

# BALM: Bundle Adjustment for Lidar Mapping [1]

Jinxi Xiao, Yi'ang Ju

7th May

- 1 Introduction
- 2 Direct BA Formulation
  - Plane feature
  - Line feature
  - General view of the Loss function
- 3 Second Order Approximation
  - Derivative
  - Derive the second order approximation
- 4 Codes and Implementations
- 5 Rethink
- 6 Appendix
- 7 Reference

# Motivations

- Odometry obtained from methods like ICP contains a certain degree of errors. And when the errors get accumulated, we may observe a drift in the trajectory.
- As a result, we want to optimize a collection of key frames, both in the respect of Lidar **poses** as well as **feature points**, just like Bundle adjustment(BA) in visual SLAM.
- And this variant is named as BALM.

# Difference between BALM and BA

Generally, for Lidar SLAM, exact point matching is infeasible.

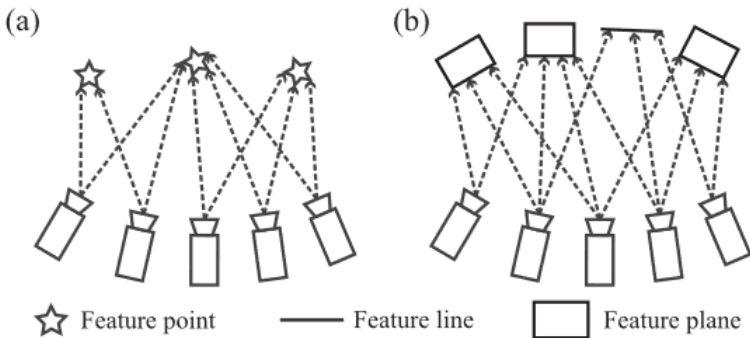


Figure: Difference between BALM and BA

# Main Idea for BALM

- 1 It is most similar to multi-view registration.
- 2 To defining a metric that effectively evaluates the **alignment quality** of sparse points from all scans and, in the meantime, allows efficient optimization.
- 3 Use the geometric information of the point cloud, such as **plane** features and **line** features.

- 1 Introduction
- 2 Direct BA Formulation
  - Plane feature
  - Line feature
  - General view of the Loss function
- 3 Second Order Approximation
  - Derivative
  - Derive the second order approximation
- 4 Codes and Implementations
- 5 Rethink
- 6 Appendix
- 7 Reference

# Terminologies

- There are  $M$  scans, and denote the poses of scans as  $\mathbf{T} = (\mathbf{T}_1, \dots, \mathbf{T}_M)$ , where  $\mathbf{T}_j = (\mathbf{R}_j, \mathbf{t}_j) \in SO(3) \times \mathbb{R}^3$ .
- Feature points correspond to the **same feature**(plane or line):  $\mathbf{p}_{f_i} (i = 1, \dots, N)$ . Suppose  $i$ -th feature point is drawn from the  $s_i$ -th scan, where  $s_i \in \{1, \dots, M\}$ .

- All feature points can be convert to the global frame by

$$\mathbf{p}_i = \mathbf{R}_{s_i} \mathbf{p}_{f_i} + \mathbf{t}_{s_i}, \quad i = 1, \dots, N$$

- Denote  $\mathbf{q}$  to be a point on the plane(line), and  $\mathbf{n}$  to be the normal vector of the plane(direction of the line).

# Loss function for a plane feature

$$\begin{aligned}
 \min_{\mathbf{T}, \mathbf{n}, \mathbf{q}} \quad & \frac{1}{N} \sum_{i=1}^N \left( \mathbf{n}^T (\mathbf{p}_i - \mathbf{q}) \right)^2 \\
 \text{s.t.} \quad & \mathbf{T}_j \in SO(3) \times \mathbb{R}^3 \\
 & \mathbf{n}, \mathbf{q} \in \mathbb{R}^3 \\
 & \|\mathbf{n}\|_2 = 1
 \end{aligned} \tag{1}$$

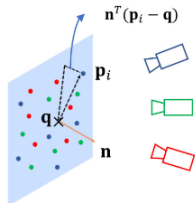


Figure: A plane feature



# Derive the solution

Does this optimization problem even have an optimal solution?

