Seminars in Artificial Intelligence and Robotics

Computer Vision for Intelligent Robotics

Hints on Visual Odometry and Visual SLAM

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Alberto Pretto

Some material taken from Davide Scaramuzza's slides "Visual Odometry and SLAM: past, present, and the robust-perception age"

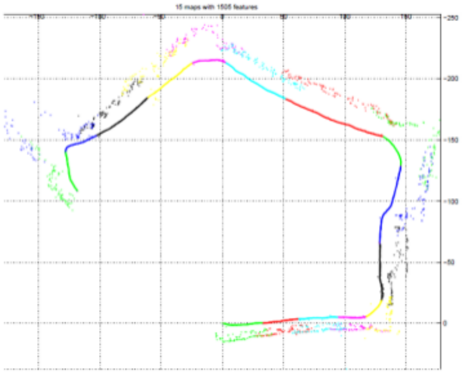
Visual Odometry vs Visual SLAM

Visual Odometry

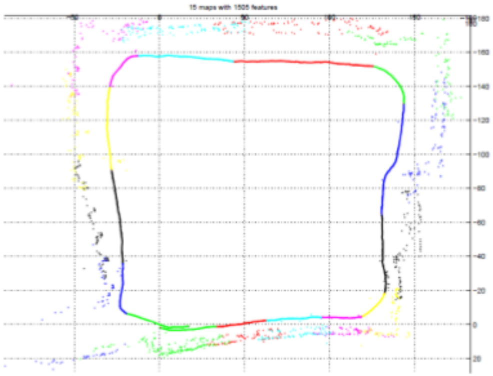
- Focus on incremental estimation/local consistency

Visual SLAM: Simultaneous Localization And Mapping

- Focus on globally consistent estimation
- Visual SLAM = visual odometry + **loop detection** + graph optimization



Visual odometry



Visual SLAM

Image courtesy from [Clemente et al., RSS'07]

Visual Odometry/SLAM approaches

Epipolar geometry

Filtering

Direct approach

Bundle adjustment

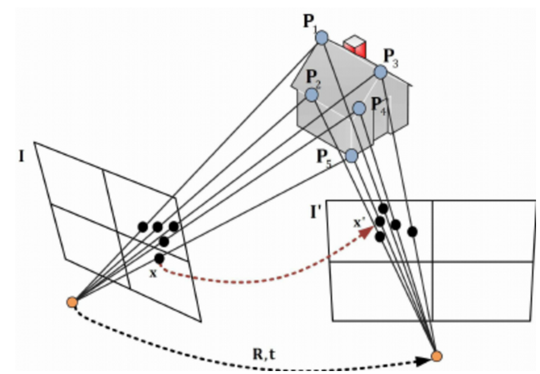
Hybrid approaches

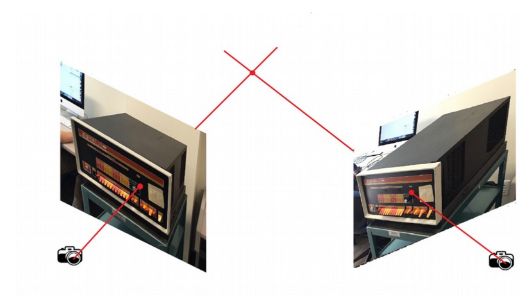
Epipolar geometry

- Extract and match a set of salient features

- Infer ego-motion using the epipolar geometry constraints (essential or fundamental matrix) inside a sample consensus framework (e.g., 8-point algorithm + RANSAC).

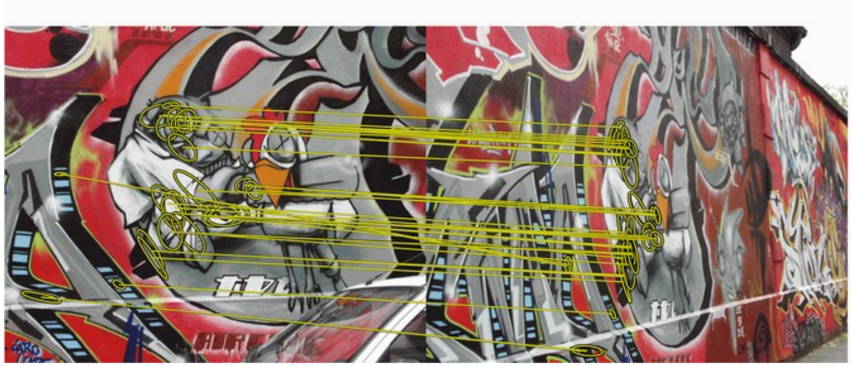
- Estimate 3D features positions using triangulation.





