Stereo Visual Odometry

Chris Beall
CVPR 2014 Visual SLAM Tutorial
• Motivation
• Algorithm Overview
• Feature Extraction and matching
• Incremental Pose Recovery
• Practical/robustness considerations
• What else do you get?
• Results
Motivation

• Why stereo Visual Odometry?
  • Stereo avoids scale ambiguity inherent in monocular VO
  • No need for tricky initialization procedure of landmark depth
Algorithm Overview

1. Rectification

2. Feature Extraction

3. Stereo Feature Matching

4. Temporal Feature Matching

5. Incremental Pose Recovery/RANSAC
Undistortion and Rectification
Feature Extraction

- Detect local features in each image
  - SIFT gives good results (can also use SURF, Harris, etc.)

Lowe ICCV 1999
Stereo Matching

- Match features between left/right images
- Since the images are rectified, we can restrict the search to a bounding box on the same scan-line
Temporal Matching

- Temporally match features between frame $t$ and $t-1$
Relative Pose Estimation/RANSAC

- Want to recover the incremental camera pose using the tracked features and triangulated landmarks

- There will be some erroneous stereo and temporal feature associations → Use RANSAC
  - Select N out of M data items at random (the minimal set here is 3)
  - Estimate parameter (incremental pose from t-1 to t)
  - Find the number K of data items that fit the model (called inliers) within a given tolerance
  - Repeat S times
  - Compute refined model using full inlier set
Relative Pose Estimation

- Camera pose can be recovered given at least three known landmarks in a non-degenerate configuration.
- In the case of stereo VO, landmarks can simply be triangulated.
- Two ways to recover pose:
  - Absolute orientation
  - Reprojection error minimization.
Absolute Orientation

- Estimate relative camera motion by computing relative transformation between 3D landmarks which were triangulated from stereo-matched features.
Absolute Orientation

• First, in 2D:
  • Given two sets of corresponding points \( \{p^l_i\} \) and \( \{p^r_i\} \) related by a rigid 2D transformation \( T^l_r = (R^l_r, t^l_r) \):
    \[
    p^l = R^l_r p^r + t^l_r
    \]
  • First recover rotation (2 points), then translation (1 point)

B. K. P Horn JOSA 1987
Absolute Orientation

- **Rotation:**
  - Given two point correspondences \((p^l_1, p^r_1)\) and \((p^l_2, p^r_2)\), the vector \(v\) between the two points will be rotated by the desired angle \(\theta\)
  - Specifically, the vectors are related by
    \[
    v^l = (p^l_2 - p^l_1) = R^l_r (p^r_2 - p^r_1) = R^l_r v^r
    \]
  - Finally, recover the angle
    \[
    \theta = \arccos \frac{v^l \cdot v^r}{\|v^l\| \|v^r\|}
    \]
    and translation \(t^l_r = p^l - R^l_r p^r\)
Absolute Orientation

• Now in 3D:
  • Create coordinate frames from three corresponding points
    • Take x-axis by connecting \( p_1^l \) to \( p_2^l \): \( \hat{x}_l \propto (p_2^l - p_1^l) \)
    • Construct y-axis in the plane formed by three points, perpendicular to x-axis: \( \hat{y}_l \propto (p_3^l - p_1^l) - [(p_3^l - p_1^l) \cdot \hat{x}_l] \hat{x}_l \)
    • Complete frame with z-axis: \( \hat{z}_l \propto (\hat{x}_l \times \hat{y}_l) \)
Absolute Orientation

- Normalize the axes and we have the rotation of the frame with respect to the global reference frame,

\[ R^g_l = [ \hat{x}_l \ \hat{y}_l \ \hat{z}_l ] \]

- Repeat for the right frame, and obtain the relative rotation,

\[ R^l_r = R^l_g R^g_r = (R^g_l)^T R^g_r \]

- As in the 2D case, the translation can then be recovered using a single point
Absolute Orientation

- The Absolute Orientation approach assumes a relatively noiseless case, and does not work well otherwise.
- No simple way to average out noisy points by considering more data.
  - Use SVD-based method instead.
  - Use different approach based on projective geometry.