## Assumption 1

It is similar to ICP algorithm. That is, there could be infinitely many solutions, or exactly one solution. In the latter case, the local minimum would be the global minimum.

Notice that  $\mathbf{q}$  is not constraint, find the derivative of  $\mathbf{q}$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{q}} &= 0 \\ \Rightarrow \frac{1}{N} \sum_{i=1}^N -2\mathbf{nn}^T(\mathbf{p}_i - \mathbf{q}) &= 0 \\ \Rightarrow \mathbf{nn}^T \frac{1}{N} \sum_{i=1}^N (\mathbf{p}_i - \mathbf{q}) &= 0\end{aligned}\tag{2}$$

## Derive the solution

It would be impossible to solve (2) if we have got to consider all the variables. But luckily,

$$\mathbf{q} = \bar{\mathbf{p}} = \frac{1}{N} \sum_{i=1}^N \mathbf{p}_i \quad (3)$$

is always a solution to (2).

Also notice that any point on the plane can be chosen, so we should have infinitely many solutions to  $\mathbf{q}$  in theory.

### Assumption 2

Although the point can move freely on the plane, we assume that the final loss function would be the same no matter what  $\mathbf{q}$  is.

As a result, we derive a solution for  $\mathbf{q}$  in (3).

## Derive the solution

Plug (3) to (1), we can rewrite (1) as:

$$\min_{\mathbf{T}, \mathbf{n}} \frac{1}{N} \sum_{i=1}^N \mathbf{n}^T (\mathbf{p}_i - \bar{\mathbf{p}}) (\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{n} = \mathbf{n}^T \mathbf{A} \mathbf{n} \quad (4)$$

where

$$\mathbf{A} = \frac{1}{N} \sum_{i=1}^N (\mathbf{p}_i - \bar{\mathbf{p}}) (\mathbf{p}_i - \bar{\mathbf{p}})^T \quad (5)$$

since  $\mathbf{A}$  is real symmetric and PSD, we can find the minimum value to be the minimum eigenvalue<sup>1</sup> and  $\mathbf{n}$  to be the corresponding eigenvector.

Thus, we have found a simpler representation of (1):

$$\min_{\mathbf{T}} \lambda_3(\mathbf{A}) \quad (6)$$

---

<sup>1</sup>proof see here

## Loss function for a line feature

$$\begin{aligned}
 \min_{\mathbf{T}, \mathbf{n}, \mathbf{q}} \quad & \frac{1}{N} \sum_{i=1}^N \left\| \left( \mathbf{I} - \mathbf{n}\mathbf{n}^T \right) (\mathbf{p}_i - \mathbf{q}) \right\|_2^2 \\
 \text{s.t.} \quad & \mathbf{T} \in SO(3) \times \mathbb{R}^3 \\
 & \mathbf{n}, \mathbf{q} \in \mathbb{R}^3 \\
 & \|\mathbf{n}\|_2 = 1
 \end{aligned} \tag{7}$$

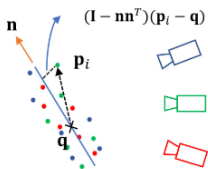


Figure: A line feature

## Derive the solution

Similarly, we can find one solution for  $\mathbf{q}$  the same as (3). Define  $\mathbf{x}_i = \mathbf{p}_i - \bar{\mathbf{p}}$  and rewrite (7) to:

$$\begin{aligned}
 & \min_{\mathbf{T}, \mathbf{n}} \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i^T \left( \mathbf{I} - \mathbf{n}\mathbf{n}^T \right) \mathbf{x}_i \\
 &= \min_{\mathbf{T}, \mathbf{n}} \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i^T \mathbf{x}_i - \mathbf{x}_i^T \mathbf{n}\mathbf{n}^T \mathbf{x}_i \\
 &= \min_{\mathbf{T}, \mathbf{n}} \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i^T \mathbf{x}_i - \mathbf{n}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{n} \\
 &= \min_{\mathbf{T}, \mathbf{n}} \left( \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i^T \mathbf{x}_i \right) - \mathbf{n}^T \mathbf{A} \mathbf{n}
 \end{aligned} \tag{8}$$

