Structure from Motion (SFM) Algorithms Implementation and Bundle Adjustment Method on Video Input

- Project Proposal -

Aiden & Rainsford

Nov 14 2022

Contents

Overview

The purpose of this project is to build a 3D reconstruction with camera motions from a video. We will implement this first by detecting and matching features in consecutive frames. Then, we will use the damped Newton matrix factorization method to find the structure matrix and motion matrix. Finally, we will adapt the bundle adjustment for a more accurate 3D reconstruction.

1 Background Knowledge

The interest in this field starts from recovering the 3D structure of the real-world object. The most instinctual way is through a video capturing that real-world structure like architecture or a famous sculpture. All we have is an uncontrolled video that has a shaky camera and irrelevant background objects. However, based on this input video, by applying certain algorithms, we could recover not only the 3D reconstruction of this video but also how the camera moves in this scene. This problem is called the structure from motion. During the lecture, we only construct the 3D structure from two views of an image. Also, the two images cameras' intrinsic parameters and ground truth are provided. Besides, we only looked at recovering the structure based on the observation matrix and ignored the camera motion matrix. However, in this project, we will realize all the elements that are missing but essential in making a real-world 3D structure based on video input. Without the context of specific algorithms, it would be hard to explain for the author and hard to understand for readers without knowing the meaning of specific terms in each algorithm. Therefore, more detailed connections to our previous work are detailed introduced in the description part of each milestone.

2 Dataset

For the first three milestones, we will test our algorithms using the Middlebury dataset[8]. The reference of this dataset could be found in the reference section. Data would be compared quantitatively. We will download the 312 temple sampled views from the multi-view stereo data set. The object that we will recover is a synthesized reproduction of "Temple of the Dioskouroi" in Agrigento, Sicily. Each image has a 640×480 size and the resolution of ground truth model is $0.00025m$. We will download the first package shown in the website called the "Temple" data set (77Mb). Our output for each algorithms is simply a consecutive functions that have inputs of images and outputs the point related to each milestone. Notice that this Middlebury dataset is specifically used for 3D construction. Therefore, it doesn't have any specified image matching comparison test.

For our first milestone, we will generate the most significant 15 matching across two random images in the Middlebury dataset and quantitatively eval-

uate the matches. The accuracy of the matches depends on raw eyes. The threshold for good matches should be within roughly 8 pixels, which is constrained by the maximum zooming of the image reader built in MATLAB.

For our second milestone, we will evaluate the matrix factorization with Missing data by a similar approach. We will still use the same 312 images dataset to construct the observation matrix. However, unlike the matches observation that could be done with ease, we will need to take a step further and use the resulted motion matrix and structure matrix to build the structure using the MATLAB function 'triangulate' and look for potential defects in the constructed scene so that we could optimize the result in the next milestone, bundle adjustment. However, since this milestone takes into consideration of missing data, images that shoot the back and the front of the temple are considered most carefully to evaluate the result from missing feature points. This extra difficulty allows us to evaluate the effect of this algorithm regarding the missing data.

For the third milestone, we will use the same approach by iterating the function on those 312 images on the Bundle Adjustment function we wrote. We evaluate the final result using the same approach as before, constructing a 3D scene using the images but with the extra step of utilizing the Bundle Adjustment function. We will compare the resulting 3D construction with the previous result without using the bundle adjustment qualitatively. The main reason we cannot run the quantitative test on the Middlebury dataset is that the function generates discrete points rather than a full 3D reconstruction. So, the accuracy of the reconstruction will be underestimated.

For the last milestones, we will use iphoneXS to film the 3819 classroom at 24:00 at dark outside with no one in the classroom and no computer screen is open. We will set the refresh rate of the filming to be 30 FPS (frame per second) with 1080 resolution with no zooming changes. The person will walk around the classroom in the clockwise direction. The cameraman will stand at the door at first, filming at the whiteboard's direction at first. The cameraman will hold the phone by hand and turn 90 degrees right when reaching the computer's location at the front of the stage. Then, the cameraman will continue to walk until reaching the right front corner of the classroom and turn 90 degrees right again and continue walking at a normal pace until reaching the back of the classroom. During this process, the cameraman's camera has the same direction of the walking direction. When the cameraman reaches the bottom right corner of the classroom, the cameraman makes his last 90 degrees turn and walks until reaching the start of the filming process. This filmed video should take around 30-40 seconds.

