DTU

Perception for Autonomous Systems 31392:

Visual SLAM

Simultaneous Localization & Mapping

Lecturer: Evangelos Boukas—PhD

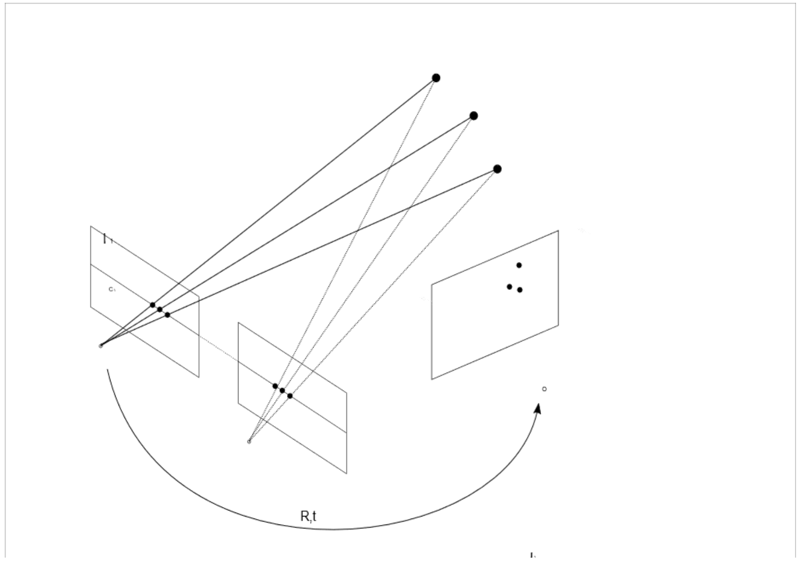
Visual Odometry is great

- Lets Sum Up What we did last time

- We performed 3D-to-2D

Algorithm 3. VO from 3-D-to-2-D Correspondences.

1) Do only once:
1.1) Capture two frames I_{k-2}, I_{k-1}
1.2) Extract and match features between them
1.3) Triangulate features from I_{k-2}, I_{k-1}
2) Do at each iteration:
2.1) Capture new frame I_k
2.2) Extract features and match with previous frame I_{k-1}
2.3) Compute camera pose (PnP) from 3-D-to-2-D matches
2.4) Triangulate all new feature matches between I_k and I_{k-1}
2.5) Iterate from 2.1).



3D-2D

Visual Odometry is great

- Lets Sum Up What we did last time

- We performed 3D-to-2D

Visual Odometry is great

- Lets Sum Up What we did last time
- We performed 3D-to-2D
- Any better approach? How could we

Visual Odometry is great

- Lets Sum Up What we did last time

- We performed 3D-to-2D

- Any better approach? How could we
  - How about some more features?

Visual Odometry is great

- Lets Sum Up What we did last time

- We performed 3D-to-2D

- Any better approach? How could we
  - How about some more features?
    - Let’s keep the features from multiple frames

Visual Odometry is great

- Lets Sum Up What we did last time

- We performed 3D-to-2D

- Any better approach? How could we
  - How about some more features?
    - Let’s keep the features from multiple frames
  - How could do we know that those are new features?

Visual Odometry is great

- Lets Sum Up What we did last time

- We performed 3D-to-2D

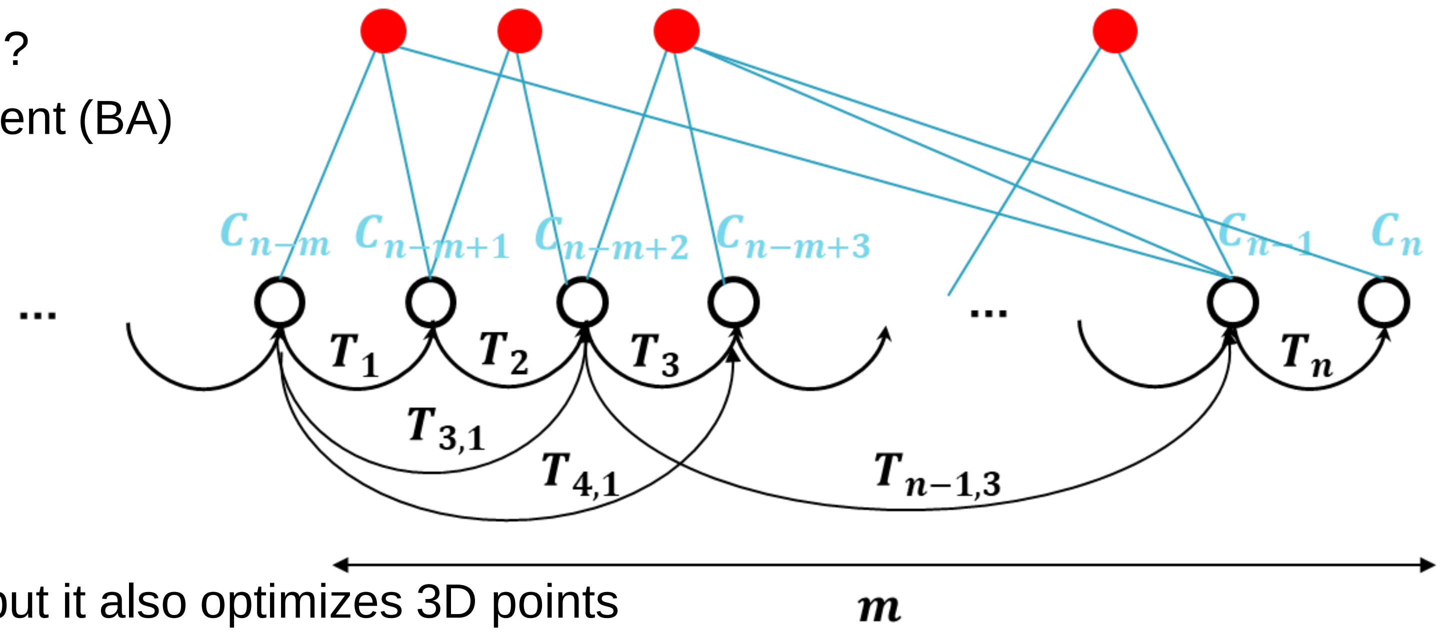
- Any better approach? How could we
  - How about some more features?
    - Let’s keep the features from multiple frames
  - How could do we know that those are new features?

Visual Odometry is great

- Lets Sum Up What we did last time
- We performed 3D-to-2D
-

Visual Odometry is great

- Anything more we mentioned?
  – Windowed Bundle Adjustment (BA)

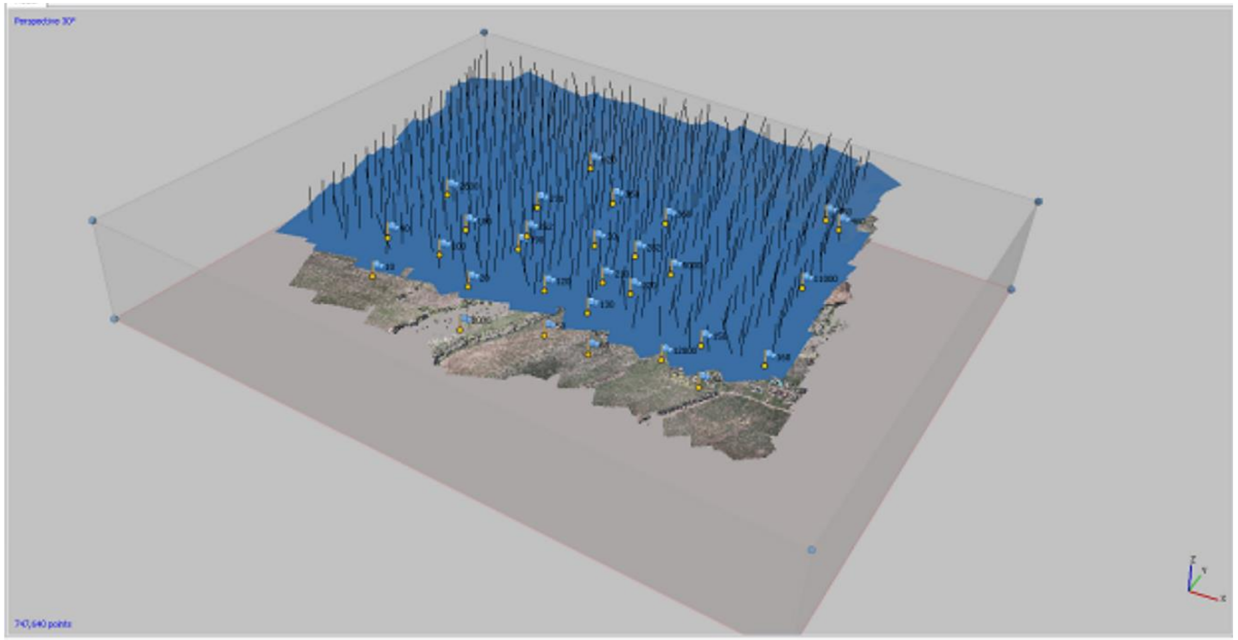


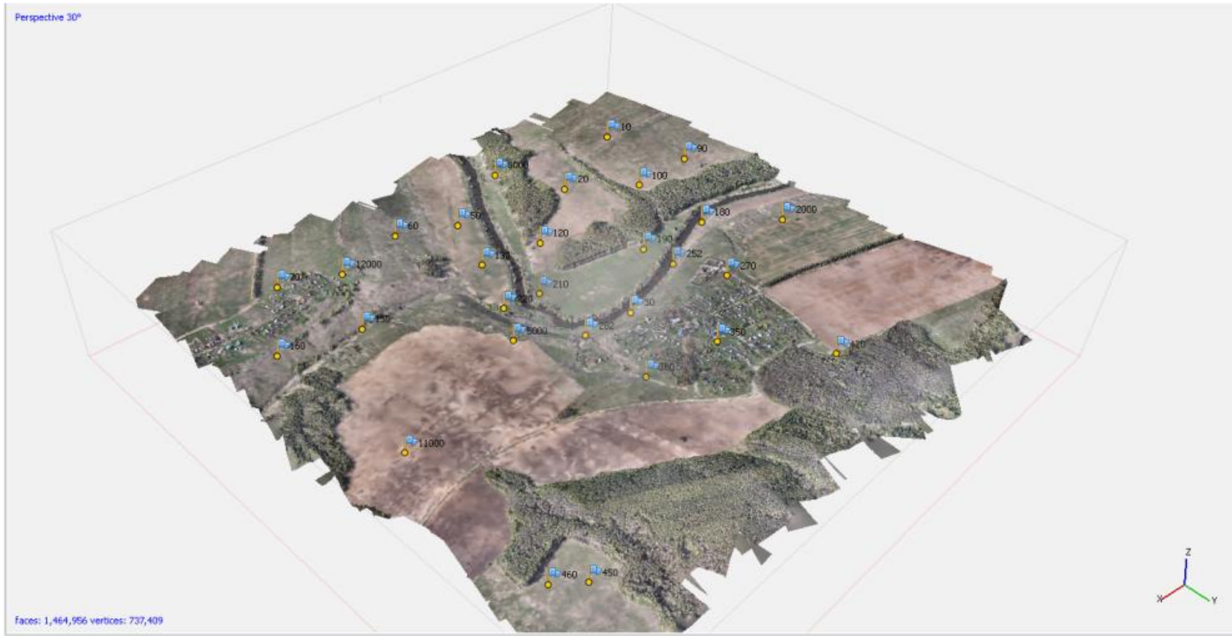
- Similar to pose-optimization but it also optimizes 3D points

