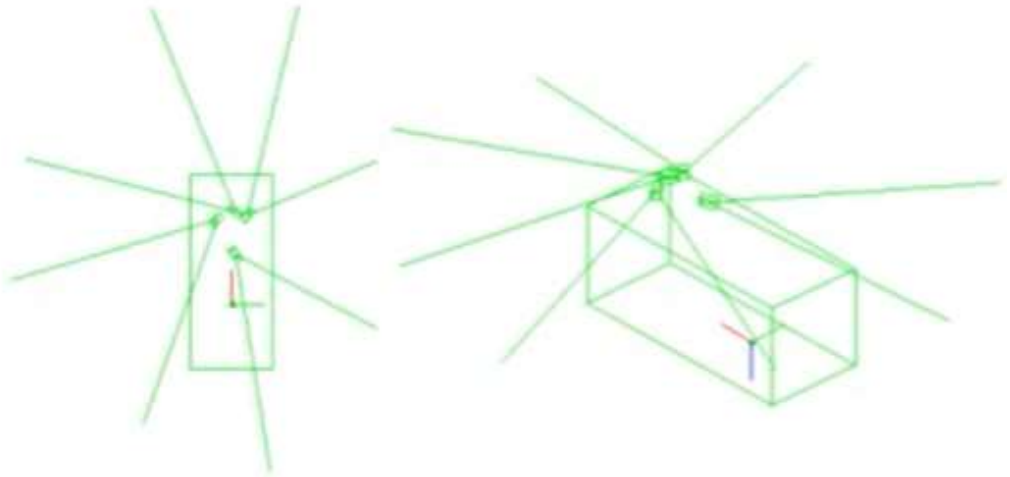
Bundle Adjustment

Example Application for Bundle Adjustment

- A vehicle needs to map its environment that it is moving through, and estimate its self motion
- The main sensors are cameras mounted on the vehicle



Four camera configuration on vehicle

Four cameras are used for looking in the various diagonal directions.

Bundle adjustment

- The most accurate (and flexible) way to estimate structure and motion is to directly minimize the squared reprojectio errors for the 2D points
- We can do this for N camera poses (N≥2)
- The unknowns are:
  - The (X,Y,Z) positions of the K 3D points
  - The relative poses of the camera in N-1 poses
    - The first pose is arbitrary and we can just set it to the origin
  - There are a lot of unknown parameters to find!
- We can use non-linear least squares
  - It will be slow … probably not suitable for real-time
  - We will need a good starting guess

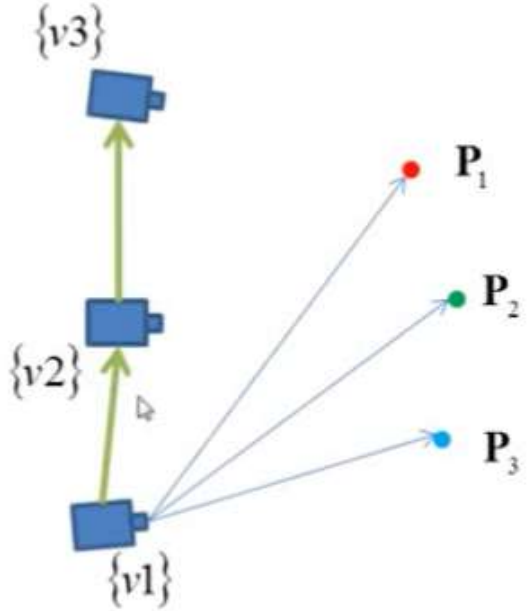
Structure from Motion

- Parameters to be estimated
  - *N*-1 vehicle poses wr.t. 1st vehicle pose; each has 6 degrees of freedom
  - *K* 3D point locations w.r.t. 1st vehicle pose

- Measurements
  - Image point observations

    $$\mathbf{p}_{ij} = \begin{pmatrix} u & v \end{pmatrix}^T, \quad i = 1..N, \; j = 1..K$$

