CS 6790 : Geometry & Photometry-based Computer Vision
Project Report
Photometric Bundle Adjustment for Dense Multi-View 3D Modeling

Vishwesh (CH16B024) and Gautam (CS16B009)
Github link: https://github.com/cs-9/PBA

19th July 2020

Contents

---

1 Introduction

This paper[3] focuses on jointly refining shape and camera parameters by minimizing photometric reprojection error between generated model images and observed images. This minimization is performed using a gradient descent on the energy function accounting for visibility changes and optimizes for both shape and camera parameters. This paper is used as a last refinement step in the 3D reconstruction pipeline.
The energy function we minimize is:

$$E(\mathbf{X}, \mathbf{T}, \mathbf{\Pi}) = \sum_j A_j \int_{\mathsf{T}} \left(I_i(\Pi_i(\mathbf{x}(\mathbf{u}))) - T(\mathbf{u})\right)^2 \boldsymbol{\alpha} \cdot \mathbf{n}_j \, \nu_{\mathcal{S}}(\mathbf{x}(\mathbf{u})) \, \mathrm{d}\mathbf{u}$$

Figure 1: Photometric loss

Where,
$\mathbf{X} = [\mathbf{x_1}....\mathbf{x_n}]$ is the array of vertices such that $\mathbf{x_k}$ is the k^{th} vertex of the polyhedral representation of S
S_j is the j_{th} triangle in the polyhedral representation
$\mathbf{n_j}$is the normal of the triangle and $\mathbf{u}$ parameterizes the surface area A_j of triange S_j
The points in the triangle follow barycentric coordinates, s.t $\mathbf{u} = (u, v) \in T = \{(u, v)|u \in [0, 1]\ and\ v \in [0, 1 - u]\}$
Additionally $\mathbf{\Pi}$ represents the camera parameters, T represents the texture, v_s represents the visibility function. For this equation $\alpha = f_x f_y \frac{\mathbf{d}}{\mathbf{d_z^3}}$ where $\mathbf{d}$ is the vector from the camera center to the point.

2 Methodology

To solve this particular problem and to get a better scene reconstruction using photometric bundle adjustments, the following methodology was followed. This section describes the overall overview of the methodology. The next section shines more light on the intermediate steps.

1. The first step involved the initial scene $\mathbf{\Omega^0}$ construction using Structure from motion(SFM). This initialization from SFM also gives us the initial camera calibration $\mathbf{\Pi}$ and initial point cloud $\mathbf{X}$. This construction is done using OpenMVG software. In order to deal with the surface, we need to form a mesh. Using this point cloud a surface is constructed using triangular meshes. We used OpenMVS for this process of mesh reconstruction.

2. Since our photometric loss depends on the visibility of mesh from each camera, we calculate visibility of each of the mesh from a particular camera position. Here its assumed each point in the mesh will have same visibility as that of the mesh

3. We next estimate the texture using the images from each camera and the point cloud

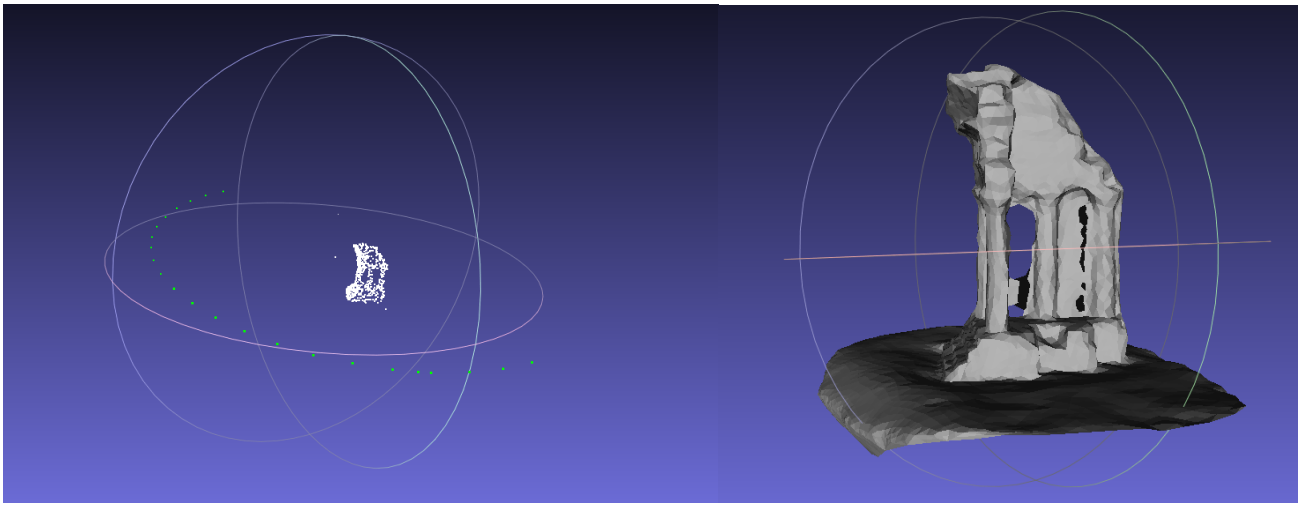
4. Now using all the terms we use the total photometric loss to form the energy function

