</th>
<th>Relocation &amp; Loops</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harris</td>
<td>Patch</td>
<td>No</td>
<td>No</td>
<td>++++</td>
<td>-</td>
<td>++++</td>
</tr>
<tr>
<td>Shi-Tomasi</td>
<td>Patch</td>
<td>No</td>
<td>No</td>
<td>++++</td>
<td>-</td>
<td>++++</td>
</tr>
<tr>
<td>SIFT</td>
<td>SIFT</td>
<td>Yes</td>
<td>Yes</td>
<td>++</td>
<td>++++</td>
<td>+</td>
</tr>
<tr>
<td>SURF</td>
<td>SURF</td>
<td>Yes</td>
<td>Yes</td>
<td>++</td>
<td>++++</td>
<td>++</td>
</tr>
<tr>
<td>FAST</td>
<td>BRIEF</td>
<td>No</td>
<td>No</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>ORB</td>
<td>ORB</td>
<td>Yes</td>
<td>No</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
</tr>
</tbody>
</table>

From Juan D. Tardos (Univ. Zaragoza) slides

As for Visual Odometry, usual SIFT or SURF are clearly not the best choice!
Feature matching

- Compare descriptors
- Spurious matchings
  - Least-squares is very sensitive to spurious data
  - A single spurious match may ruin the estimation

→ Search for consensus with a robust technique: RANSAC
  From Juan D. Tardos (Univ. Zaragoza) slides

RANSAC

= RANdom SAmple Consensus

- General statistical method for robust estimation in presence of outliers

- Principle: iterate the steps below
  - randomly select a subset of points to estimate a model
  - compute the # of other points compatible with the model
  - If enough inliers, re-estimate model with all, and compute error
  - If lower than current-best, replace it with new model (& inliers)
Loop closure

• Loop closing problem
  SLAM is working, and you come back to a previously mapped area
  ➢ Loop detection: to avoid map duplication
  ➢ Loop correction: to compensate the accumulated drift

Correction after loop closure

Requires a place recognition technique

Place recognition also necessary for relocation
(« kidnapped robot » problem, …)

Outline

• Definition, motivations, principle & methods overview
• Visual Odometry
• Tracked Features: visual keypoints
• Visual SLAM
• Place visual recognition
• Visual ego-localization with Deep-Learning
Coarse visual localization

This is an Image Retrieval particular case

Image Retrieval with BoVW

Offline
1. Feature detection & description
2. Training visual vocabulary
3. Image description
4. Feature detection & description
5. Image description
6. Initial ranking
7. Re-ranking with geometric verification
Bag-of-Visual-Words (BoVW)

Inspired from text analysis in which a piece of text is represented by a sparse vector of the number of occurrences of each word of a dictionary

Adapted to images using *keypoints descriptors* as a representation of image content:

- descriptor vectors are quantized (usually by K-means partitioning) into a codebook of « visual words »
- An (sub-)image is represented by an histogram of codebook occurrences

---

Image description

1. Feature detection & description, e.g. SIFT
2. Represent the set of features
   - as a sparse histogram (BoVW)
   - as an aggregated vectors (VLAD, FV)

From Akihiro Torii (Tokyo Tech) slides
Building visual « vocabulary » (offline)

Detect & describe features (e.g. SIFT)

Find representative features (e.g. approximate k-means)

A B C D E

Visual words (centroids)

Feature (128D)
Centroid (128D)

Computing BoVW sparse histogram for an image

For each image,
1. we assign features to the visual word closest in the feature space.
2. we build a histogram by voting.

Feature (128D)
Centroid (128D)

Histogram

Visual words (centroids)

tf-idf weighted and L2 normalized

Design of the weights directly impacts the retrieval results!

See also [Jegou-CVPR09, Zheng-CVPR13]
**TF-IDF**

**Term Frequency-Inverse Document Frequency**

For a term $i$ in document $j$:

$$w_{i,j} = tf_{i,j} \times \log \left( \frac{N}{df_i} \right)$$

- $tf_{i,j}$ = number of occurrences of $i$ in $j$
- $df_i$ = number of documents containing $i$
- $N$ = total number of documents

**Giving larger weights to uncommon words**

---

**Alternative to BoVW: VLAD**

**Vector of Locally Aggregated Descriptors**

[Jégou et al. 2010]

Sum of residual vectors assigned to each centroid

- Concatenate these vectors

**VLAD = 128 x #centroids**

Signed square root + L2 normalization

Use a sum of residuals instead of a TF-IDF histogram
**Retrieval from query:**
 inverted file indexing

Query:

```
A B C D E F G H I J K L M N O
```

**Count matches**

- **A:** {2} → 1
- **C:** {2} → 2
- **E:** {1, 2} → 2
- **F:** {1, 3} → 0
- **K:** {3} → 0
- **N:** {1, 3} → 0

**Score**

```
Can be weighted
```

**Can be evaluated**

Not even evaluated

---

**Re-ranking by geometric verification**

**Query image**

**Step 1**: Initial ranking/shortlisting

**Step 2**: Re-ranking to improve the list

- Generate tentative matches of features
- Verify the tentative matches by fitting geometric transformations (affine, homography, ...), i.e. RANSAC
- Re-rank the shortlisted images by the number of verified matches

