Project Update

Krister Ulvog

Progress Summary:
My initial intension was to implement work from depth denoising and incorporate point cloud models possibly using method inspired from work such as surface reconstruction literature. However, since depth denoising work from [**1**] is sophisticated and implementation is going to tricky (reviewed on **Section 1**). In addition to that, implementation of Poisson surface reconstruction [2] (probably the most famous surface reconstruction algorithm) was not as clear I expected to be. Since my motivation was to explore point cloud models, I am going to implement point cloud denoising via feature graph learning [**3**] since I am confident about the details (reviewed on **Section 2**) and is also a method that is used in [1].

1. Review of Stanley's [1] work

1.1 Objective

The goal of the literature is to estimate 2 depth maps $\boldsymbol{x_l}$ (left view), $\boldsymbol{x_r}$ (right view) from corrupted depth maps $\boldsymbol{y_l}$, $\boldsymbol{y_r}$ from MAP estimation.

$$\begin{aligned}\underset{x_l}{\operatorname{argmax}}\, P(\boldsymbol{y_l}, \boldsymbol{y_r} \mid \boldsymbol{x_l}, \boldsymbol{x_r}) P(\boldsymbol{x_l}, \boldsymbol{x_r}) &= P(\boldsymbol{y_l}, \boldsymbol{y_r} \mid \boldsymbol{x_l}, \boldsymbol{g}(\boldsymbol{x_l})) P(\boldsymbol{x_l}, \boldsymbol{g}(\boldsymbol{x_l}) \\ &= P(\boldsymbol{y_l} \mid \boldsymbol{x_l})\, P(\boldsymbol{y_r} \mid \boldsymbol{g}(\boldsymbol{x_l}))\, P(\boldsymbol{x_l})\, P(\boldsymbol{g}(\boldsymbol{x_l}))\end{aligned}$$

$\boldsymbol{g}(\boldsymbol{x_l})$ is a warped image of $\boldsymbol{x_l}$. $\boldsymbol{g}(\boldsymbol{x_l})$ is obtained by linear interpolation.

1.2 Warping:

In the literature, left sensor and the right sensor is assumed to be rectified. Therefore, we only need to worry about warping each row.

$$\boldsymbol{g}(\boldsymbol{x_l}) = \boldsymbol{W^T x_l}$$

$\boldsymbol{W}$ is a matrix whose element is defined as such.

$$w_{ij} = \frac{1}{Z} exp\left(-\frac{i - \delta(x_{l,i}) - j}{\sigma_s^2}\right) = \frac{1}{Z} exp\left(-\frac{i - fDx_{l,i}^{-1} - j}{\sigma_s^2}\right)$$

Z is a normalization factor. $\delta(x_{l,i})$ is a disparity (pixel offset to the corresponding point in the right view) at the location i, which is inversely i proportional to the depth $x_{l,i}$.

1.3 Likelihood Term

Noise model used in this literature are signal-dependent noise followed by quantization. For each pixel, we have the following relation.

$$y_i = Q(x_i + n_i)$$

Given x_i^*, n_i is bounded by

$$n^-(y_i, x_i^*) \leq n_i \leq n^+(y_i, x_i^*)$$

The bounds are defined as such.

$$n^-(y_i, x_i^*) = \min_{s.t.\ Q(x_i^* + n_i) = y_i} n_i$$

$$n^{+}(y_i, x_i^*) = \max_{s.t.\ Q(x_i^*+n_i)=y_i} n_i$$

Likelihood can now be express as product of integral of Gaussian PDF.

$$P(\boldsymbol{y_l} \mid \boldsymbol{x_l}) = \prod_{i=1}^{N} \int_{n^-}^{n^+} \frac{1}{\sqrt{2\pi\sigma(x_{l,i})^2}} \exp\left(-\frac{n_i^2}{2\sigma(x_{l,i})^2}\right) dn_i$$

Since we are dealing with signal-dependent noise, $\sigma(x_{l,i})$ is a function of $x_{l,i}$.

1.4 <u>Prior</u>

Prior of $\boldsymbol{x_l}$ is model by

$$P(\boldsymbol{x_l}) = \exp\left(-\frac{\boldsymbol{x_l^T L_l x_l}}{\sigma_p^2}\right)$$

$\boldsymbol{L_l}$ is a graph Laplacian matrix.