# Derive the solution

Notice that

$$\text{tr}(\mathbf{x}_i \mathbf{x}_i^T) = \mathbf{x}_i^T \mathbf{x}_i$$

We can conclude (7) to

$$\min_{\mathbf{T}} \text{tr}(\mathbf{A}) - \lambda_1(\mathbf{A}) = \min_{\mathbf{T}} \lambda_2(\mathbf{A}) + \lambda_3(\mathbf{A}) \quad (9)$$

# The loss function

Now BALM is equivalent to minimizing

$$\lambda_k(\mathbf{p}(\mathbf{T})) \quad (10)$$

where  $\mathbf{p} = [\mathbf{p}_1^T \cdots \mathbf{p}_N^T]^T$  is the vector of all feature points corresponding to the same feature.

## A geometric view

Eigenvalues of the covariance denotes the trend of spread of the data. For a plane feature, it has a degree of freedom of 2, which corresponds to  $\lambda_1$  and  $\lambda_2$ , thus we only need to minimize  $\lambda_3$ . The same applies to line features.

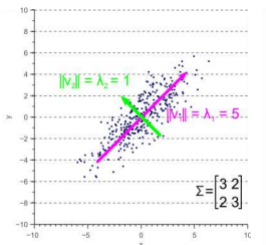


Figure: A visual explanation of eigenvalues of the covariance.<sup>2</sup>

<sup>2</sup>Figure is from here



- 1 Introduction
- 2 Direct BA Formulation
  - Plane feature
  - Line feature
  - General view of the Loss function
- 3 Second Order Approximation**
  - Derivative
  - Derive the second order approximation
- 4 Codes and Implementations
- 5 Rethink
- 6 Appendix
- 7 Reference

# First-order derivative

## Theorem

For a group of points,  $\mathbf{p}_i$  ( $i = 1, \dots, N$ ) and the covariance matrix  $\mathbf{A}$  defined in (5). Assume  $\mathbf{A}$  has eigenvalues  $\lambda_k$  corresponding to eigenvectors  $\mathbf{u}_k$  ( $k = 1, 2, 3$ ), then

$$\frac{\partial \lambda_k}{\partial \mathbf{p}_i} = \frac{2}{N} (\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{u}_k \mathbf{u}_k^T, \quad (11)$$

where the  $\bar{\mathbf{p}}$  is the average of the  $N$  points as in (3).

# Second-order derivative

## Theorem

If  $\lambda_i \neq \lambda_k$  when  $i \neq k$ , then

$$\frac{\partial^2 \lambda_k}{\partial \mathbf{p}_j \partial \mathbf{p}_i} = \begin{cases} \frac{2}{N} \left( \frac{N-1}{N} \mathbf{u}_k \mathbf{u}_k^T + \mathbf{u}_k (\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{U} \mathbf{F}_k^{\mathbf{p}_j} \right. \\ \left. + \mathbf{U} \mathbf{F}_k^{\mathbf{p}_j} (\mathbf{u}_k^T (\mathbf{p}_i - \bar{\mathbf{p}})) \right), & i = j \\ \frac{2}{N} \left( -\frac{1}{N} \mathbf{u}_k \mathbf{u}_k^T + \mathbf{u}_k (\mathbf{p}_i - \bar{\mathbf{p}})^T \mathbf{U} \mathbf{F}_k^{\mathbf{p}_j} \right. \\ \left. + \mathbf{U} \mathbf{F}_k^{\mathbf{p}_j} (\mathbf{u}_k^T (\mathbf{p}_i - \bar{\mathbf{p}})) \right), & i \neq j \end{cases} \quad (12)$$