- In order to not get stuck in local minima, the initialization should be close the minimum
- Levenberg-Marquadt can be used

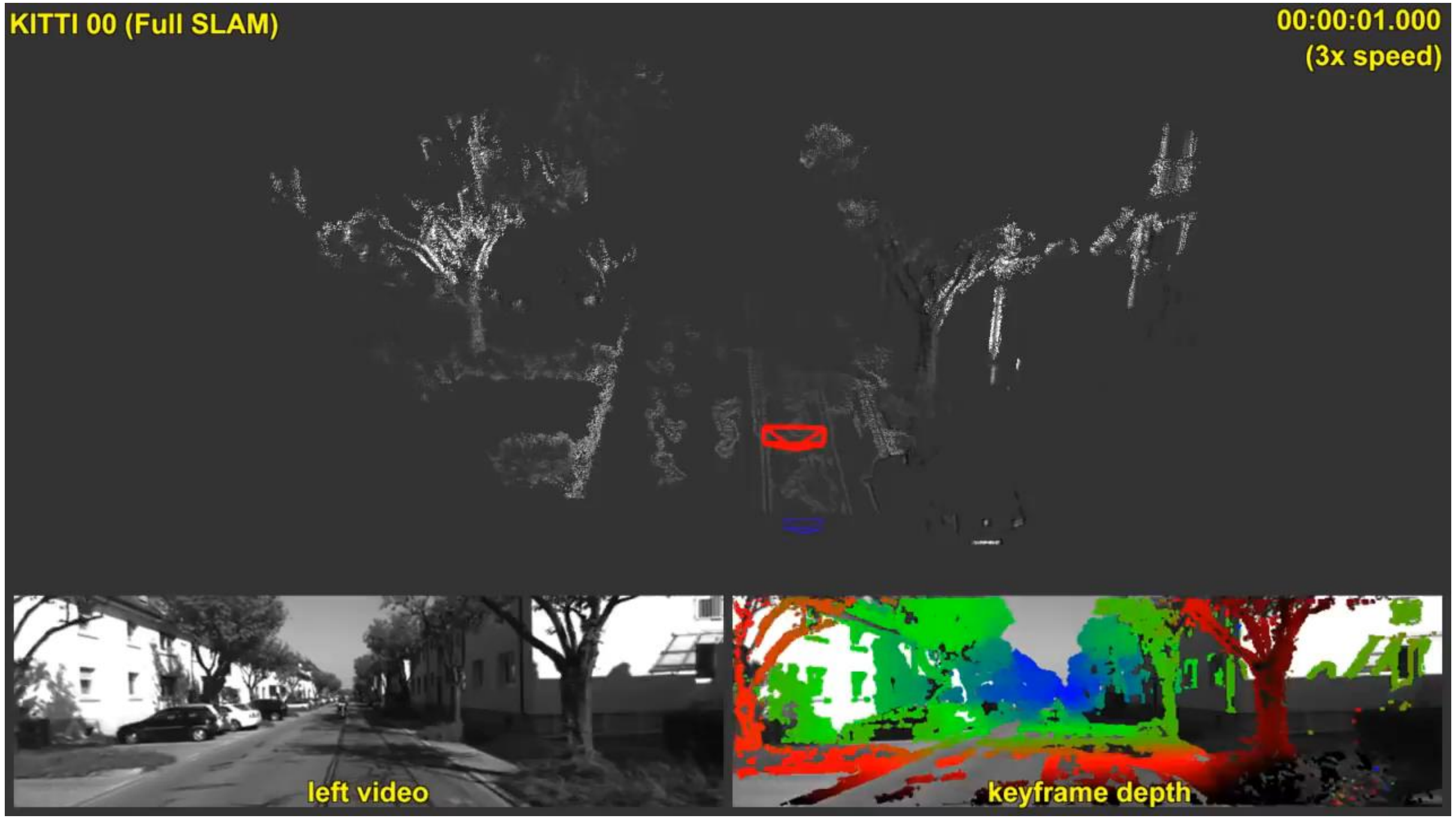
- Visual Odometry
- Structure from Motion (SFM)
- Bundle Adjustment
- Visual SLAM

- Visual Odometry
- Structure from Motion (SFM)
- Bundle Adjustment
- Visual SLAM

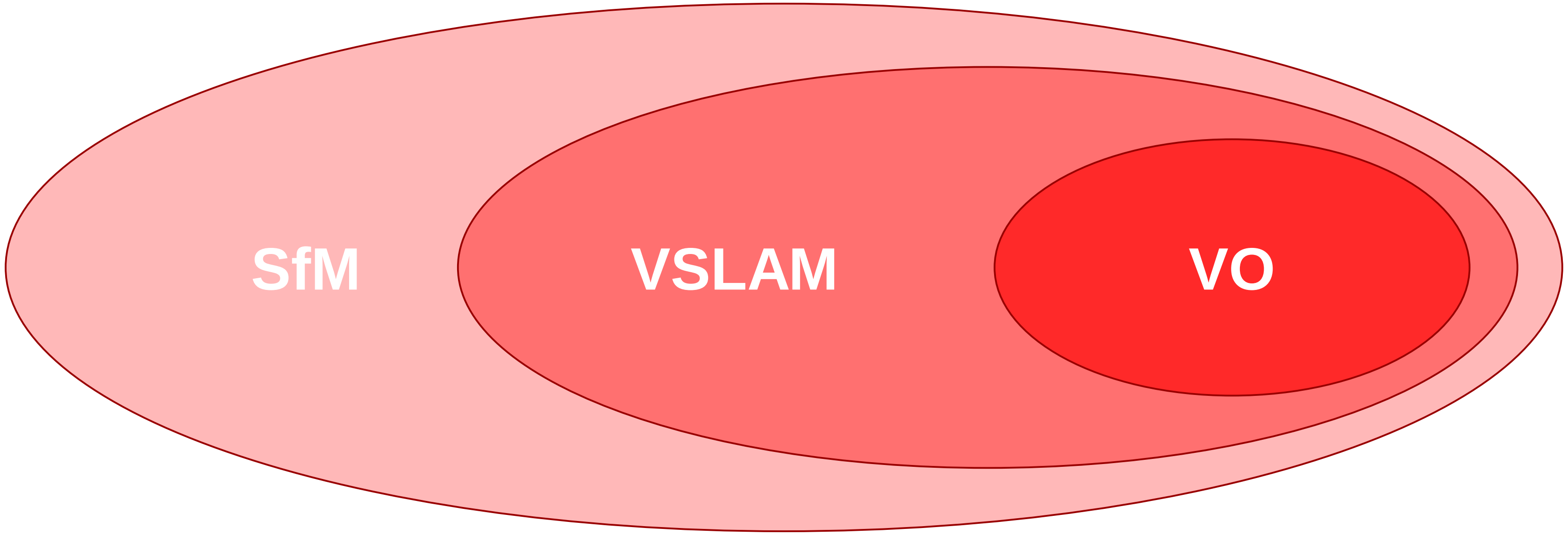




- Visual Odometry
- Structure from Motion (SFM)
- Bundle Adjustment
- Visual SLAM



- Visual Odometry
- Structure from Motion (SFM)
- Bundle Adjustment
- Visual SLAM

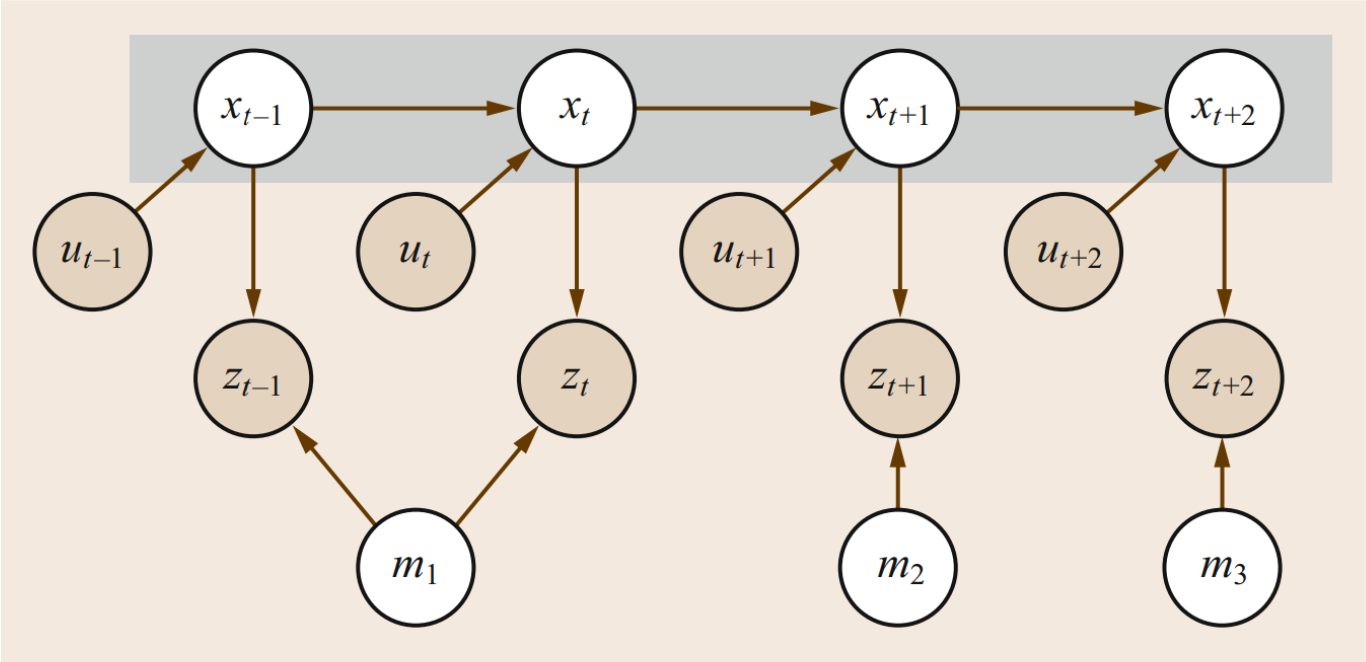


Outline

- Sum up Localization from last time
- Some terminology
- Pose-Landmark Graph Slam
- Example of Linear 1D SLAM
- Non-Linear Optimization approaches
- Bundle Adjustment
- Visual Slam System architecture
- ORBSLAM

Pose-Landmark Graph-Slam

- SLAM problem depicted as Bayes network graph
- At each **location $\mathbf{x}_t$**
- Observes a nearby feature in the **map $\mathbf{m} = \{\mathbf{m}_1; \mathbf{m}_2; \mathbf{m}_3\}$**
- Movement **$\mathbf{u}_t$**
- An arrow defines causal relationship