---

From Akihiro Torii (Tokyo Tech) slides
• Estimation of translation+rotation by multiple matches of keypoint descriptors (SIFT, SURF, ORB, vLAD, …) between query and match
• Requires elimination of outliers by RANSAC

Outline

• Definition, motivations, principle & methods overview
• Visual Odometry
• Tracked Features: visual keypoints
• Visual SLAM
• Place visual recognition
• Visual ego-localization with Deep-Learning
PoseNet: 6-DoF camera pose regression with Deep-Learning

![Image](image_url)

Input RGB Image ➔ Convolutional Neural Network (GoogLeNet) ➔ 6-DOF Camera Pose

Trained with a naïve end-to-end loss function to regress camera position, $x$, and orientation, $q$

$$\text{loss}(I) = \|x - \hat{x}\|_2 + \beta \left\| q - \frac{\hat{q}}{\|\hat{q}\|_2} \right\|_2$$


**PoseNet applies Transfer learning for a task totally different from classification!**

By removing last layer(s) (those for classification) of a convNet trained on ImageNet, one obtains a transformation of any input image into a semi-abstract representation, which can be used for learning SOMETHING ELSE (« transfer learning ») by creating new convNet output and perform learning of new output layers + fine-tuning of re-used layers
PoseNet training data and test results

training data in green, test data in blue, PoseNet results in red


PoseNet results on other tests

Figure 4: Map of dataset showing training frames (green), testing frames (blue) and their predicted camera pose (red). The testing sequences are distinct trajectories from the training sequences and each scene covers a very large spatial extent.
## PoseNet results summary

<table>
<thead>
<tr>
<th>Scene</th>
<th># Frames</th>
<th>Spatial Extent (m)</th>
<th>SCoRe Forest (Uses RGB-D)</th>
<th>Dist. to Conv. Nearest Neighbour</th>
<th>PoseNet</th>
<th>Dense PoseNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>King’s College</td>
<td>1220</td>
<td>140 x 40m</td>
<td>N/A</td>
<td>3.34m, 2.96°</td>
<td>1.92m</td>
<td>1.66m, 2.43°</td>
</tr>
<tr>
<td>Street</td>
<td>3015</td>
<td>500 x 100m</td>
<td>N/A</td>
<td>1.95m, 4.51°</td>
<td>3.67m</td>
<td>2.96m, 3.00°</td>
</tr>
<tr>
<td>Old Hospital</td>
<td>895</td>
<td>50 x 40m</td>
<td>N/A</td>
<td>5.38m, 4.51°</td>
<td>2.31m</td>
<td>2.62m, 2.45°</td>
</tr>
<tr>
<td>Shop Façade</td>
<td>231</td>
<td>35 x 25m</td>
<td>N/A</td>
<td>2.10m, 5.20°</td>
<td>1.46m</td>
<td>1.41m, 3.59°</td>
</tr>
<tr>
<td>St Mary’s Church</td>
<td>1487</td>
<td>80 x 60m</td>
<td>N/A</td>
<td>4.48m, 5.65°</td>
<td>2.65m</td>
<td>2.45m, 3.98°</td>
</tr>
</tbody>
</table>

## PoseNet robustness

Tolerance to environment, unknown intrinsics, weather, etc.

### Blur
![Blur Image]

### Occlusion
![Occlusion Image]

### Dusk
![Dusk Image]

### Night
![Night Image]

**PoseNet summary:**

- Robust to lighting, weather, dynamic objects
- Fast inference, <2ms per image on Titan GPU
- Scale not dependent on number of training images

**Coarse accuracy**

**Difficult to learn both position vs orientation**

---


---

**PoseNet: importance of relative weighting of position-orientation errors**

\[
\text{loss}(I) = \| x - \hat{x} \|_2 + \beta \left\| q - \frac{\hat{q}}{\|\hat{q}\|} \right\|_2
\]

---

Median Position Error (m)

<table>
<thead>
<tr>
<th>Beta Weight $\beta$</th>
<th>100</th>
<th>1,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position Error</td>
<td>3.5</td>
<td>2</td>
</tr>
<tr>
<td>Orientation Error</td>
<td>3.0</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Median Orientation Error ($^\circ$)

<table>
<thead>
<tr>
<th>Beta Weight $\beta$</th>
<th>100</th>
<th>1,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position Error</td>
<td>8</td>
<td>6.5</td>
</tr>
<tr>
<td>Orientation Error</td>
<td>8</td>
<td>6.5</td>
</tr>
</tbody>
</table>
PoseNet performance improves with more data

Scales very well:
- Constant inference time (single forward pass of the network)
- Constant memory (~5 MB of neural network weights)


PoseNet: graceful degradation with increased spacing of training images

PoseNet: importance of transfer learning

![Graph showing comparison of different models](image)

**PoseNet vs. traditional methods**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>King’s College</td>
<td>0.88m, 1.04°</td>
<td>0.42m, 0.55°</td>
</tr>
<tr>
<td>Resolution</td>
<td>256 x 256 px</td>
<td>1920 x 1080 px</td>
</tr>
<tr>
<td>Inference Time</td>
<td>2 ms</td>
<td>78 ms</td>
</tr>
</tbody>
</table>

PoseNet less precise, but much faster and can work with much smaller images