1.5 <u>Optimization</u>

The likelihood term and warping operation are both linearized by Taylor expansion and optimized by accelerated gradient descent.

2. <u>Feature Graph Learning for Point Cloud Denoising</u>

2.1 <u>Graph Laplacian Regularizer</u>

Graph signal refers to data that resides on the nodes of a graph, such as functionality of regions on a neural network and temperatures on a sensor network.

Laplacian matrix is usually defined as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$$

$\boldsymbol{W}$ is the adjacency matrix, a symmetric matrix whose element represents the weight of edge(i,j) and $\boldsymbol{D}$ is the degree matrix, a diagonal matrix whose entries is $\sum_{j=1}^{N} w_{i,j}$

Laplacian regularizer is

$$\boldsymbol{x^T L x} = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}\left(x_i - x_j\right)^2$$

If $\boldsymbol{x^T L x}$ is small, most signal energies are in the low graph frequencies, or $\boldsymbol{x}$ is roughly low pass.

2.2 <u>Signal-dependent Graph Laplacian Regularizer</u>

Another form of Laplacian regularizer is to have the weights $w_{i,j}$ to be a function of x_i and x_j.

$$\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{L}(\boldsymbol{x})\boldsymbol{x} = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}(x_i, x_j) \cdot (x_i - x_j)^2$$

We want $w_{i,j}$ to reflect on the similarity between x_i and x_j. As an example, we can represent the weights of bilateral filter as

$$w_{i,j} = \exp\left(-\left(\begin{bmatrix} x_i \\ i \end{bmatrix} - \begin{bmatrix} x_j \\ j \end{bmatrix}\right)^T \begin{bmatrix} 1/\sigma_x^2 & 0 \\ 0 & 1/\sigma_p^2 \end{bmatrix} \left(\begin{bmatrix} x_i \\ i \end{bmatrix} - \begin{bmatrix} x_j \\ j \end{bmatrix}\right)\right)$$

We take this concept and create a general form

$$w_{i,j} = \exp\left(-(\boldsymbol{f_i} - \boldsymbol{f_j})^{\boldsymbol{T}}\boldsymbol{M}(\boldsymbol{f_i} - \boldsymbol{f_j})\right)$$

$\boldsymbol{f_i}$ is the feature vector of $\boldsymbol{x_i}$ The argument of the exponential can also be interpreted as feature distance or Mahalanobis distance. The goal of this literature is to learn the optimal $\boldsymbol{M}$ from point cloud during denoising.

2.3 <u>Optimization of M</u>

We are interested in finding $\boldsymbol{L}(\boldsymbol{x})$ or $\boldsymbol{M}$ that minimizes the following cost.

$$\min_{\substack{\boldsymbol{L} \\ s.t.\ \boldsymbol{L} \succ 0}} \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{L}(\boldsymbol{x})\boldsymbol{x} = \min_{\substack{\boldsymbol{M} \\ s.t.\ \boldsymbol{M} \succ 0}} \sum_{i=1}^{N}\sum_{j=1}^{N} \exp\left(-(\boldsymbol{f_i} - \boldsymbol{f_j})^{\boldsymbol{T}}\boldsymbol{M}(\boldsymbol{f_i} - \boldsymbol{f_j})\right) \cdot (x_i - x_j)^2$$

However, optimization can be unstable since the solution will lead to $m_{i,i} = +\infty$. Therefore, additional constraint is added.

$$\min_{\substack{\boldsymbol{M} \\ s.t.\ \boldsymbol{M} \succ 0 \\ tr(M) \leq C}} \sum_{i=1}^{N}\sum_{j=1}^{N} \exp\left(-(\boldsymbol{f_i} - \boldsymbol{f_j})^{\boldsymbol{T}}\boldsymbol{M}(\boldsymbol{f_i} - \boldsymbol{f_j})\right) \cdot (x_i - x_j)^2$$

2.3.1 <u>Optimization of Diagonal Entries</u>

Here is the formulation of optimization of diagonal entries of **M** given off-diagonal entries.