3 Four Milestones

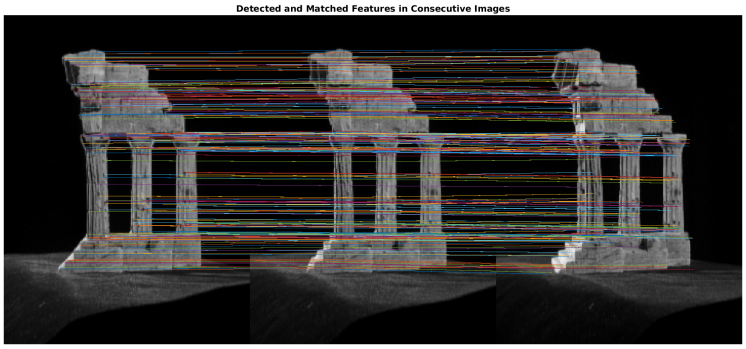
3.1 Feature Detection & Matching in Consecutive Frames

Overview. We will start with the observation matrix. Suppose that we have a video of a scene. There are features in this video that we could capture. We track the image coordinates of these features and rearrange them into a matrix, which is our observation matrix. This observation matrix is what we will work with, containing all the information we need to reconstruct the 3D scene. However, how do we get the observation Matrix? As we explained, the observation matrix is formed by continuously tracked features throughout the filming process.

Description. We use the feature detection method that was implemented in the Feature Detection lab[5] and is originally from Szeliski's textbook[**?**] *Computer Vision: Algorithms and Applications* and his paper *Multi-image matching using multi-scale oriented patches*[2]. For the feature matching method, we use SSD(Sum of Squared Difference) like what we did in the Stereo Disparity Lab[6] as suggested on Piazza by Professor Jerod Weinman. We calculate SSD using a convolution on a SSD window and find matched features on a search window.

Possible Risks. One problem with these feature detection and matching is inaccuracy. In our feature detection method, the algorithm detects features at a particular scale, and this means the detected features are not robust against different scales. Similarly, since our feature matching method simply uses SSD(Sum of Squared Difference), there is a high probability of false matched features. However, we think those risks are negligible because the changes in features will not be big in consecutive frames of a video with a high enough frame rate. If those risks become problematic in the future, we might consider a better feature detection and matching technique.

Result. The consecutive images with detected and matched features are shown below.



On the consecutive images, the features are detected and matched well with the

SSD method. There are more matched features, but they do not appear in the images. Only some of the matched features are shown for better visualization. If every matched point is plotted, the images will be fully covered with matching lines. Based on our result, the methods used for feature detection and matching work well on consecutive images with slightly different angles. We will adjust some variables in the methods to get a better result in the later parts.

3.2 Matrix Factorization with Missing Data

Overview. The observation matrix has an interesting property: a low rank. We could use this low-rank property to construct a rank constraint. Then, the rank constraint is used to decompose the observation matrix into two matrices: structure matrix and motion matrix. The structure matrix essentially tells you the 3D structure of the scene, and the motion matrix tells you how the camera moved during this filming process. We would use the singular value decomposition (SVD) for the first step to find the structure matrix and motion matrix.

$$[\text{ObservationMatrix}] = [U][\Sigma][V].$$

The rank of the observation matrix, also known as the rank constraint, is three. Therefore, the diagonal matrix we get in the middle of SVD, also known as the sigma matrix, could only have up to three non-zero singular values, and the rest of the singular values are zero. As a result, the remaining zeros in the singular values forces U to have only three significant columns and force V to have only three significant rows. This is the idea of rank constraints, simplifying the problem in a descent format. Besides, these non-singular values are listed in descending order, meaning that the first sigma value in the diagonal matrix is the most significant value for the 3D reconstruction. So far, this is what we covered during the Structure from Motion lecture and lab[7].