5. Using the gradients over point cloud $\mathbf{X}$ and camera parameters $\mathbf{\Pi}$, we use gradient descent over it to minimize the energy function/photometric loss

3 Implementation details

3.1 Initilization

The initial calibration $\mathbf{\Pi^0}$ is obtained from openMVG[1] and openMVS[2]. OpenMVG gives us a 3D reconstruction and the inital estimate of camera parameters for the given set of images. openMVS is used to densify the point cloud obtained from openMVG and to construct a mesh.



(a) Output generated by openMVG on the Temple Ring dataset (b) Mesh generated by openMVS

Figure 2: SFM to mesh generation pipeline

3.2 Visibility

Calculation of visibility was done on the mesh level instead of the point level suggested in the paper. We have assumed that each point in the mesh will have the same visibility as that of the mesh. A mesh is termed as visible if the ray from the camera center connecting to the centroid of the mesh does not encounter any other mesh. We can determine if a line segment intersects a triangle by comparing the signed volumes formed by different sections. Using this methodology, we successfully calculated the visibility of a single mesh from a single camera. Now this step was repeated for all the meshes and cameras. Since this step is computation intensive hence it was calculated only once.

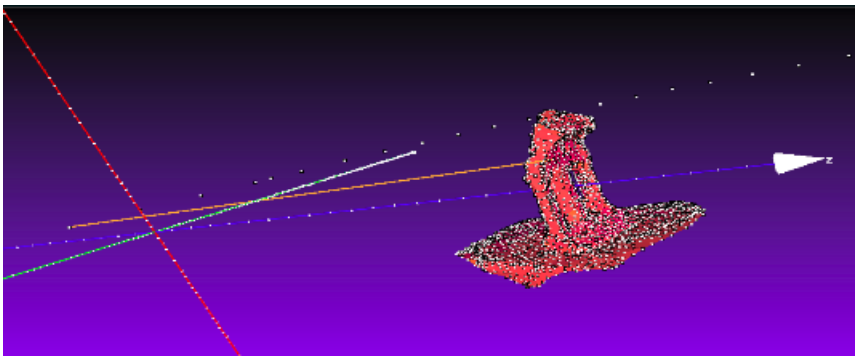


Figure 3: Mesh not visible from the camera

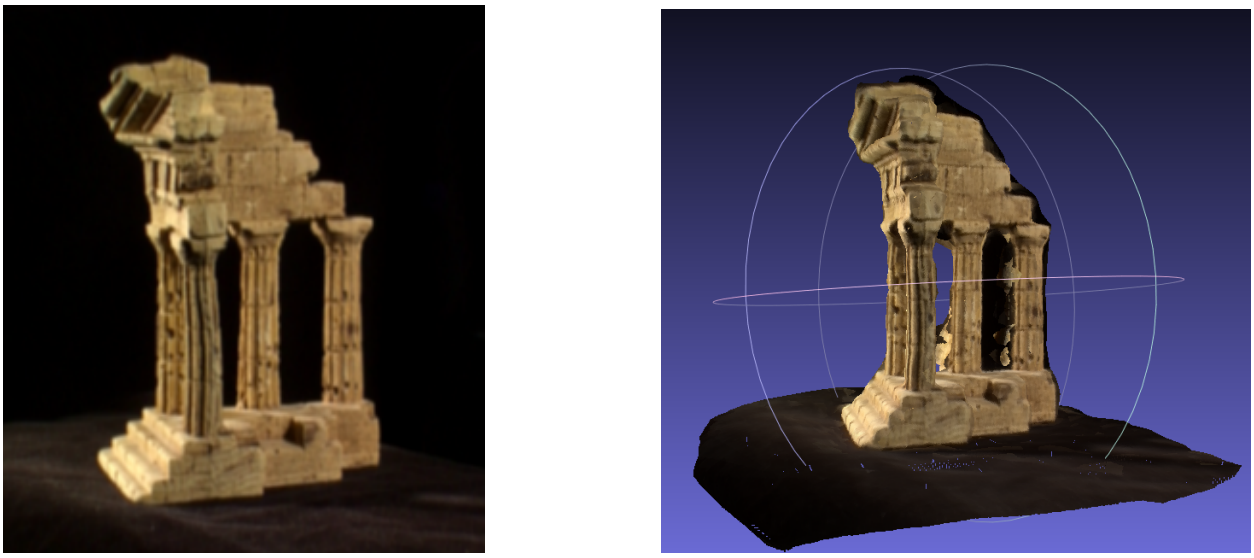
3.3 Texture

The texture at each point is computed using the following equation.

$$\mathbf{T}(x) = \frac{\Sigma_i \mathbf{I}_i(\Pi_i(x))\mathbf{w_i}(x)}{\Sigma_i \mathbf{w_i}(x)}$$

Where,

$\mathbf{T}(x)$ is the texture value at point x
$\mathbf{I_i}(\mathbf{\Pi_i}(\mathbf{x}))$ is the color of the pixel which is the projection of $\mathbf{x}$ on image i using $\mathbf{x} = \mathbf{PX}$ relationship