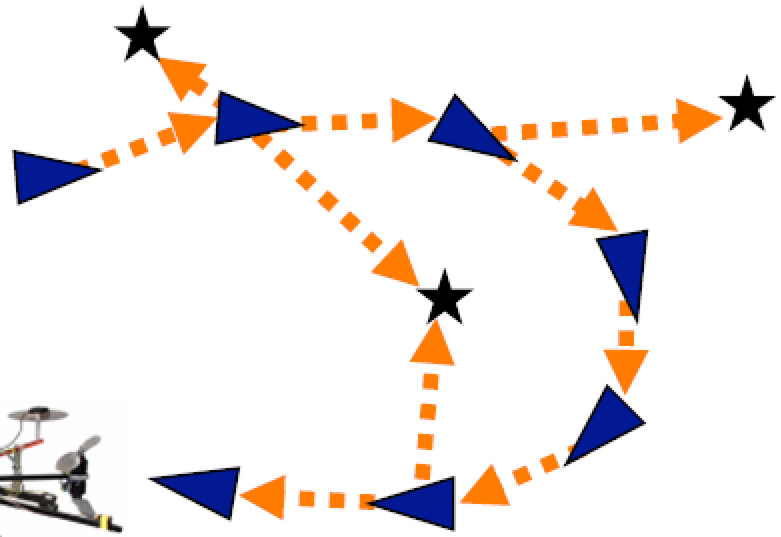


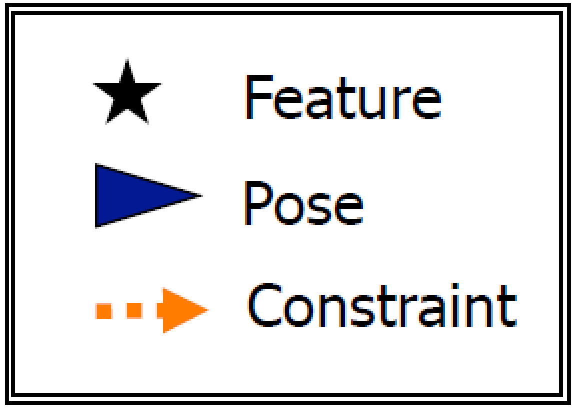
Graph-Based SLAM

Definition

- Use a graph to represent the problem
- Nodes represent:
  – poses or
  – locations
- Edges Represent:
  – Landmark observations
  – Odometry Measurements

- The minimization optimizes
  the landmark locations and robot poses

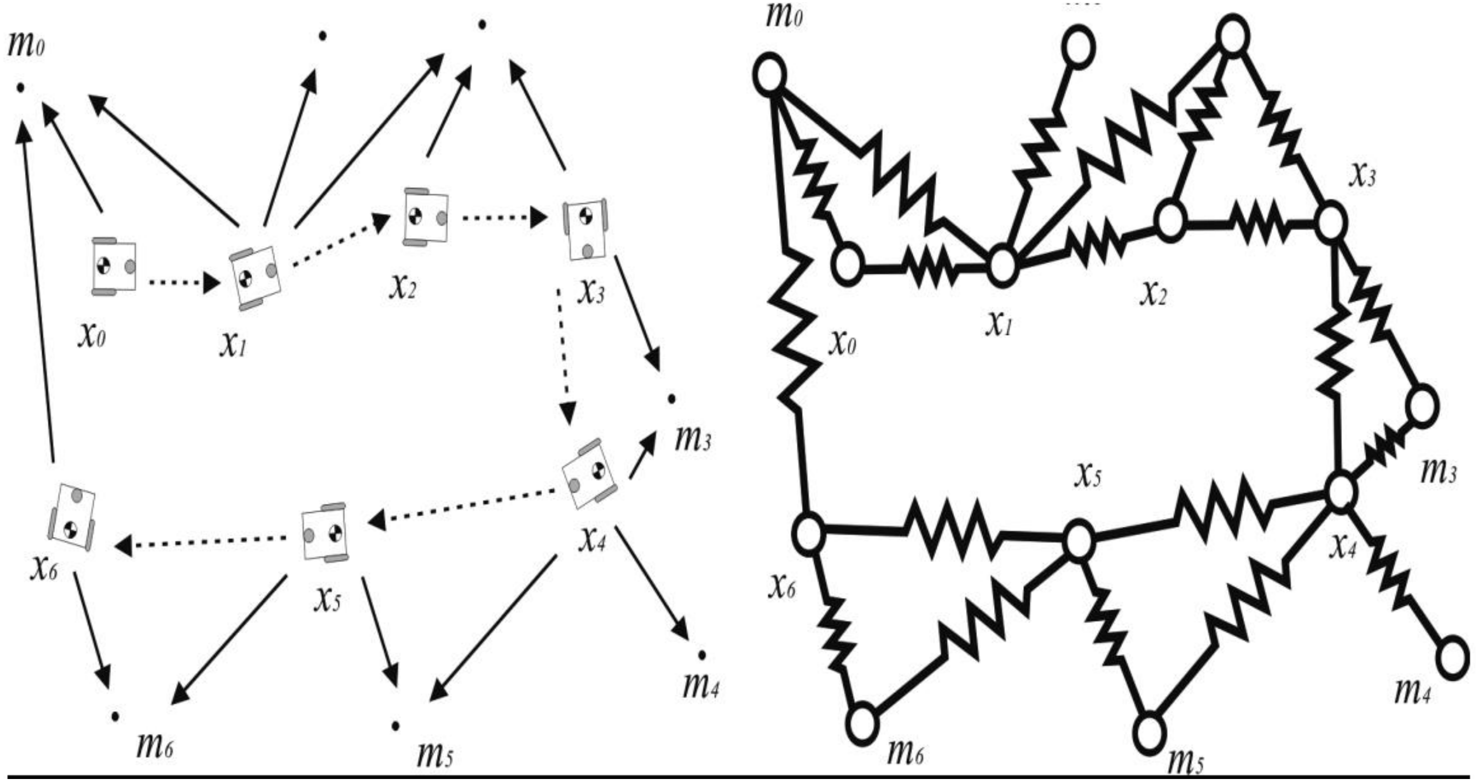




Graph-Based SLAM: Build the graph and find a node configuration that minimize the error introduced by the constraints

Graph-Based SLAM

- Observing previously seen areas generates constraints between non-successive poses
- Treat constraints (motion and measurement) as “soft” elastic springs
- Want to minimize the total energy in the springs



Graph-Based SLAM

- Observing previously seen areas generates constraints between non-successive poses
- Treat constraints (motion and measurement) as “soft” elastic springs
- Want to minimize the total energy in the springs

We can define the error as follows

- Expected observation (2D sensor): $\underbrace{\hat{\mathbf{z}}_{ij}(\mathbf{x}_i}_{\text{robot}}, \underbrace{\mathbf{x}_j}_{\text{landmark}}) = \mathbf{R}_i^T(\mathbf{x}_j - \underbrace{\mathbf{t}_i}_{\text{robot translation}})$

  – With the error:
$$\begin{aligned} \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) &= \hat{\mathbf{z}}_{ij} - \mathbf{z}_{ij} \\ &= \mathbf{R}_i^T(\mathbf{x}_j - \mathbf{t}_i) - \mathbf{z}_{ij} \end{aligned}$$

1D Linear SLAM

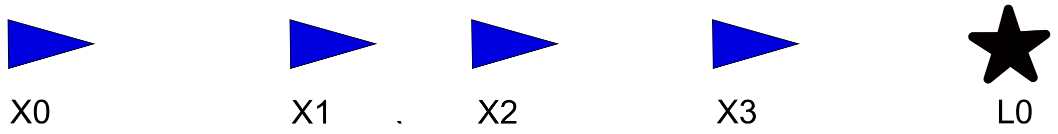
- In the linear case we can solve as follows:

  – First construct all constrains
    - Absolute Constrain:

      X(0) = Q - starting position
    - Movement Constrains:

      X(t) = X(t-1) + Dx(t)
    - Measurement constrains:

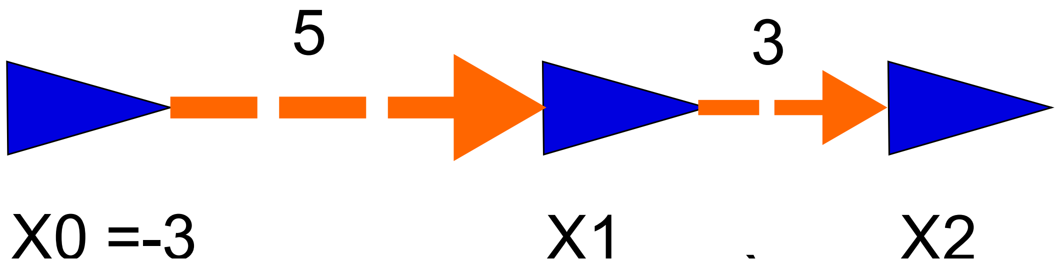
      L(k) = X(t) + N
    - *Then, solve linear equations*



1D Linear SLAM – case 1

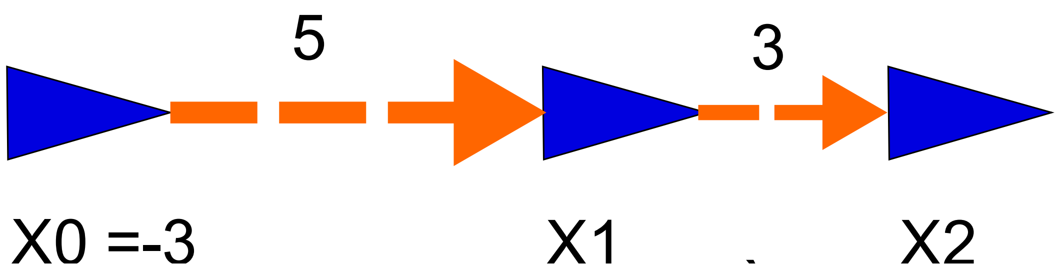
- Case 1 - Exact solution exists:

$$\left[\qquad\qquad\right]\cdot\left[\quad\right]=\left[\qquad\right]$$



1D Linear SLAM – case 1

- Case 1 - Exact solution exists:

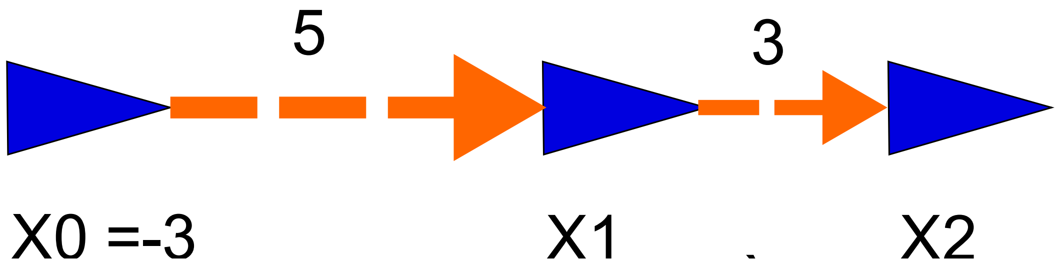


$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \cdot \begin{bmatrix} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \end{bmatrix} \qquad A^*X = B$$

1D Linear SLAM – case 1

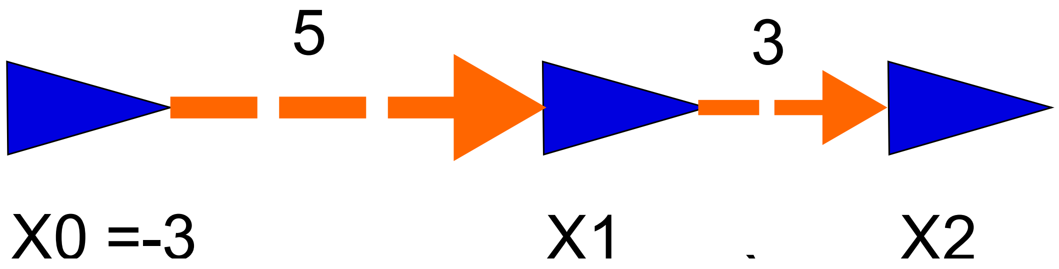
- Case 1 - Exact solution exists:



$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \end{bmatrix} \qquad A^*X = B \qquad X = A^{-1*}B \qquad A^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