Footnote: matching vs tracking

Matching: extract features for every image, compute a descriptor (signature) for each feature and the match them between images looking for “similar” descriptors.



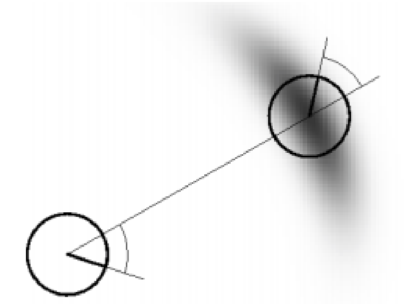
Tracking: a) extract features in the first image b) for each feature, look for a “similar” patch in a neighborhood on the (temporally close) next image; c) update the feature location and repeat from b)

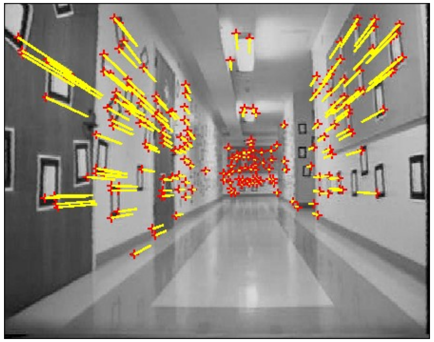
Probabilistic filtering

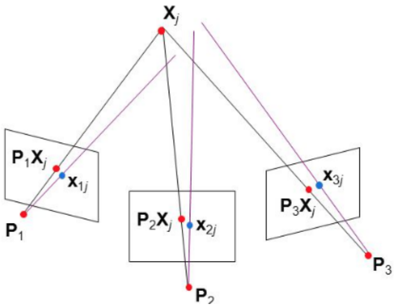
- Define a state that includes camera position and features location, along with other components (velocity, acceleration, etc..) used in the kinematic equations (KE).

-For each image:

  - Track a set of features;
  - Update the state and its uncertainty using the kinematic equations.
  - Predict the position of the tracked features after this motion update and compute the re-projection error.
  - "Correct" the state, e.g. using probabilistic tools (EKF).







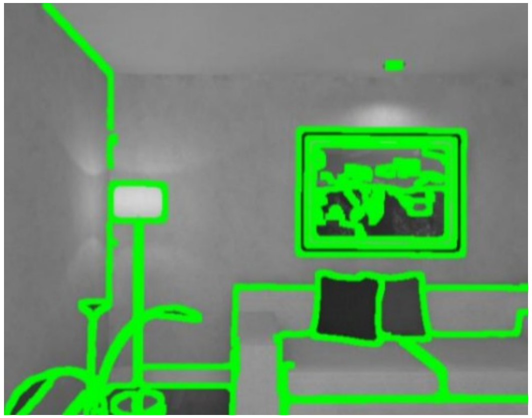
Direct approach

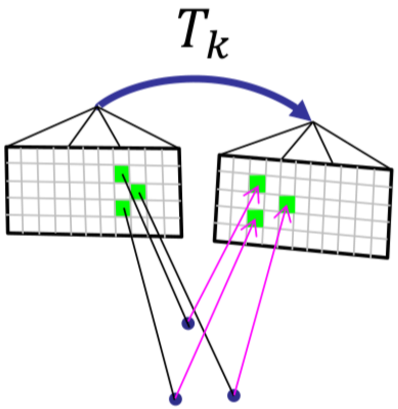
Camera motion and the scene structure are computed directly from image intensity discrepancies using optimization techniques

For each involved pixel:

- Take note of the its previous intensity *y_i*
- Un-project it back to 3D using its current depth guess *d_i*
- Re-project onto the next image using the current transformation *T* guess
- Take note of its current intensity y_i'
- Compute the per-point residual *y_i-y_i'*