$\mathbf{w_i}$ here is the weighting faction, here it is $\mathbf{v_s}(\mathbf{x}(\mathbf{u}))$ which is the visibility function
The texture value was computed for 55 points in each mesh face.
Given below is the textured mesh obtained using this equation vs one of the input images.



(a) Input image

(b) Texture generated using the above equation

Figure 4: Texture generation pipeline

3.4 Energy function

$$E(\mathbf{X}, \mathbf{T}, \mathbf{\Pi}) = \sum_j A_j \int_{\mathsf{T}} \left(I_i(\Pi_i(\mathbf{x}(\mathbf{u}))) - T(\mathbf{u})\right)^2 \boldsymbol{\alpha} \cdot \mathbf{n}_j \, \nu_{\mathcal{S}}(\mathbf{x}(\mathbf{u})) \, \mathrm{d}\mathbf{u}$$

Figure 5: Photometric loss

Since we operate in a discretized fashion hence we calculated the integration term by discretizing each mesh into 55 points using a barycentric coordinate system. For each of those points, we fetch the texture, visibility , area, normal vector term. The image term was obtained from pixel value at $\mathbf{x}$ where $\mathbf{x} = \mathbf{PX}$. Combining all these terms we get our energy function.

3.5 Performing Gradient descent

Since the theoritical gradients $\frac{d\mathbf{E}}{dX}$and $\frac{d\mathbf{E}}{d\mathbf{\Pi}}$ consits of very calculation intensive terms which would have taken lots of time to calculate. Also they have used GPUs for their calculations, hence for faster and less intensive calculation method. We have used central difference method on energy function to get our gradient updates and used gradient descents on it.
Let $f(x)$ be our function, we have,

$$\frac{df(x)}{dx} = \frac{f(x+h) - f(x-h)}{2h}$$

For our implemenation, we have used $h = 0.0000001$, for both vertex $\mathbf{X}$ and camera parameters $\mathbf{\Pi}$ gradient calculation.
Finally we used for updates,