1D Linear SLAM – case 1

- Case 1 - Exact solution exists:

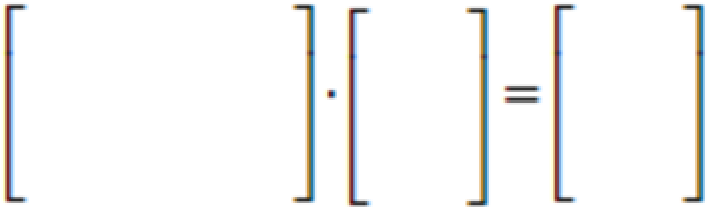


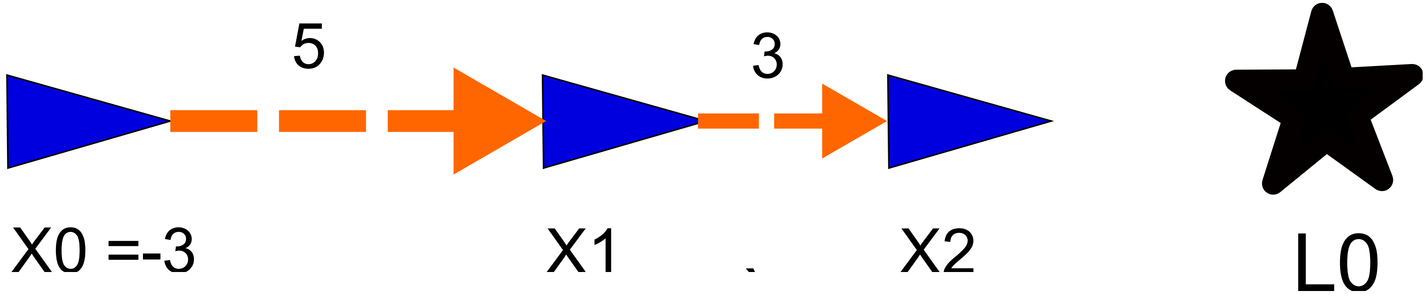
$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \end{bmatrix} \quad A^*X = B \quad X = A^{-1*}B \quad A^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\boxed{x = [\,-3 \;\; 2 \;\; 5\,]}$$

1D Linear SLAM – case 2

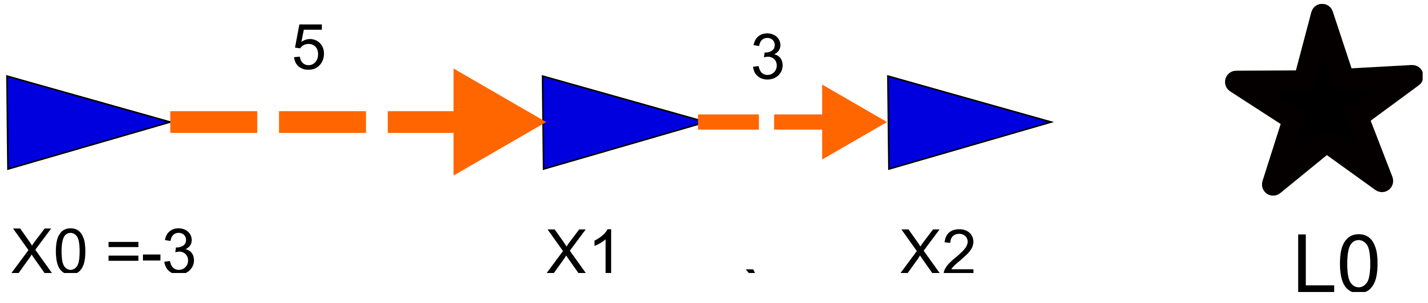
- Case 2 – Overdefined problem:
  - X_0 sees L_0 at distance 10
  - X_1 sees L_0 at distance 5
  - X_2 sees L_0 at distance 2





1D Linear SLAM – case 2

- Case 2 – Overdefined problem:
  - X_0 sees L_0 at distance 10
  - X_1 sees L_0 at distance 5
  - X_2 sees L_0 at distance 2

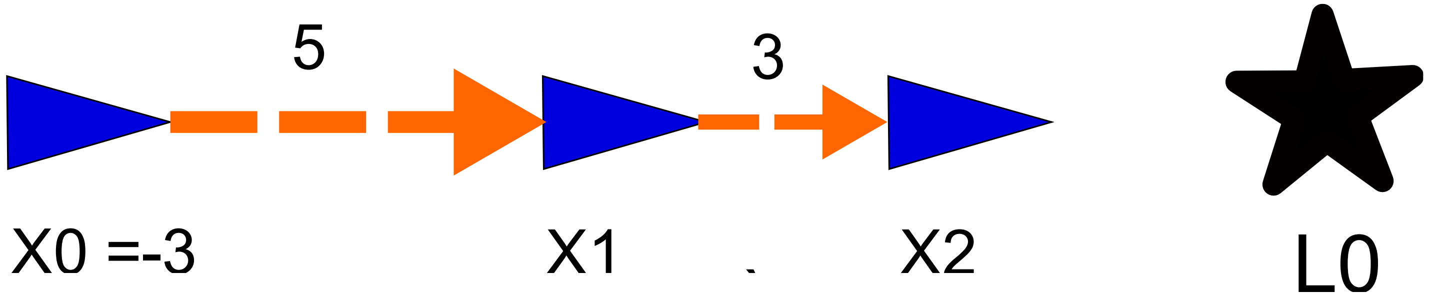


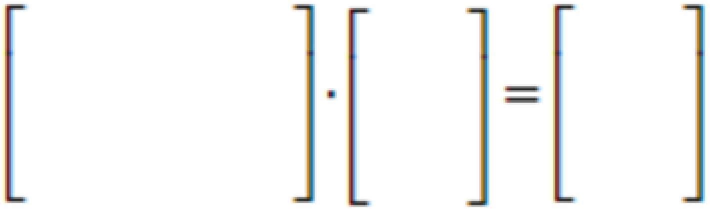
$$A*X=B \qquad X=A^{-1}*B \qquad A^{-1}=[?]$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L_0 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \\ -10 \\ -5 \\ -2 \end{bmatrix}$$

1D Linear SLAM – case 2

- Case 2 – Overdefined problem:
  - X_0 sees L_0 at distance 10
  - X_1 sees L_0 at distance 5
  - X_2 sees L_0 at distance 2





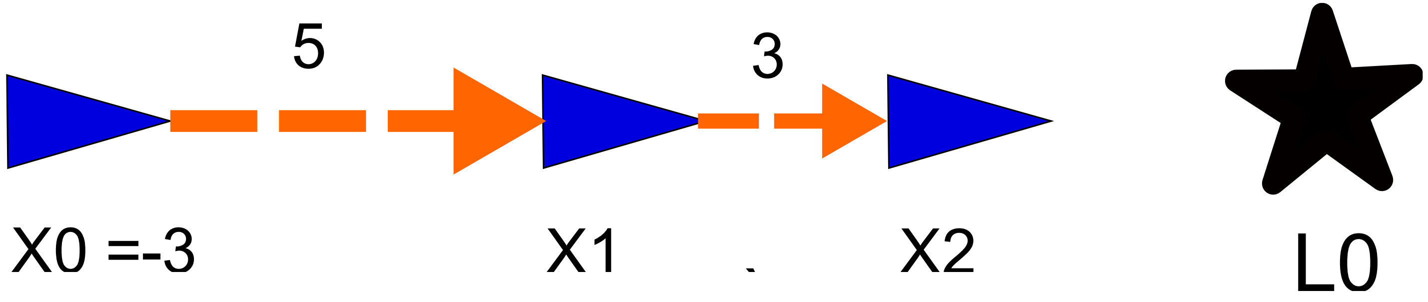
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L_0 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \\ -10 \\ -5 \\ -2 \end{bmatrix}$$

$$A*X=B \qquad X=A^{-1}*B \qquad A^{-1}=[?]$$

$$X = (A^T * A)^{-1} * A^T * B$$

1D Linear SLAM – case 2

- Case 2 – Overdefined problem:
  - X_0 sees L_0 at distance 10
  - X_1 sees L_0 at distance 5
  - X_2 sees L_0 at distance 2



$$\left[\qquad\qquad\right]\cdot\left[\quad\right]=\left[\quad\right]$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L_0 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \\ -10 \\ -5 \\ -2 \end{bmatrix}$$

$A*X=B \qquad X=A^{-1}*B \qquad A^{-1}=[?]$

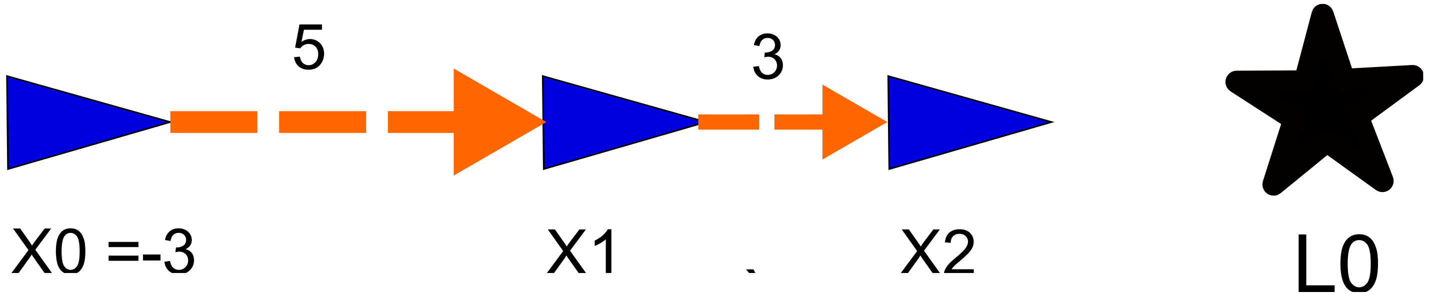
$X = (A^T * A)^{-1} * A^T * B$

$$\boxed{x = \begin{bmatrix} -3 & 2 & 5 & 7 \end{bmatrix}}$$

We infer a consistent landmark position

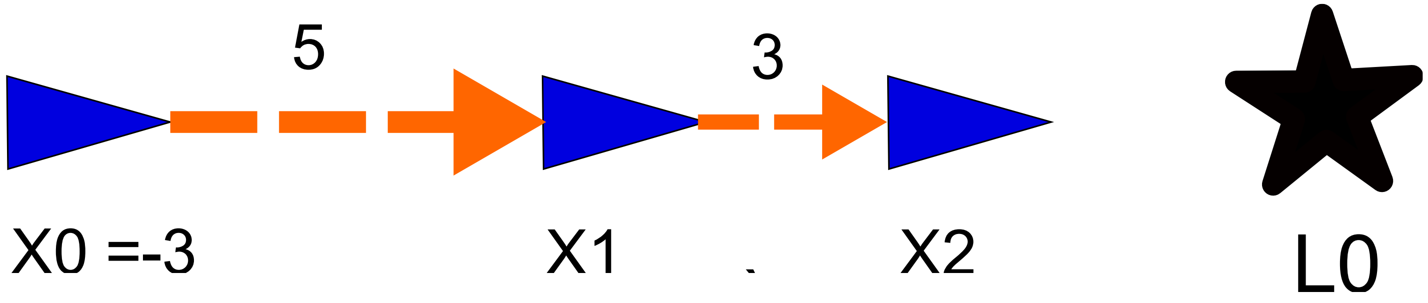
1D Linear SLAM – case 3

- Case 3 – Inconsistent Measurements :
  - – X_0 sees L_0 at distance 10
  - – X_1 sees L_0 at distance 5
  - – X_2 sees L_0 at distance **1 (Wrong)**