# Second-order derivative

## Theorem

where

$$\mathbf{F}_k^{\mathbf{p}_j} = \begin{bmatrix} \mathbf{F}_{1,k}^{\mathbf{p}_j} \\ \mathbf{F}_{2,k}^{\mathbf{p}_j} \\ \mathbf{F}_{3,k}^{\mathbf{p}_j} \end{bmatrix} \in \mathbb{R}^{3 \times 3}, \quad \mathbf{U} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3],$$

$$\mathbf{F}_{m,n}^{\mathbf{p}_j} = \begin{cases} \frac{(\mathbf{p}_j - \bar{\mathbf{p}})^T}{N(\lambda_n - \lambda_m)} (\mathbf{u}_m \mathbf{u}_n^T + \mathbf{u}_n \mathbf{u}_m^T), & m \neq n \\ \mathbf{0}_{1 \times 3} & , m = n \end{cases}$$

What if  $\lambda_i = \lambda_k$  for  $i \neq k$  ?

## Derive the second order approximation

Approximate the cost function (10) by its second order approximation as:

$$\lambda_k(\mathbf{p} + \delta\mathbf{p}) \approx \lambda_k(\mathbf{p}) + \mathbf{J}(\mathbf{p})\delta\mathbf{p} + \frac{1}{2}\delta\mathbf{p}^T \mathbf{H}(\mathbf{p})\delta\mathbf{p} \quad (13)$$

where  $\mathbf{J}(\mathbf{p})$  is the Jacobian matrix with  $i$ -th elements in (11) and  $\mathbf{H}(\mathbf{p})$  is the Hessian matrix with  $i$ -th row,  $j$ -th column elements in (12).

Recall that the point vector  $\mathbf{p}$  is further dependent on the scan poses.

Perturbing a pose  $\mathbf{T}_j$  in its tangent plane  $\delta\mathbf{T}_j = [\phi_j^T \quad \delta\mathbf{t}_j^T]^T$  using the  $\boxplus$  operation defined in [2], we have

$$\mathbf{T}_j = (\mathbf{R}_j, \mathbf{t}_j); \quad \mathbf{T}_j \boxplus \delta\mathbf{T}_j = (\mathbf{R}_j \exp(\phi_j^\wedge), \mathbf{t}_j + \delta\mathbf{t}_j) \quad (14)$$

and

$$\mathbf{p}_i = \mathbf{R}_{s_i} \exp(\phi_{s_i}^\wedge) \mathbf{p}_{f_i} + \mathbf{t}_{s_i}; \quad \frac{\delta\mathbf{p}_i}{\delta\mathbf{T}_{s_i}} = [-\mathbf{R}_{s_i}(\mathbf{p}_{f_i})^\wedge \quad \mathbf{I}] \quad (15)$$

## Derive the second order approximation

$$\mathbf{D} = \frac{\delta \mathbf{p}}{\delta \mathbf{T}} = \begin{bmatrix} \vdots & & \\ \cdots & \mathbf{D}_{ij} & \cdots \\ \vdots & & \end{bmatrix} \in \mathbb{R}^{3N \times 6M} \quad (16)$$

$$\mathbf{D}_{ij} = \begin{cases} \frac{\delta \mathbf{p}_i}{\delta \mathbf{T}_{s_i}} & \text{for } j = s_i \in \{1, \dots, M\} \\ \mathbf{0}_{3 \times 6} & \text{for else} \end{cases} \quad (17)$$

$$\lambda_k(\mathbf{T} \boxplus \delta \mathbf{T}) \approx \lambda_k(\mathbf{T}) + \underbrace{\mathbf{J} \mathbf{D}}_{\bar{\mathbf{J}}} \delta \mathbf{T} + \frac{1}{2} \delta \mathbf{T}^T \underbrace{\mathbf{D}^T \mathbf{H} \mathbf{D}}_{\bar{\mathbf{H}}} \delta \mathbf{T} \quad (18)$$