$$\min_{\substack{\boldsymbol{M} \\ s.t.\ \boldsymbol{M} \succ 0 \\ tr(M) \leq C}} \sum_{i=1}^{N}\sum_{j=1}^{N} \exp\left(-\sum_{k} m_{k,k}\left(\boldsymbol{f_i}(k) - \boldsymbol{f_j}(k)\right)^{\boldsymbol{2}}\right) \cdot (x_i - x_j)^2 + I_S(\boldsymbol{m})$$

Where

$$I_S(\boldsymbol{m}) = \begin{cases} 0, & \boldsymbol{m} \in S \\ \infty, & \text{otherwise} \end{cases}$$

$$S = \left\{\boldsymbol{m} \mid m_{i,i} > \sum_{j \neq i} |m_{i,j}|,\ \sum_{i=1}^{K} m_{i,i} \leq C, \qquad \forall i\right\}$$

Both terms are convex and second term is non-differentiable, which can be solved by Proximal Gradient Method.

2.3.2 *Optimization of Off-diagonal Entries*

Optimization of off-diagonal entries of **M** given diagonal entries, is done by block coordinate descent.

$$\boldsymbol{M} = \begin{bmatrix} m_{1,1} & \boldsymbol{M_{2,1}} \\ \boldsymbol{M_{2,1}} & \boldsymbol{M_{2,2}} \end{bmatrix}$$

The strategy used is to optimize one row and column of off-diagonal entries represented by $\boldsymbol{M_{2,1}}$ in one iteration. Different row and column is selected in the next iteration. By changing solution variable and transforming the constraint, we get

$$\min_{\substack{\boldsymbol{M_{2,1}} \\ s.t.\ m_{1,1}-\boldsymbol{M}_{2,1}^T\boldsymbol{M}_{2,2}^{-1}\boldsymbol{M}_{2,1} > 0 \\ m_{1,1} \leq C - tr(\boldsymbol{M}_{2,2})}} \sum_{i=1}^{N}\sum_{j=1}^{N} \exp\left(-\left(\boldsymbol{f_i}-\boldsymbol{f_j}\right)^{\boldsymbol{T}} \boldsymbol{M}\left(\boldsymbol{f_i}-\boldsymbol{f_j}\right)\right) \cdot \left(x_i - x_j\right)^2$$

2.4 Final Algorithm

The objective is to

$$\underset{\substack{\boldsymbol{X},\boldsymbol{M} \\ s.t.\ \boldsymbol{M}\succ 0 \\ tr(\boldsymbol{M})\leq C}}{\operatorname{argmax}} \left|\left|\boldsymbol{Y}-\boldsymbol{X}\right|\right|_F^2 + \gamma \mathrm{tr}((\mathbf{TX}-\mathbf{C})^{\mathrm{T}}\boldsymbol{L}(\boldsymbol{M})(\boldsymbol{TX}-\boldsymbol{C}))$$

$\boldsymbol{Y}$ is the noisy point cloud, $\boldsymbol{X}$ is the denoised point cloud. $\mathbf{T}$ is a sampling matrix. $(\mathbf{TX}-\mathbf{C})$ is patches extracted from the point cloud. We jointly solve for **X** and **M** by alternating the optimization.

Given **M** and fixing **L**, we can solve for **X** by solving the following set of equations.

$$\left(\gamma\boldsymbol{T^TLT}+\boldsymbol{I}\right)\boldsymbol{X_x} = \boldsymbol{Y_x} + \gamma\boldsymbol{T^TLC_x}$$
$$\left(\gamma\boldsymbol{T^TLT}+\boldsymbol{I}\right)\boldsymbol{X_y} = \boldsymbol{Y_y} + \gamma\boldsymbol{T^TLC_y}$$
$$\left(\gamma\boldsymbol{T^TLT}+\boldsymbol{I}\right)\boldsymbol{X_z} = \boldsymbol{Y_z} + \gamma\boldsymbol{T^TLC_z}$$

Where
$\boldsymbol{X} = [\boldsymbol{X_x}, \boldsymbol{X_y}, \boldsymbol{X_z}], \boldsymbol{Y} = \left[\boldsymbol{Y_x}, \boldsymbol{Y_y}, \boldsymbol{Y_z}\right], \boldsymbol{C} = [\boldsymbol{C_x}, \boldsymbol{C_y}, \boldsymbol{C_z}]$.
This equation can be solved by conjugate gradient methods, such as the LSQR algorithm.