1D Linear SLAM – case 3

- Case 3 – Inconsistent Measurements :
  - X_0 sees L_0 at distance 10
  - X_1 sees L_0 at distance 5
  - X_2 sees L_0 at distance **1 (Wrong)**

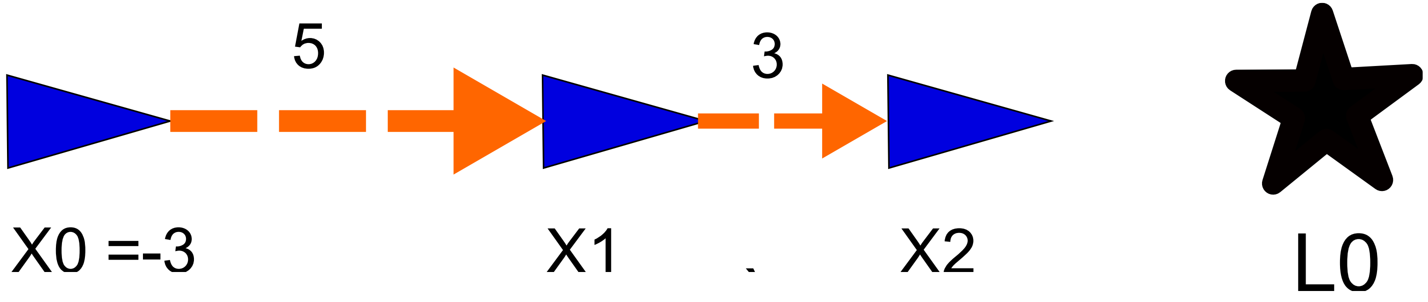


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L_0 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \\ -10 \\ -5 \\ -1 \end{bmatrix}$$

$$X = (A^T * A)^{-1} * A^T * B$$

1D Linear SLAM – case 3

- Case 3 – Inconsistent Measurements :
  - X_0 sees L_0 at distance 10
  - X_1 sees L_0 at distance 5
  - X_2 sees L_0 at distance **1 (Wrong)**



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L_0 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \\ -10 \\ -5 \\ -1 \end{bmatrix}$$

$$X = (A^T * A)^{-1} * A^T * B$$

$$\boxed{x = \begin{bmatrix} -3 & 2.125 & 5.5 & 6.875 \end{bmatrix}}$$

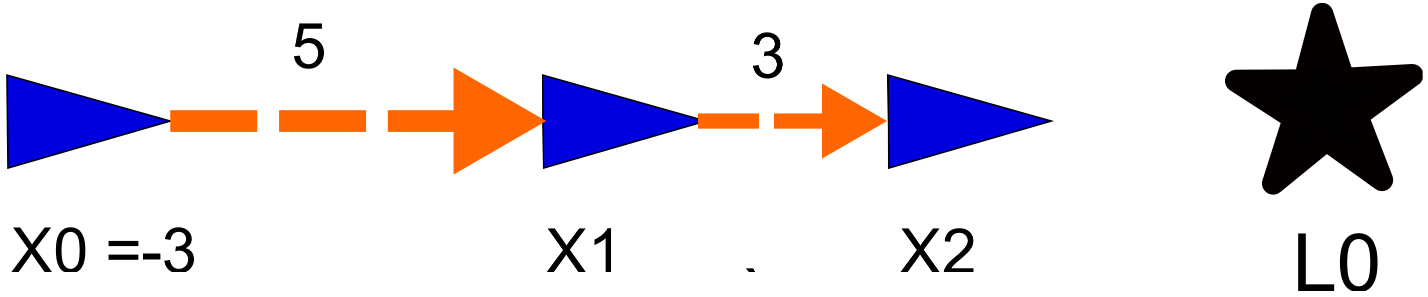
We handled inaccurate measurements

1D Linear SLAM – case 4

Case 4 – Inconsistent Measurements
with Confidence Matrix:

- Linear Least Squares allows us to include a weighting of each linear constraint.
- We can include weights in the computation
- We weight each constraint by a diagonal matrix where the weights are 1/variance for each constraint.
- Let's say X_2 **variance is 0.2**

$$X = (A^T * W * A)^{-1} * A^T * W * B$$

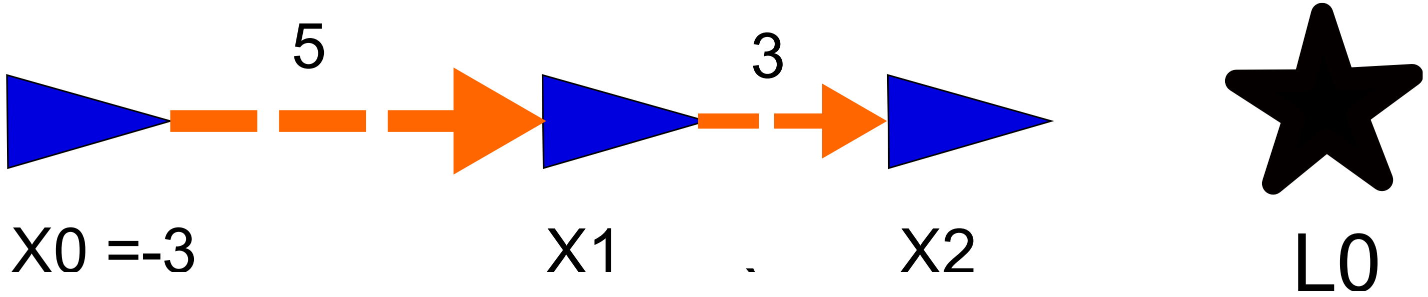


1D Linear SLAM – case 4

Case 4 – Inconsistent Measurements with Confidence Matrix:

- Linear Least Squares allows us to include a weighting of each linear constraint.
- We can include weights in the computation
- We weight each constraint by a diagonal matrix where the weights are 1/variance for each constraint.
- Let's say X_2 **variance is 0.2**

$$X = (A^T * W * A)^{-1} * A^T * W * B$$



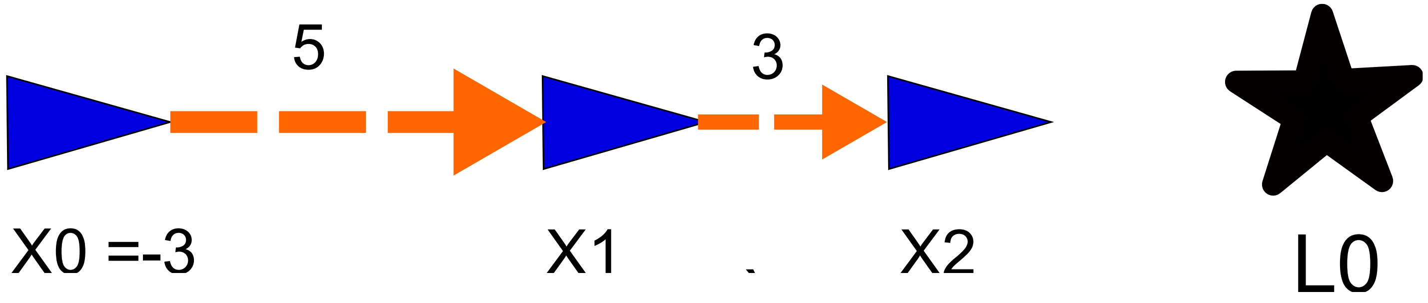
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L_0 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \\ -10 \\ -5 \\ -1 \end{bmatrix}$$

1D Linear SLAM – case 4

Case 4 – Inconsistent Measurements with Confidence Matrix:

- Linear Least Squares allows us to include a weighting of each linear constraint.
- We can include weights in the computation
- We weight each constraint by a diagonal matrix where the weights are 1/variance for each constraint.
- Let's say X_2 **variance is 0.2**

$$X = (A^T * W * A)^{-1} * A^T * W * B$$



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L_0 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \\ -10 \\ -5 \\ -1 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

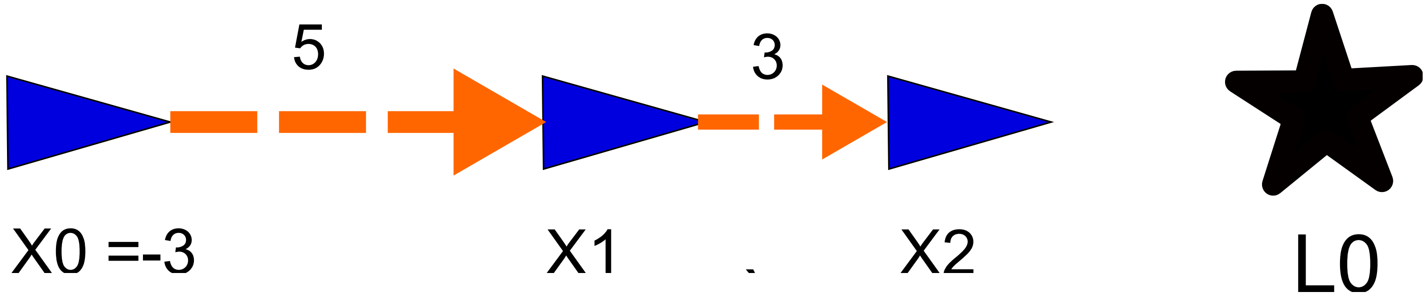
1D Linear SLAM – case 4

Case 4 – Inconsistent Measurements with Confidence Matrix:

- Linear Least Squares allows us to include a weighting of each linear constraint.
- We can include weights in the computation
- We weight each constraint by a diagonal matrix where the weights are 1/variance for each constraint.
- Let's say X_2 **variance is 0.2**

$$X = (A^T * W * A)^{-1} * A^T * W * B$$

$$\boxed{x = \begin{bmatrix} -3 & 2.18 & 5.71 & 6.82 \end{bmatrix}}$$



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} . \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L_0 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 3 \\ -10 \\ -5 \\ -1 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

Why did the estimation just become worse??

What about non-Linear Least Squares?

- Large number of geometric problems in computer vision are non-linear least-squares problems.

$$\mathbf{X} = \mathbf{h}(\theta)$$

where $\mathbf{h} : \mathbf{R}^n \rightarrow R^m$.

- $\mathbf{X}$ is the measurement vector, θ is the parameter vector.
- Write $\mathbf{f}(\theta) = \mathbf{h}(\theta) - \mathbf{X}$.
- We desire to minimize

$$\|\mathbf{f}(\theta)\|^2$$

over all choices of parameter θ.