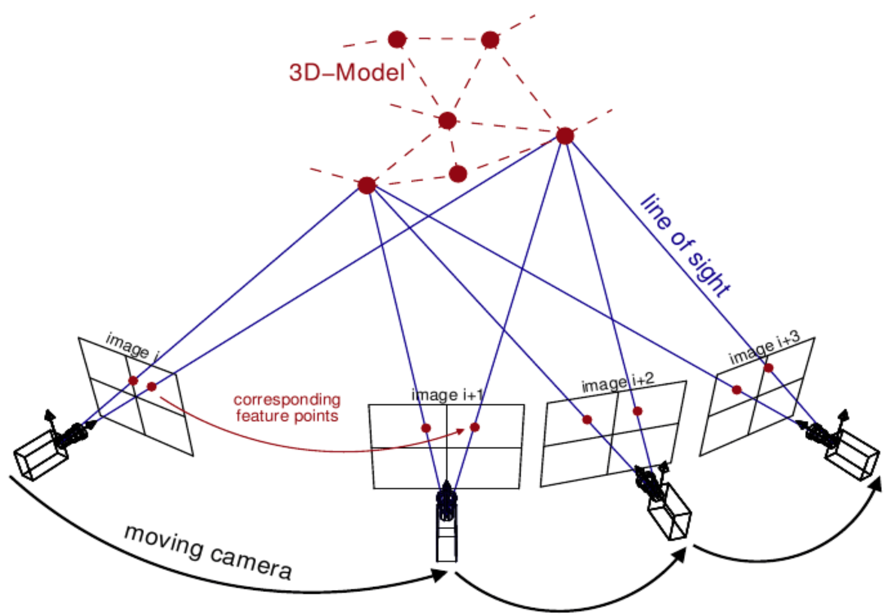
Iteratively optimize *T, d_1, …, d_n* using **least-square** over such residuals.





Bundle adjustment

Given a collection of images along with a set of points matches, estimate the points' 3D positions and **all** the camera displacements minimizing the reprojection error between the given features image locations and predicted image locations



Hybrid approaches

- Compute an initial guess for ego-motion estimation using **epipolar geometry** and/or **filtering approaches**.

- Trigger a “key-frame” every *n* images, or *m* meters, or … and **extract, describe and match** features between the last *k* key-frames.

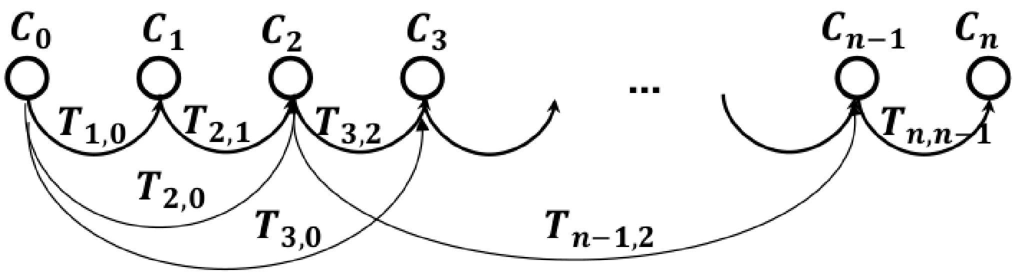
- Refine both the position of the the k key-frames and the feature position by means of **local** bundle-adjustment.

- Update accordingly the current camera position.

- **key-frames used also for loop closure detection**

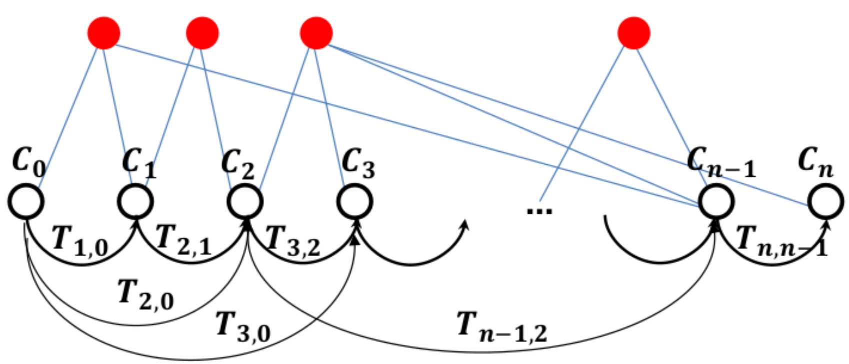
Pose-Graph Optimization vs Bundle Adjustment

Pose-Graph Optimization



$$\sum_i \sum_j \left\| C_i - T_{ij} C_j \right\|^2$$

Bundle Adjustment



$$X^i, C_k = argmin_{X^i, C_k,} \sum_{i,k} \rho_H\left(p_k{}^i - \pi(X^i, C_k)\right)$$