Optimization of **M** can be done by using the method introduced in previous section. The pair (x_i, x_j) can be replaced by corresponding pair $(\boldsymbol{p_i}, \boldsymbol{p_j})$ from the point cloud. The corresponding pair is extracted by first constructing patches $\boldsymbol{V}$ from the point cloud. For a point in a patch $\boldsymbol{p_i} \in \boldsymbol{V_i}$, the nearest neighbor to $\boldsymbol{p_i}$ from a neighboring patch $\boldsymbol{V_j}$ is the corresponding pair $\boldsymbol{p_j}$.

The feature vector $\boldsymbol{f_i}$ is replaced by the concatenation of xyz coordinate of the point and normal vector $\left[p_x, p_y, p_z, n_x, n_y, n_z\right]^T$

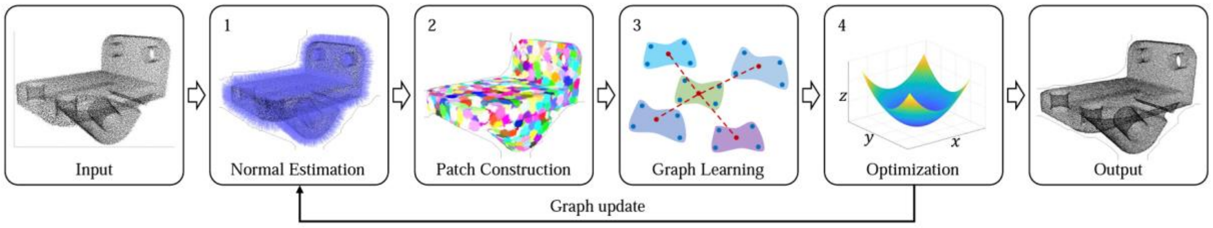

Figure 1. Flowchart of the algorithm

<u>Pseudocode</u>

Input:

noisy point cloud **Y**, number of patches M, number of nearest neighbors k, number of nearest patches ϵ, trace constraint C, optimization parameter γ

Output:

Denoised point cloud **X**

```
Initialize $\boldsymbol{X}$ as $\boldsymbol{Y}$
iteration i = 0, 1, 2, …, k
    1.  Estimate normal of $\boldsymbol{X}$
    2.  $\boldsymbol{C} \leftarrow$ Uniform down sample of $\boldsymbol{X}$
    3.  for each $c_i$ in $\boldsymbol{C}$, find k nearest neighbors in $\boldsymbol{X}$, define patch $\boldsymbol{V_i}$ as
        $\boldsymbol{V_i} \leftarrow \{ 0 \cup \mathrm{knn}(c_i) - c_i \}$
    4.  for each $\boldsymbol{V_i}$, find $\epsilon$ nearest patches $\boldsymbol{B_{j,i}}$
    5.  for each $\boldsymbol{p_i} \in \boldsymbol{V_i}$, find corresponding point $\boldsymbol{p_j} \in \boldsymbol{B_{j,i}}$
    6.  Compute $(\boldsymbol{p_i} - \boldsymbol{p_j})^2$ for all (i,j)'s
    7.  Compute $(\boldsymbol{f_i} - \boldsymbol{f_j})^2$ for all (i,j)'s
    8.  Optimize M
        6.1 initialize M as I
        6.2 repeat the following until convergence
                Optimization of off-diagonal entries of M
                Optimization of diagonal entries of M
    9.  Compute Adjacency Matrix $\boldsymbol{W}$, Laplacian Matrix $\boldsymbol{L}$.
    10. Solve for $\boldsymbol{X}$ using conjugate gradient method
```

<u>References</u>

[1] X. Zhang, G. Cheung, J. Pang, Y. Sanghvi, Y., A. Gnanasambandam., & S.H. Chan. " Graph-Based Depth Denoising & Dequantization for Point Cloud Enhancement," *ArXiv, abs/2111.04946*, Oct. 2022.

[2] M. Kazhdan, M. Bolitho, & H. Hoppe. (2006). "Poisson Surface Reconstruction". In *Symposium on Geometry Processing*. The Eurographics Association.

[3] W. Hu, X. Gao, G. Cheung, and Z. Guo, "Feature graph learning for 3D point cloud denoising," *IEEE Trans. Signal Process.*, vol. 68, pp. 2841–2856, 2020.