Gauss Newton Solution

1. Start from an initial value θ_0.
2. At step i assume a linear approximation for the function at θ_i

$$\mathbf{f}(\theta_i + \Delta) = \mathbf{f}(\theta_i) + \mathbf{f}_\theta \boldsymbol{\Delta} \text{ where } \mathbf{f}_\theta = \partial \mathbf{f} / \partial \theta = \mathbf{J} \ .$$

3. Solve

$$\mathbf{f}(\theta_i + \Delta) = \mathbf{f}(\theta_i) + \mathbf{J}\boldsymbol{\Delta} = 0$$

   or

$$\mathbf{J}\boldsymbol{\Delta} = -\mathbf{f}(\theta_i)$$

4. This is a linear least-squares problem (solve for $\boldsymbol{\Delta}$):

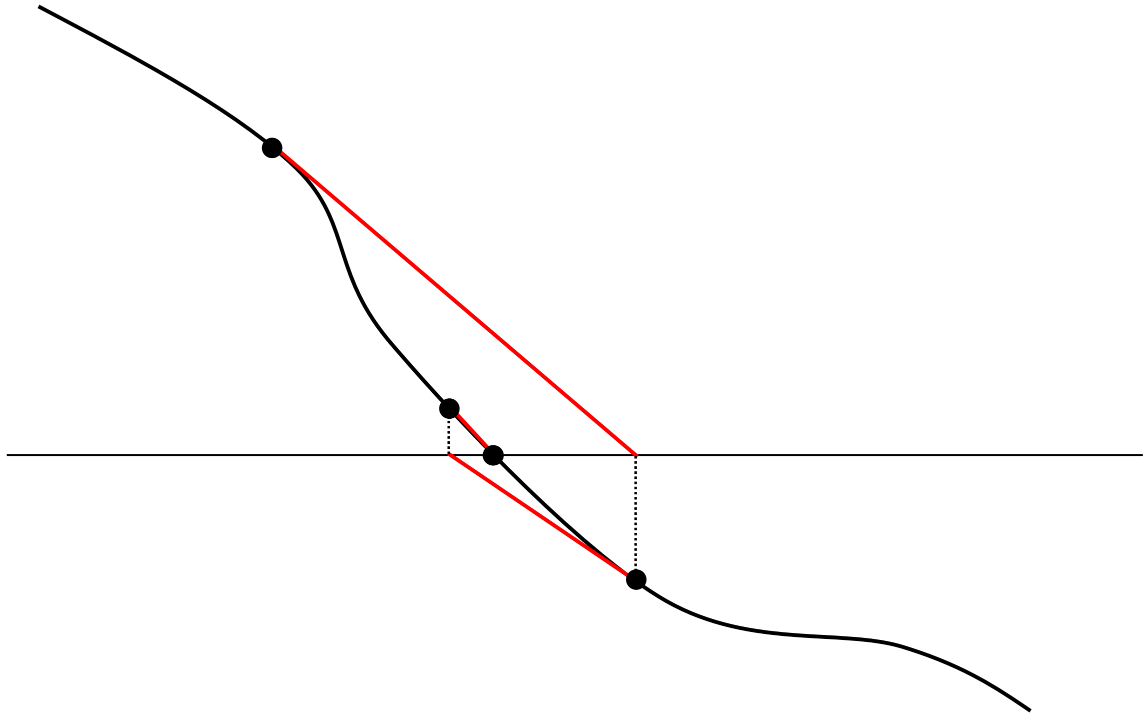
$$\mathbf{J}^\top \mathbf{J}\boldsymbol{\Delta} = \mathbf{J}^\top \mathbf{f}(\theta_i)$$

5. Then set $\theta_{i+1} = \theta_i + \boldsymbol{\Delta}$.

Gauss-Newton update equation

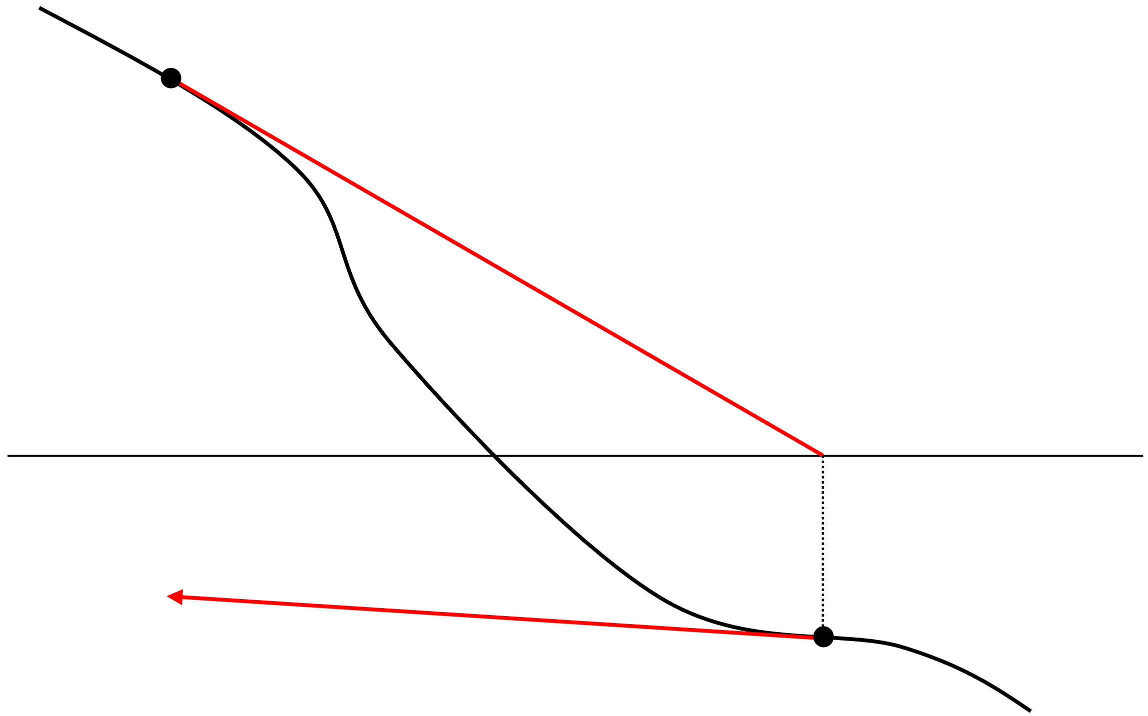
$$\mathbf{J}^\top \mathbf{J}\boldsymbol{\Delta} = -\mathbf{J}^\top \mathbf{f}$$

Gauss Newton



1D Gauss-Newton (Newton) iteration.

Gauss Newton



1D Gauss-Newton (Newton) iteration (failure)

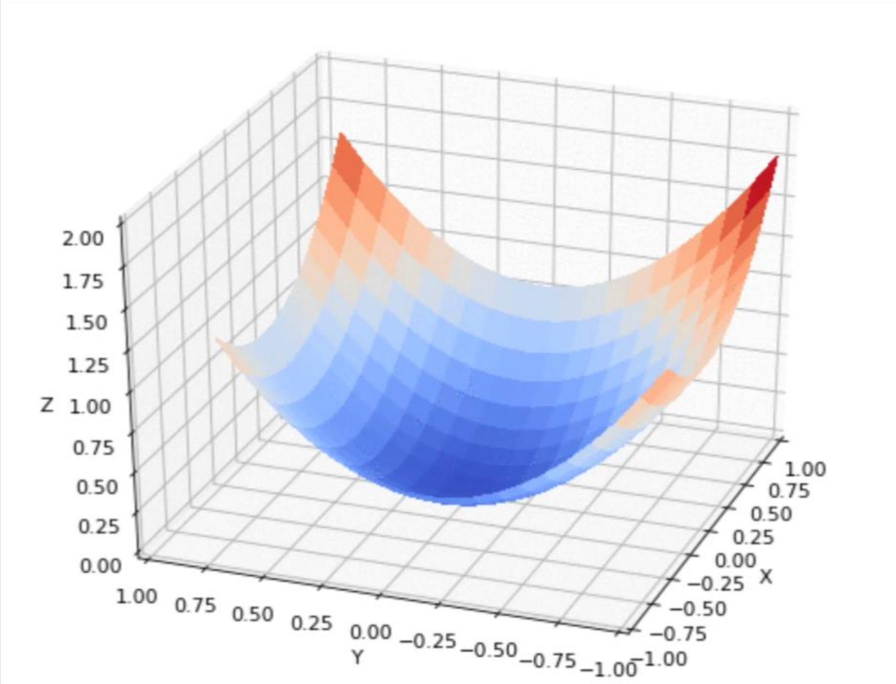
Gradient Descent

Search direction is the direction of fastest descent of the function g.

Gradient descent update equation

$$\lambda\Delta = -g_\theta = -\mathbf{J}^\top\mathbf{f}$$

Requires a 1D line search in λ to find the optimum direction.

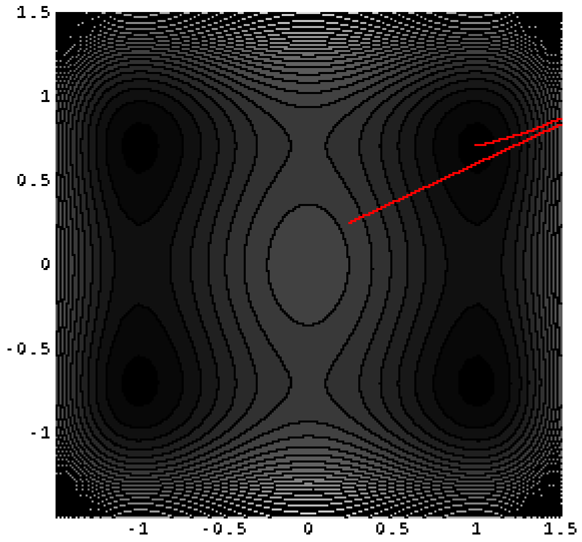


Levenberg-Marquadt

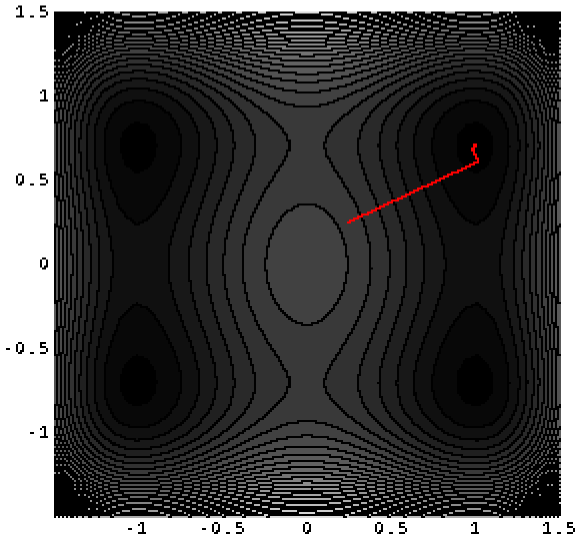
- Mixture of Gauss-Newton and Gradient descent.
- Acts like Gauss-Newton when close to the minimum (quadratic region)
- Gradient descent when improvement is difficult.
- Depends on a parameter λ which
  1. Controls the mixture of Gauss-Newton and Gradient Descent
  2. Controls the step-length.

What about non-Linear Least Squares?