Description. Then, the next step of finding the structure matrix and motion matrix is through factorization. This approach is based on the result we get from SVD. Sigma could be broken down into two vectors. The result becomes

$$[\text{ObservationMatrix}] = [U][\Sigma^{\frac{1}{2}}][\Sigma^{\frac{1}{2}}][V]$$

This multiplication holds because Σ is a diagonal matrix and each non-zero value in the matrix could only multiply itself. Then, we could multiply by another matrix Q and Q^T to get the structure matrix and motion matrix that we want:

$$\begin{aligned}
[\text{ObservationMatrix}] &= [U][\Sigma^{\frac{1}{2}}][Q][Q^T][\Sigma^{\frac{1}{2}}][V] \\
[\text{MotionMatrix}] &= [U][\Sigma^{\frac{1}{2}}][Q] \\
[\text{StructureMatrix}] &= [Q^T][\Sigma^{\frac{1}{2}}][V].
\end{aligned}$$

Now, the question turns to how we find the appropriate matrix Q, or more specifically, [StructureMatrix] and [MotionMatrix] with the lost feature matches

during the camera motion. Buchanan and Fitzgibbon suggest an approach by using the Damped Newton algorithms[3]. We will denote this algorithm on the abbreviation of the above equation by $[O] = [M][S]$. The purpose is to find the optimized $[M]$ and $[S]$ for matrix factorization.

The idea of this algorithm is that, instead of exploiting the orthogonality of the two matrices that are vulnerable to camera rotation in the Tomasi-Kanade Factorization approach (also what we did in the lab[7]), we now use the idea of gradient descent by the Damped Newton method. Compared with the normal Newton method, extra damping allows the algorithm to select the optimal step sizes for each iteration by including the second derivative by a certain factor. Newton's method allows us to converge to the optimal $[M]$ and $[S]$ for each iteration, optimizing the motion and structure of the construction.

The algorithm is shown below:

1. Inputs include $[I]$ (the original image of a frame), $[O]$ (the observation matrix), $[W]$ the weighted matrix on error adjustment.
2. Reduce the noise of a frame by applying Gaussian filter, and then calculate the $[O_{\text{gauss}}]$
3. Apply the SVD to $[O]$, our original observation matrix, and calculate the corresponding [M] and [S] by the process we explained in [StructureMatrix] and [MotionMatrix] part.
4. Calculate the root mean squared, denoted as Frobenius norm, on $||[W] \cdot [O_gauss] - [M][S]||^2{}_F$. We will denote this as (WeightedError).
5. Give a random value to lambda1 and lambda2 and calculate $F = \lambda_1 \cdot RMS[M] + \lambda 2 \cdot RMS[S] + (WeightedError)$.
6. Then, we declare a looping lambda starting with a value of 0.01.
7. We calculate the first derivative of F denoted as d and the second derivative of F denoted as H.
8. $\lambda = \lambda * 10$, and calculate $[Y] = [O] - (H + \lambda * [IdentityMatrix])' * d$(Damped Newton's method).
9. Comparing $F([Y])$ and $F([O])$, repeat 8 until $F([Y]) < F([O])$.
10. $[O] = [Y]$.
11. $\lambda = \lambda / 10$.
12. Repeat from 7 to 11 until convergence
13. Unvectorize $[O]$ by applying SVD and output new $[M]$ and $[S]$.

14. Then, with the optimized $[M]$ and $[S]$, we could recover the motion matrix and structure matrix for 3D construction.

Possible Risks. However, there are a few drawbacks to this approach. Weighted matrix [W], arbitrary lambda1 and lambda2 are randomly chosen which could lead to an unstable number of iterations based on different given images. Secondly, the weighted matrix [W] is very difficult to give since the original author didn't give a sample matrix as an example, unlike the looping lambda of 0.01. So, this weighted matrix is hard to come up with and requires a lot of experiments. Thirdly, similar to the CNN approach exploiting the idea of gradient descent, this approach depends on the original data point that we start with. There may be a chance that it could fall into a local minimizer. Fourthly, the author didn't specify what convergence means, and it's very difficult to select a halt condition based on this vague description. Lastly, applying convolution in the first place and calculating a large number of frames by applying the root mean square means this method is computationally heavy and depends on the efficiency of the hardware.