  - Assumptions
    - Calibrated cameras
    - Point correspondences are known



Bundle adjustment

- We need a function that
  - Predicts the image locations of the K 3D points, in each of the N images
  - The inputs to the function are the current guess for the locations of the 3D points, and the N-1 poses
- As we did before, we can take the Jacobian of the function and iteratively add corrections to our guess
- Note - there are somewhat more efficient algorithms than the Newton's method that we used
  - One is called "Levenberg-Marquardt"
    - It adaptively switches between Newton's method and steepest descent
  - It is implemented in Matlab's "lsqnonlin" function
    - This is part of the optimization toolbox

Scale Factor

- Results have an unknown scale factor
  - All points, and all translations are scaled by the same unknown amount
- Possible ways to determine scale factor:
  - Take an picture of a known object in one of the images
  - Use stereo cameras
  - Use some other data source, such as odometry
- Odometry
  - Many vehicles can estimate self-motion from wheel encoders and heading sensors
  - It is fairly accurate over short distances, but can drift over longer distances

Structure from Motion

- Algorithm approach
  - Search for parameters to minimize residual errors between predictions and measurements

$$\text{minimize} \quad E = \sum_i \left\| {}^{v_i}_{v_{i+1}}\boldsymbol{\beta}^{odom} - f\left({}^{v_1}_{v_{i-1}}\boldsymbol{\beta},\ {}^{v_1}_{v_i}\boldsymbol{\beta}\right) \right\|^2_{\Omega_i} + \sum_{i,j} \left\| \mathbf{p}_{ij} - g\left(\mathbf{P}_j,\ {}^{v_1}_{v_i}\boldsymbol{\beta}\right) \right\|^2_{\Xi_j}$$

Predicts odometry measurement

Projects point i into image j to predict image measurement

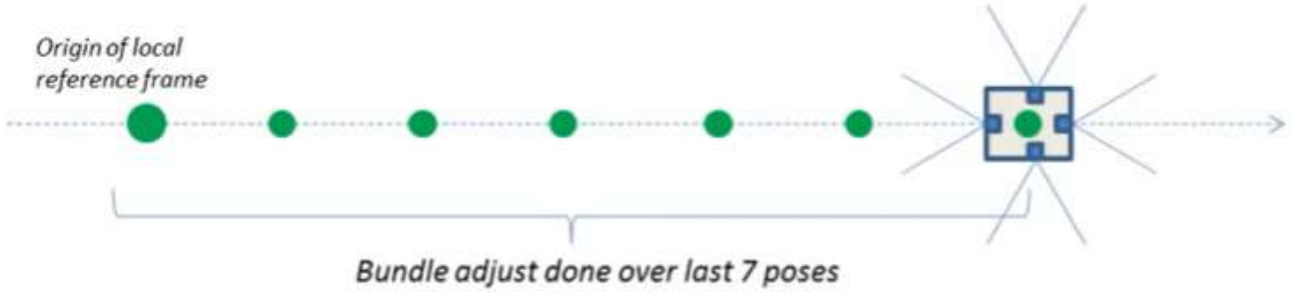
  - where

$$\|\mathbf{y}\|^2_{\Sigma} = \mathbf{y}^T \Sigma^{-1} \mathbf{y}$$

- Nonlinear least squares problem
  - Levenberg Marquardt algorithm
  - "bundle adjust"

Minimization Algorithm

- Bundle adjust is computationally expensive
- Limit # poses, # points
- Use a sliding window of last N poses
  - Solutions are with respect to the first pose of this set
  - Effectively, mapping is done w.r.t. a moving local coordinate system



- Also, a good initial guess is needed
  - Pose guesses come from odometry
  - Point locations are initialized using triangulation

Mapping Program Control Flow

do

- Get next set of measurements (images and odometry)
- Predict pose of vehicle from odometry
- Track existing points
- Eliminate inactive points
- If predicted distance travelled is at least D meters
  - Define this pose as a “keyframe”
  - Try to acquire new points to track
  - Try to initialize 3D locations of 2D points using triangulation
  - Do bundle adjust algorithm over last 7 keyframes

while measurements are available

Thanks