Thus using Levenberg-Marquardt to solve

$$(\bar{\mathbf{H}}(\mathbf{T}) + \mu I) \delta \mathbf{T}^* = -\bar{\mathbf{J}}(\mathbf{T})^T \quad (19)$$

- 1 Introduction
- 2 Direct BA Formulation
  - Plane feature
  - Line feature
  - General view of the Loss function
- 3 Second Order Approximation
  - Derivative
  - Derive the second order approximation
- 4 Codes and Implementations**
- 5 Rethink
- 6 Appendix
- 7 Reference

# Codes and Implementations

Some Engineering Techniques in BALM:

- Techniques for feature extraction
- Adaptive voxelization
- BALM with LOAM[3] as frontend



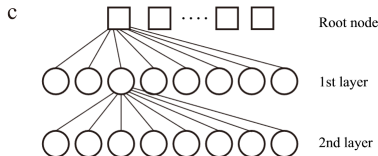
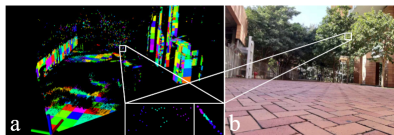
# Techniques for feature extraction

BALM uses improved feature extraction method of LOAM, which includes:

- Filtering out excessively close points
- Assigning scan ID(VFov angle) for each point
- Extract edge points and planar points based on smoothness of local surface(curvature)
- Evenly distribution of feature points by separating scan into 4 sub-regions
- Skip points on parallel lines and in occluded regions

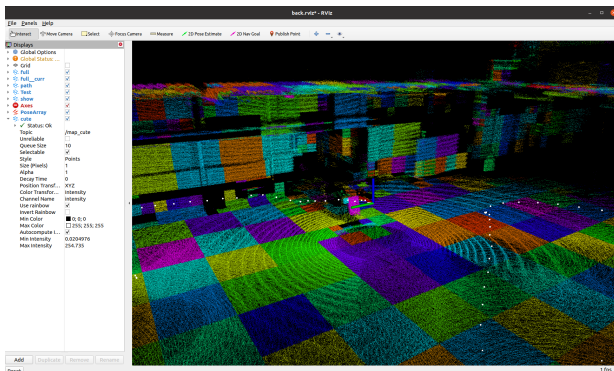
# Adaptive voxelization

- Better than full Kd-tree under scene with large planes or long edges(early termination)
- Repeatedly voxelize the 3D space from a default size
- If all feature points (from all scans) in the current voxel does not lie on the same plane or edge, breaks it into eight octants unless reaches minimal size(0.125m)

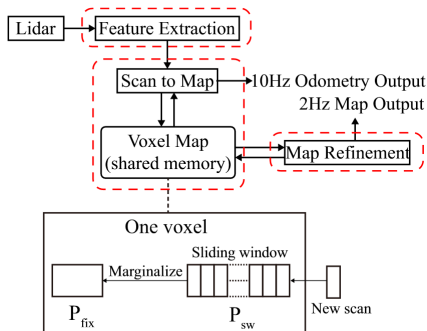


# Adaptive voxelization

- To reduce the depth of the octree, use a set of octrees indexed by a Hash table
- When searching for feature correspondences, only check feature points in nearby voxels