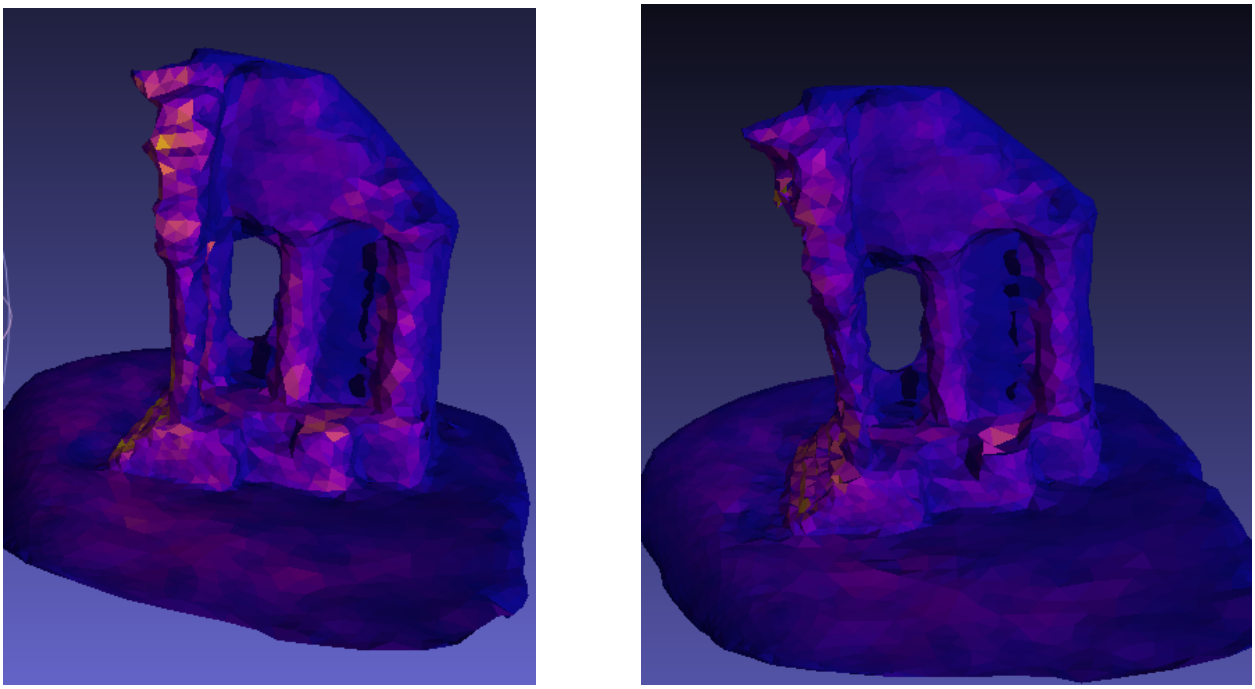
$$\mathbf{X}[t+1] = \mathbf{X}[t] - \lambda_1 \frac{d\mathbf{E}}{dX}$$

$$\mathbf{\Pi}[t+1] = \mathbf{\Pi}[t] - \lambda_2 \frac{d\mathbf{E}}{d\mathbf{\Pi}}$$

4 Results

Photometric Bundle Adjustment was done on 2 datasets: Temple Ring and Dino Ring.

Dataset	Initial Photometric Error	Final Photometric Error
Temple Ring	33.78	31.01
Dino Ring	21.96	21.60



(a) Photometric Error visualized on the mesh before PBA

(b) Photometric Error visualized on the mesh after PBA

Figure 6: Final photometric visualization

We have used plasma colour map denoting yellow regions having high photometric loss, we can see using PBA we have successfully reduced high loss regions.

5 Conclusion

We observed that the loss didn't decrease by much and the results almost looks the same. We have come with some explainations as to why this might happen.

1. Computing visibility function is expensive. In the paper, they made use of a GPU to compute visibility function. We did not consider visibility function updates during gradient descent of camera parameters.
2. openMVG + openMVS does not return good results. OpenMVG seems to be considering only a few cameras. There are a few views of the object where very few (or even no cameras in the case of dino ring) images are directed at.
3. Texture was not handled as explained in the paper. No utilities were found for creating texture maps.
4. Meshing was done with openMVS, so re-meshing could not be done.
5. Due to calculation intensive nature of gradients, we went ahead with numerical gradient calculation. This may have caused error in the gradient updates due to the loss function not being smooth.

References

[1] https://github.com/openMVG/openMVG

[2] https://github.com/cdcseacave/openMVS

[3] Amaël Delaunoy, Marc Pollefeys. Photometric Bundle Adjustment for Dense Multi-View 3D Modeling. 2014. hal-00985811