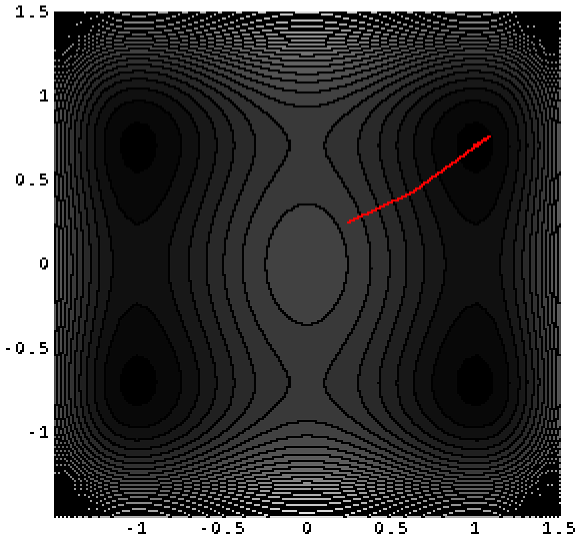
- Lets See some examples 1:



Gauss-Newton



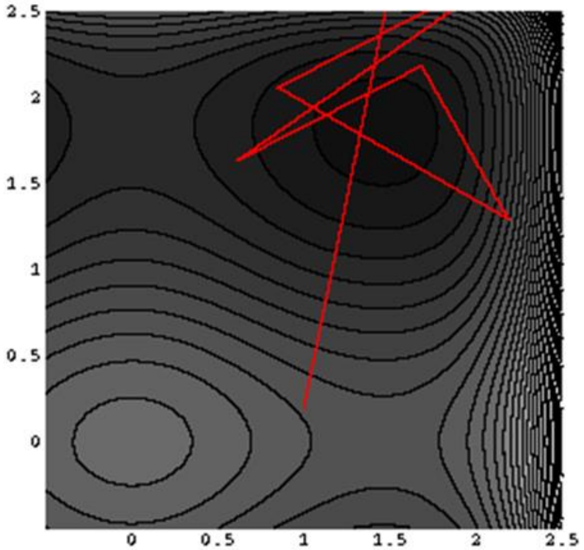
Gradient descent



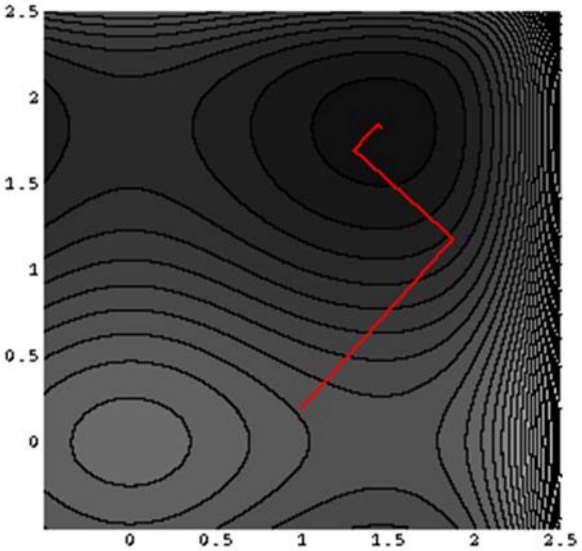
Levenberg

What about non-Linear Least Squares?

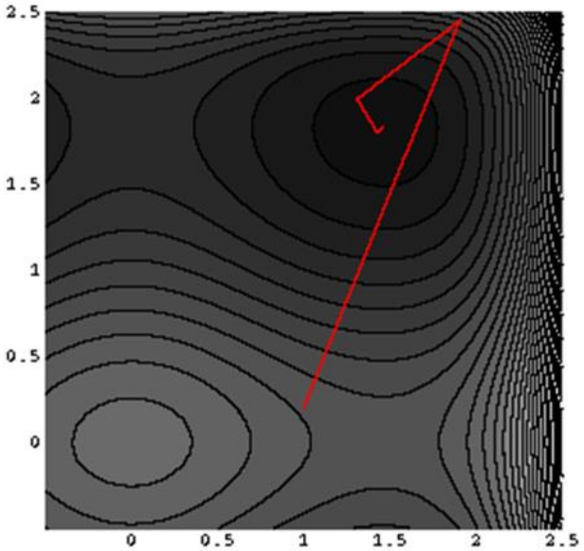
- Lets See some examples 2:



Gauss-Newton



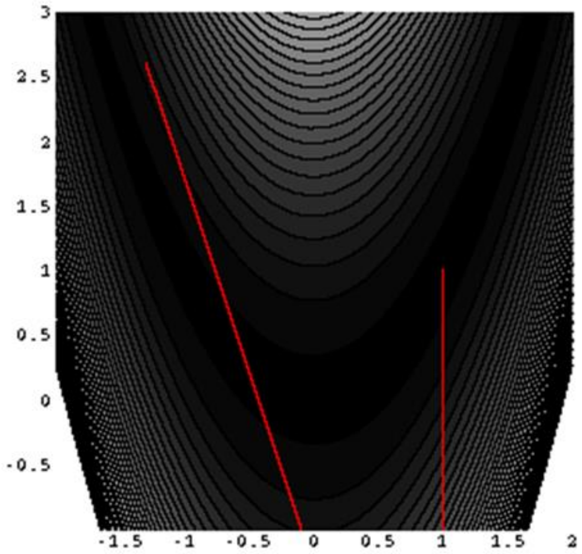
Gradient descent



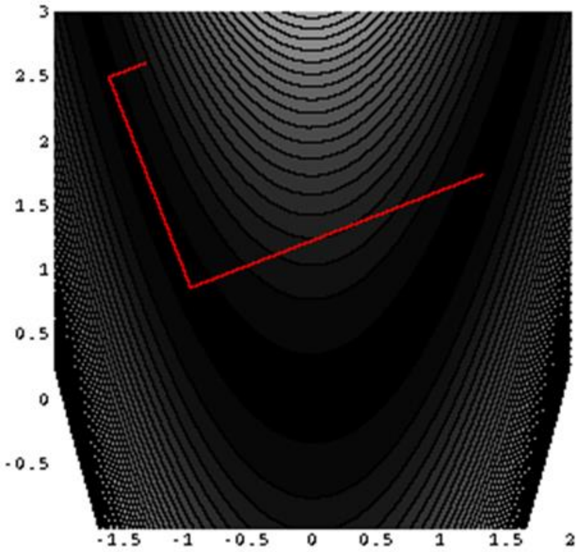
Levenberg

What about non-Linear Least Squares?

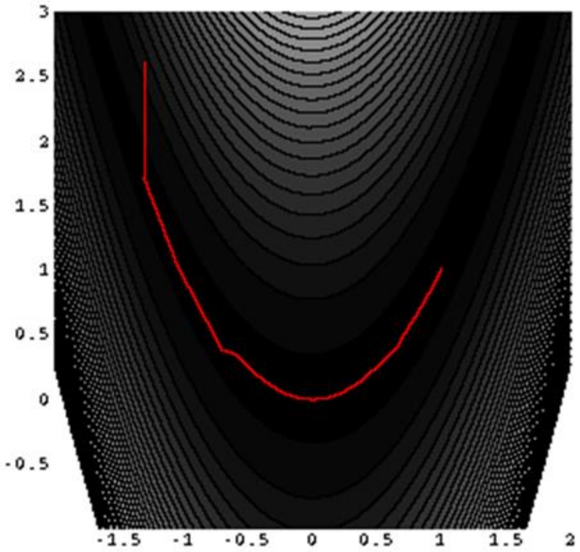
- Lets See some examples 3:



Gauss-Newton



Gradient descent



Levenberg

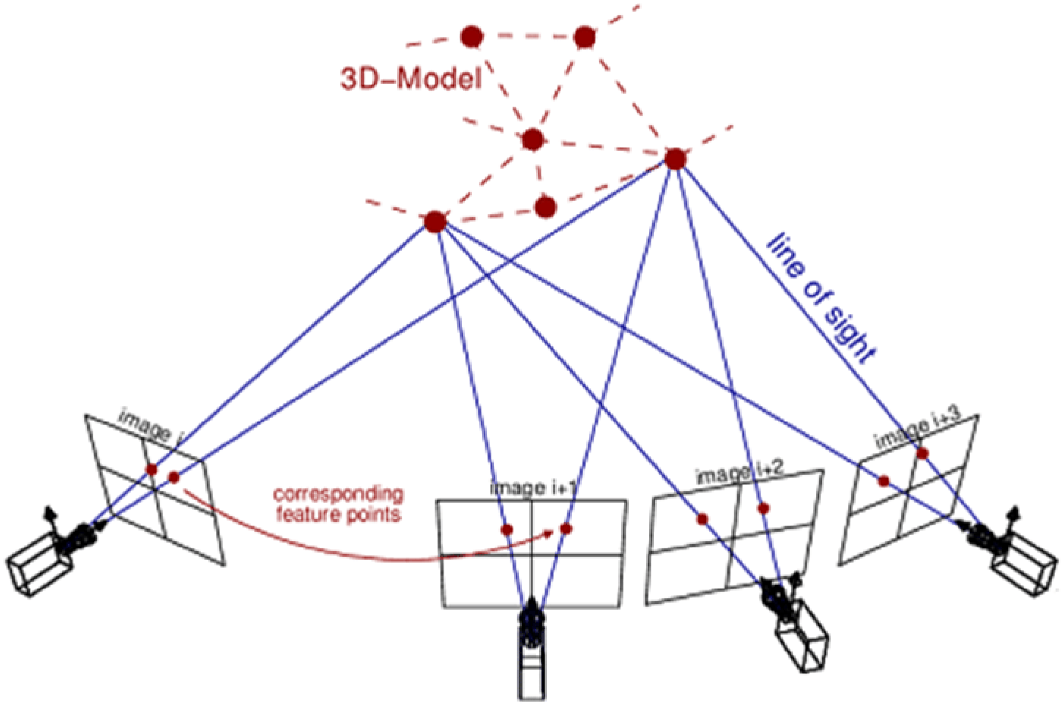
- It is obvious that Levenberg Marquadt displays robustness

Bundle Adjustment

- Bundle Adjustment is the employment of nonlinear optimization in the problem of the minimization of the re-projection error, by finding the optimal Poses (extrinsics) of the cameras and the locations of the 3D points.

$$\arg\min_{\mathbf{w},\boldsymbol{\theta}} \sum_{f=1}^{F} \sum_{n=1}^{N} ||\mathbf{x}_n^f - \pi(\mathbf{w}_n; \boldsymbol{\theta}^f)||_2^2$$

$\mathbf{x}$	2D projection	$\mathbf{w}$	3D point
$\boldsymbol{\theta}$	extrinsics	N	no. of points
π	projection function		
F	no. of frames		



Bundle Adjustment – Linearization

$$\pi(\mathbf{w}_n + \Delta\mathbf{w}_n; \boldsymbol{\theta}_f \circ \Delta\boldsymbol{\theta}_f) \approx \pi(\mathbf{w}_n; \boldsymbol{\theta}_f) + \mathbf{J}_n^f \begin{bmatrix} \Delta\boldsymbol{\theta}_f \\ \Delta\mathbf{w}_n \end{bmatrix}$$

$$\Downarrow$$

$$\arg\min_{\Delta\boldsymbol{\theta}, \Delta\mathbf{w}} \sum_{f=1}^{F} \sum_{n=1}^{N} \rho_n^f || \mathbf{x}_n^f - \pi(\mathbf{w}_n; \boldsymbol{\theta}_f) - \mathbf{J}_n^f \begin{bmatrix} \Delta\boldsymbol{\theta}_f \\ \Delta\mathbf{w}_n \end{bmatrix} ||_2^2$$

$\mathbf{x}$ 2D projection $\mathbf{w} \leftarrow$ 3D point

$\boldsymbol{\theta}$ extrinsics N no. of points

π projection function

F no. of frames $\rho \rightarrow$ visibility $\in [0, 1]$

Bundle Adjustment – Linearization

- The Linearization of the minimization happens by calculating the Jacobian of the projection matrix
- Assuming the following projection matrix:

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} f_x & s_k & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