The rotation matrix $R_{lr}$ can be recovered linearly without trigonometric functions using the fact that

$$R_{lr} = \begin{pmatrix} \cos \phi & \sin \phi \\
-\sin \phi & \cos \phi \end{pmatrix} = \begin{pmatrix} c & s \\
-sc & c \end{pmatrix}$$

which yields the following linear system in $c$ and $s$:

$$v_l x v_l y = \begin{pmatrix} c & s \\
-sc & c \end{pmatrix} v_r x v_r y = \begin{pmatrix} v_r x \\
sv_r y + cv_r y \\
v_r y \end{pmatrix}$$

A solution that is symmetric in left and right is the following,

$$c = k v_l x v_r x + v_l y v_r y$$

and

$$s = k v_l x v_r y$$

with $k$ chosen to make $c^2 + s^2 = 1$.

### 2. Estimating rotations in 3D

In 3D, we cannot recover the 3 DOF rotation with a single vector, but we can after we find two corresponding “triads”, i.e., right-handed coordinate frames. Following Horn (1987), instead of two point correspondences we now need three, which we denote by $(p_l^1, p_r^1)$ and $(p_l^2, p_r^2)$ and $(p_l^3, p_r^3)$. We can then create a coordinate frame (say in left) by pointing the x-axis from $p_l^1$ to $p_l^2$, where the proportional sign indicates we normalize $\hat{x}_l$ to have unit norm. We construct the y-axis perpendicular to it in the $p_l^1$ $p_l^2$ $p_l^3$ plane,

$$\hat{y}_l / (p_l^3 - p_l^1) \cdot (p_l^3 - p_l^1) \cdot \hat{x}_l$$

and comple the triad by having a z-axis perpendicular to the first two:

$$\hat{z}_l / (\hat{x}_l \cdot \hat{y}_l)$$

After we do this, the rotation from the left frame to the global frame is simply $R_{gl} = [\hat{x}_l \ \hat{y}_l \ \hat{z}_l]$ and after we repeat the construction for the three points in the right frame, we recover the unknown rotation by simply composing the two rotations:

$$R_{lr} = R_{lg} R_{gr} = (R_{gl})^T R_{gr}$$
Reprojection Error Minimization

- A better approach is to estimate relative pose by minimizing reprojection error of 3D landmarks into images at time $t$ and $t-1$. 

\[ T_t \quad T_{t-1} \]
Practical/Robustness Considerations

• The presented algorithm works very well in feature-rich, static environments, but...

• A few tricks for better results in challenging conditions:
  • Feature binning to cope with bias due to uneven feature distributions
  • Keyframing to cope with dynamic scenes, as well as reducing drift of a stationary camera

• Real-time performance
Feature Binning

- Incremental pose estimation yields poor results when features are concentrated in one area of the image
- Solution: Draw a grid and keep k strongest features in each cell
Challenges: Dynamic Scenes

- Dynamic, crowded scenes present a real challenge
- Cannot depend on RANSAC to always recover the correct inlier set
- Example 1: Large van “steals” inlier set in passing
Challenges: Dynamic Scenes

- Example 2: Cross-traffic while waiting to turn left at light

Only accept incremental pose if:
- translation > 0.5m
- Dominant direction is forward

Without keyframing: Incorrect sideways motion

With keyframing:

Georgia Institute for Robotics and Intelligent Machines
Real-time performance

- Parallelization of feature extraction and stereo matching steps allows real-time performance even in CPU-only implementation.
What else do you get?

• Stereo Visual Odometry yields more than just a camera trajectory!

• Tracked landmarks form a sparse 3D point cloud of the environment
  • Can be used as the basis for localization
What else do you get?

- 3D Point cloud on KITTI Benchmark, Sequence 2

http://www.cvlibs.net/datasets/kitti/
Results on KITTI Benchmark

- Representative results on KITTI VO Benchmark
  - Average translational/rotational errors are very small
  - Accumulate over time, resulting in drift
GTSAM VO Example with Point Cloud

• StereoVOExample_large.m in GTSAM
  • Takes VO output and improves result through bundle adjustment (more on that later!)

tinyurl.com/gtsam