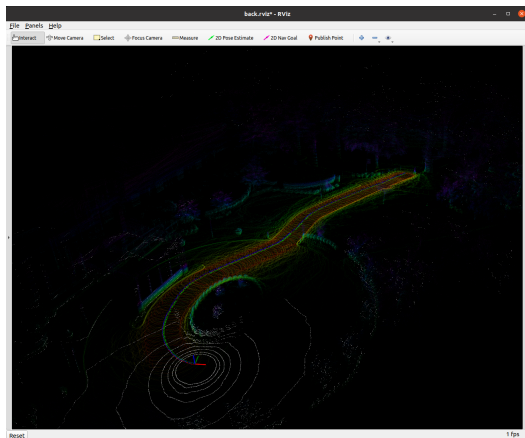


## BALM with LOAM as frontend



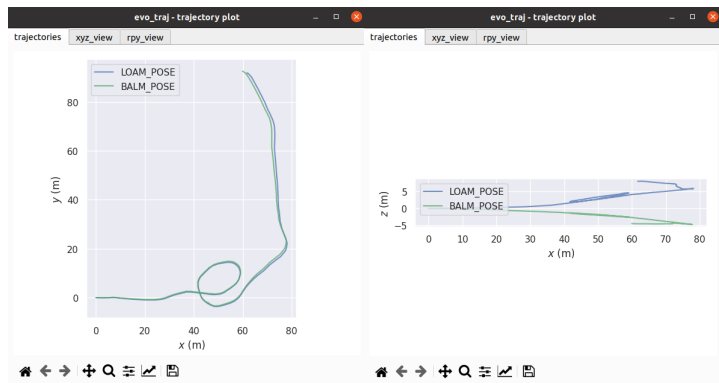
- Adaptive voxel map is used in odometry to speedup the matching process.
- Points are searched with respect to the voxel it lies in and added to the leaf node of the corresponding octree.

# BALM with LOAM as frontend



- After pushing a certain number of new scans to the voxel map, a map-refinement is triggered: two voxel maps form a local BA on a sliding window of lidar poses.

# Tests on datasets



Tested on Stevens-VLP16-Dataset, BALM shows lower drift on z-axis.

- 1 Introduction
- 2 Direct BA Formulation
  - Plane feature
  - Line feature
  - General view of the Loss function
- 3 Second Order Approximation
  - Derivative
  - Derive the second order approximation
- 4 Codes and Implementations
- 5 Rethink**
- 6 Appendix
- 7 Reference

# Are we actually optimizing the 3D points?

- **Maybe Not**

- Different from visual SLAM, we have direct depth info. And once we have the pose of the lidar, 3D points are fixed.
- Recall the loss function (10), which only includes  $\mathbf{T}$  as the optimization variable, thus we are only optimizing lidar poses.

*"It is most similar to multi-view registration."*



# Why not use loss function in LOAM?

line distance:

$$d_{\mathcal{E}} = \frac{|(\tilde{X}_{(k+1,i)}^L - \bar{X}_{(k,j)}^L) \times (\tilde{X}_{(k+1,i)}^L - \bar{X}_{(k,l)}^L)|}{|\bar{X}_{(k,j)}^L - \bar{X}_{(k,l)}^L|} \quad (20)$$

plane distance:

$$d_{\mathcal{H}} = \frac{\left| \begin{array}{c} (\tilde{X}_{(k+1,i)}^L - \bar{X}_{(k,j)}^L) \\ ((\bar{X}_{(k,j)}^L - \bar{X}_{(k,l)}^L) \times (\bar{X}_{(k,j)}^L - \bar{X}_{(k,m)}^L)) \end{array} \right|}{|(\bar{X}_{(k,j)}^L - \bar{X}_{(k,l)}^L) \times (\bar{X}_{(k,j)}^L - \bar{X}_{(k,m)}^L)|} \quad (21)$$

# Why not use loss function in LOAM?

- Eigenvalues of the covariance has great properties, and they are important in the step of building the voxel map.
- Loss function in LOAM are designed for calculating the relative transformation between successive frames, which is mainly used for odometry part. And line distance requires two points on the line while plane distance require three points.

# Why not use loss function in LOAM?

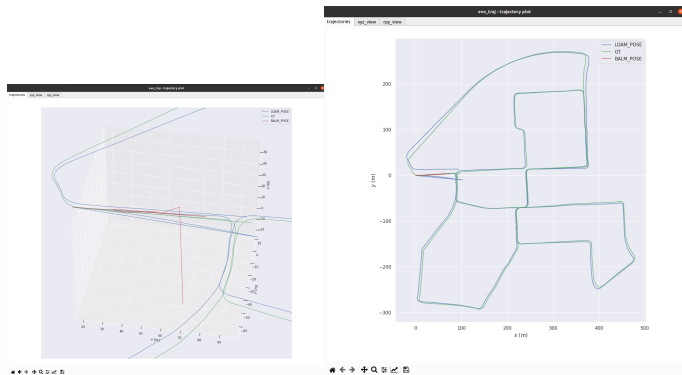
## In class discussion

Prof. mentions one thing, it is that when a loop closure occurs, we will perform a pose graph optimization at first. Then the whole map may be changed, including the feature points. As a result, we may need to build the voxel map once again. However, if we have an explicit (or implicit, to be honest I still cannot fully understand this) expression for BA, maybe we can move the voxel map along with those feature points, thus save quite a lot of time on building the map.