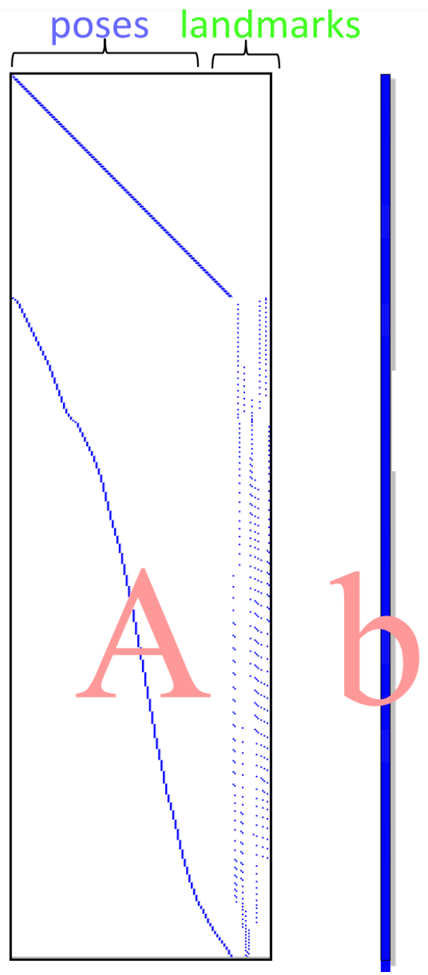
- We first define the orientation as the rotation matrix associated with the axis angle w_x, w_y, w_z using the Rodriques equation

- The Jacobian **FUNCTION** can be calculated as:

$$\mathrm{J} = \begin{bmatrix} \frac{\partial u}{\partial w_x} & \frac{\partial u}{\partial w_y} & \frac{\partial u}{\partial w_z} & \frac{\partial u}{\partial f} & \frac{\partial u}{\partial u_0} & \frac{\partial u}{\partial v_0} & \frac{\partial u}{\partial X} & \frac{\partial u}{\partial Y} & \frac{\partial u}{\partial Z} \\ \frac{\partial v}{\partial w_x} & \frac{\partial v}{\partial w_y} & \frac{\partial v}{\partial w_z} & \frac{\partial v}{\partial f} & \frac{\partial v}{\partial u_0} & \frac{\partial v}{\partial v_0} & \frac{\partial v}{\partial X} & \frac{\partial v}{\partial Y} & \frac{\partial v}{\partial Z} \end{bmatrix}$$

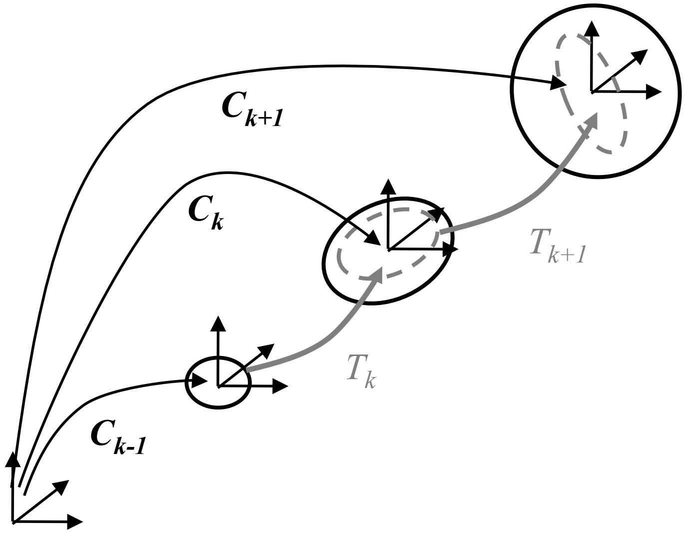
Bundle Adjustment – Comments

- Bundle adjustment (and graph optimization) is the backbone of all SLAM algorithms
- Keep in mind that:
  - We need to provide the Jacobian of the projection
  - We usually provide a covariance matrix (See the linear case for uncertainty)
  - It is solved using Levenberg-Marquadt
  - There are a lot of computational issues which are overcome exploiting the sparsity of the function AX=b (see least squares)
    Look at the following A and b Matrices
    This solution is called **Sparse Bundle Adjustment!**



Bundle Adjustment – Covariance

- The uncertainty of the camera pose C_k is a combination of the uncertainty at C_{k-1} (black-solid ellipse) and the uncertainty of the transformation T_k (gray dashed ellipse)



- $C_k = f(C_{k-1}, T_k)$
- The combined covariance Σ_k is

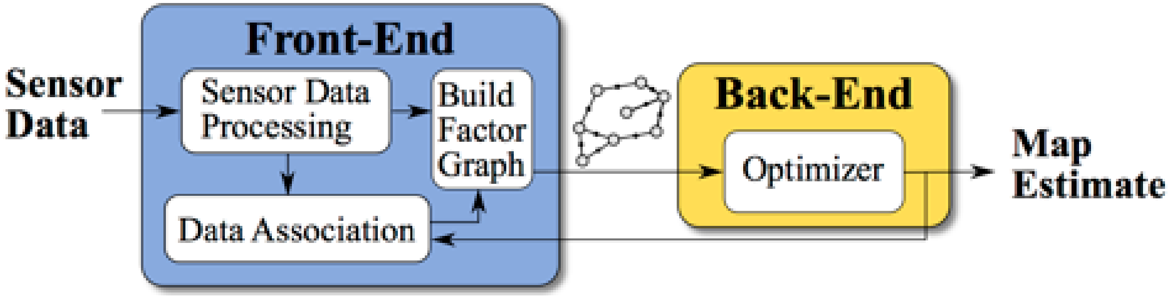
$$\begin{aligned}\Sigma_k &= J \begin{bmatrix} \Sigma_{k-1} & 0 \\ 0 & \Sigma_{k,k-1} \end{bmatrix} J^\top \\ &= J_{\vec{C}_{k-1}} \Sigma_{k-1} J_{\vec{C}_{k-1}}{}^\top + J_{\vec{T}_{k,k-1}} \Sigma_{k,k-1} J_{\vec{T}_{k,k-1}}{}^\top\end{aligned}$$

- The camera-pose uncertainty is always increasing when concatenating transformations. Thus, it is important to keep the uncertainties of the individual transformations small

Recent Visual Slam Solutions - Intro

- That was too much info, let’s see now some recent solutions to the Slam Problem:

- Most recent visual SLAM methods are split in two parts:
- The **Frontend**: where the raw data are converted into pose graphs and Loop constrains and the **Backend** where, given a graph with constrains, the new pose of the robot is calculated as well as the surrounding map points.



Recent Visual Slam Solutions – Common Architecture

- Front End
  - Data Association
    - Frame to Frame
    - Multi-frame
    - Loop Closure Detection
  - Geometric Initialization
    - Pose Estimation
    - Landmark Triangulation
  - System Formation
    - Observation Matrix
    - Covariance Matrix
    - Graph Generation and Update

- Back End
  - Filter-Based State Estimation
    - Extended Kalman Filter
    - Particle Filters
  - Least squares optimization
    - Bundle Adjustment
    - Graph Optimization
      - Key Frame

Recent Visual Slam Solutions – recent advances

Towards Realtime operation?:

- The computational cost of bundle adjustment has lead to the idea of **keyframing**:

i.e.: identifying and describing some of the frames to be used for graph optimization.

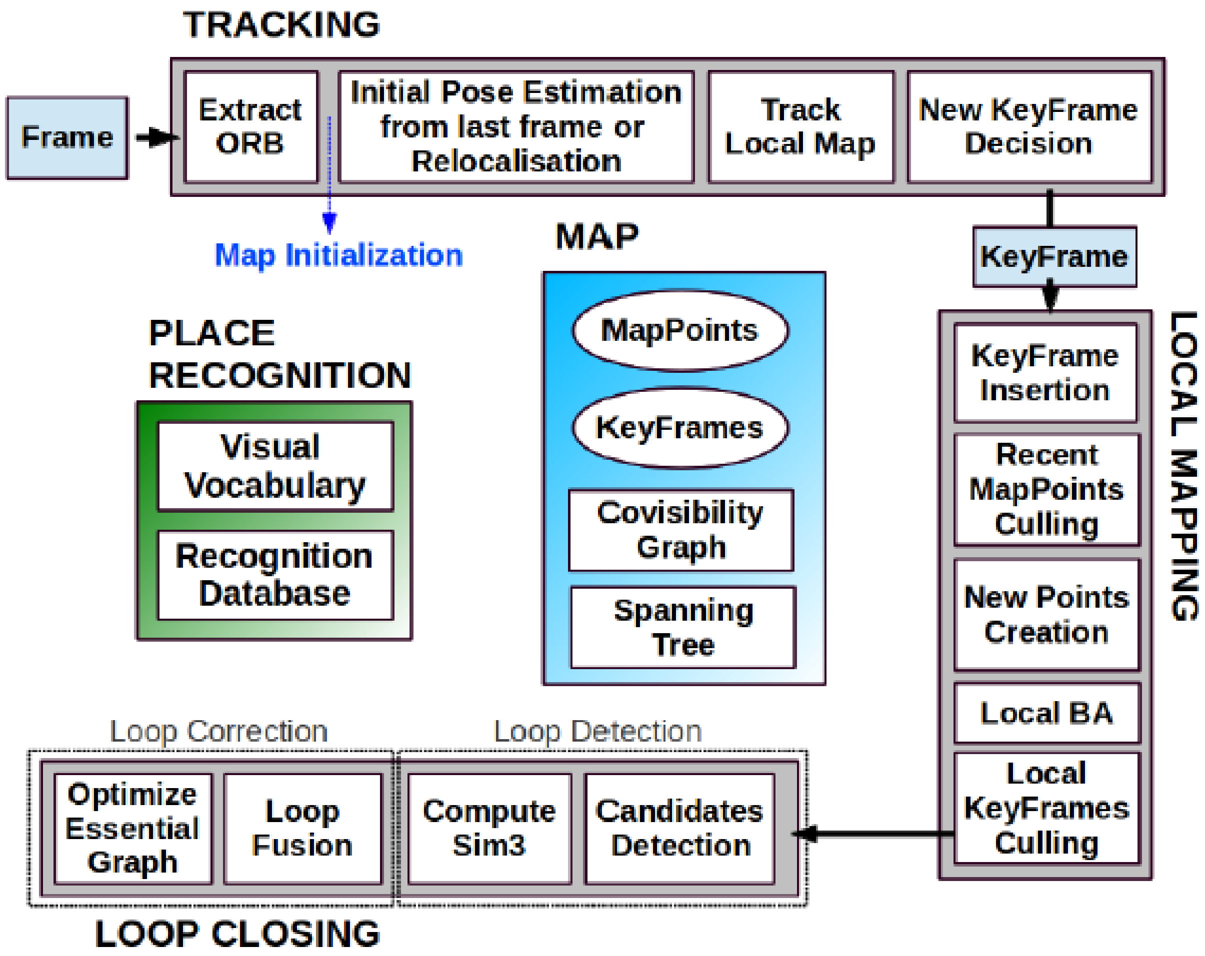
- Bags of words for robust loop closure.
  – What can you tell me about that?

- Co-visibility Graph

ORB SLAM

- The ORBSLAM algorithm is one of the most well performing opensource implementations of visual slam.
- Three parallel threads:
  – tracking,
    - localizing the camera with every frame and deciding when to insert a new keyframe
  – local mapping
    - Processes new keyframes and performs

  local BA to achieve an optimal
- reconstruction in the surroundings of the camera
  – loop closing
    - The loop closing searches for loops with every new keyframe
    - Essential Graph



ORB SLAM on Kitti

Sum up

- Sum up Localization from last time
- Some terminology
- Pose-Landmark Graph Slam
- Example of Linear 1D SLAM
- Non-Linear Optimization approaches
- Bundle Adjustment
- Visual Slam System architecture
- ORBSLAM

Perception for Autonomous Systems 31392:

Visual SLAM

Simultaneous Localization & Mapping

Lecturer: Evangelos Boukas—PhD