## Summary:

This discussion is highly related to the full SLAM system. In order to fully understand, we may need to write one ourselves.

# Excessive time used in solving BA may cause crashing



- When adapted to velodyne-64 datasets, may crash in real-time frame rate.
- Slower down ros bag play rate can help, but still crash after some time.

# Take away home messages

- Explore eigenvalues of the covariance matrix of point clouds.
  - DIMENSIONALITY BASED SCALE SELECTION IN 3D LIDAR POINT CLOUDS[4]: use eigenvalues to find the most suitable voxel size.
  - Self-Supervised Depth Correction of Lidar Measurements From Map Consistency Loss[5]: use eigenvalues as the loss function of the neural network.
- Build a more structured map(voxel map) to speed up the process.

- 1 Introduction
- 2 Direct BA Formulation
  - Plane feature
  - Line feature
  - General view of the Loss function
- 3 Second Order Approximation
  - Derivative
  - Derive the second order approximation
- 4 Codes and Implementations
- 5 Rethink
- 6 Appendix**
- 7 Reference

## Another proof for First-order derivative

Notice that

$$\mathbf{A} = \left( \frac{1}{N} \sum_{i=1}^N \mathbf{p}_i \mathbf{p}_i^T \right) - \bar{\mathbf{p}} \bar{\mathbf{p}}^T$$

we can conclude that

$$\frac{\partial \mathbf{A}}{\partial x_i} = \frac{1}{N} \begin{bmatrix} 2(x_i - \bar{x}) & y_i - \bar{y} & z_i - \bar{z} \\ y_i - \bar{y} & & \\ z_i - \bar{z} & & \end{bmatrix} \quad (22)$$

as a result






$$\frac{\partial \lambda_k}{\partial x_i} = \frac{2}{N} \begin{bmatrix} u_{k_1}^2 \\ u_{k_1} u_{k_2} \\ u_{k_1} u_{k_3} \end{bmatrix} \cdot \begin{bmatrix} x_i - \bar{x} \\ y_i - \bar{y} \\ z_i - \bar{z} \end{bmatrix} \quad (23)$$

and finally

$$\frac{\partial \lambda_k}{\partial \mathbf{p}_i} = u_k^T \frac{\partial \mathbf{A}}{\partial \mathbf{p}_i} u_k = \frac{2}{N} u_k u_k^T (\mathbf{p}_i - \bar{\mathbf{p}}) \in \mathbb{R}^3 \quad (24)$$

- 1 Introduction
- 2 Direct BA Formulation
  - Plane feature
  - Line feature
  - General view of the Loss function
- 3 Second Order Approximation
  - Derivative
  - Derive the second order approximation
- 4 Codes and Implementations
- 5 Rethink
- 6 Appendix
- 7 Reference



-  Z. Liu and F. Zhang, “Balm: Bundle adjustment for lidar mapping,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.
-  C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, “Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds,” *Information Fusion*, vol. 14, p. 57–77, Jan. 2013.
-  J. Zhang and S. Singh, “Loam: Lidar odometry and mapping in real-time,” in *Robotics: Science and Systems*, 2014.
-  J. Demantké, C. Mallet, N. David, and B. Vallet, “Dimensionality based scale selection in 3d lidar point clouds,” *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3812, pp. 97–102, 2012.
-  R. Agishev, T. Petříček, and K. Zimmermann, “Self-supervised depth correction of lidar measurements from map consistency loss,” *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4681–4688, 2023.