Future Evaluation. As we mentioned above in the data section, we will evaluate the implementation factorization with Damped Newton's method by taking a step further and calling the 'triangulate' and 'plot' functions on MATLAB to reconstruct the camera position and 3D reconstruction. The result could be evaluated on a 3-scale-system: awesome, good, and reasonable. Awesome scale means both the camera position and 3D structure of the scene are clearly labeled and shown. With pair of images that are vulnerable to missing points also construct a clear 3D structure. Good scale means that both the 3D structure and camera position are clearly plotted. However, there are several outliers still being captured and not eliminated by the algorithms. Lastly, with a reasonable result, in order to continue, we need a reasonable amount of constructed feature points to represent the temple. However, there are clearly a lot of outliers, suggesting that the Damped Newton's method doesn't have a strong effect in eliminating the outliers.

3.3 Bundle Adjustment

Overview. Bundle Adjustment is another technique we would use to estimate the 3D location of a feature point and the camera location when taking an image of that scene. By applying this method, we are trying to minimize the reprojection error. Suppose that we have the coordinates of the camera and the coordinates of the feature points in the environment, we are projecting the feature points into a camera image using our assumption value. This gives us the image coordinates of that feature point. Then, we compared the estimated pixel location of this feature point with what that feature point actually is at in our image. We are trying to minimize this discrepancy by applying the least squares to all the feature point pairs.

Description. The bundle adjustment algorithms proposed by Triggs are described as follows[4]. Our inputs include the feature point location in the real world, $p_W = [xyz]'$. The camera coordinate of this feature points is denoted as: $p_{Ci} = R_c(p_W - p_C)$. p_C is the position of the camera's center projection, R_c is the rotation matrix determined by the camera's pose estimate, p_{Ci} is the feature coordinates in the camera's image. Using the parameters above, we estimate the image coordinate x and y by following equation: $p_I = [x'; y'] = (f/([001] * p_{Ci})) * [100; 010] * p_{Ci}$. p_I is the predicted image coordinates, x and y. The difference between the predicted x, y and $x'y'$ is combined as $[\delta_X; \delta_Y]$. Then, we calculate the cost elements from every frame and every feature point. This forms a cost vector $C = [\delta_{X1}, \delta_{Y1}, \delta_{X2}, \delta_{Y2} \cdots \delta_{Xq}, \delta_{Yq}]'$. q is the total number of frames. For every frame, we have p_{Ci}, R_c, and p_W. We then combine all of them into a single vector to get matrix K. Then, the camera's orientation vector is included in K.

Possible Risks. The difficulty of this approach comes from implementing the high level algorithm in MATLAB. Unlike the previous factorization algorithm, this algorithm is not described as clearly as before. Although the high level idea is clear, the author doesn't explain detailed operations like why we combine all the camera rotation, image coordinates, and world coordinates into a single vector. Also, Triggs didn't explain how we retrieve the camera orientation vector from the final K vector. Last but not least, although the idea of using least square root is clear, but the output of such an algorithm is still unclear and input such as camera intrinsics are hard to specify given a random image or video.

Future Evaluation. Bundle adjustment plays a huge role in affecting the 3D construction result from multiple view images. Therefore, we inputs 312 images in the Middlebury dataset to the function that we created. After the 3D reconstruction is made, we quantitatively evaluate the reulst by comparing with the ground truth construction posed in the Middlebury dataset website listed in the reference page. The awesome scale would require the model generated to have corresponding construction from all the view points. The second good scale allows the constructed model to miss part of the structure or has a strong outliers effect that are not eliminated by the bundle adjustment algorithms. The third reasonable scale allows the 3D models to have some extreme outliers and large percentage of missing or erroneous structure.

3.4 3D Reconstruction from a Video

Overview. For the first three milestones, we used the Middlelbury dataset to test our methods. Now, we apply the whole process to a video. Since a video is simply just a set of image frames, we think we can use the same process to a set of image frames extracted from a video with no problem.

Description. In order to extract image frames from a video, we will use

the *VideoReader* object with the *readFrame* function in MatLab[9].

Possible Risks. Since the data for our final milestone is a real-recorded video, there might be a noise in some of the image frames due to human errors such as hand tremor. We think some missing data on the noise image frame will be solved in the process of the damped Newton factorization and the bundle adjustment. However, if the noise image frame causes a huge error in our result, we would think about a method to adjust or skip noise image frames.

Future and Final Evaluation. Since for the real-life recorded video, we do not have a ground truth value to compare with, we could only quantitatively analyze the result we get from our reconstructed model. We would analyze and compare the overall structure of the classroom. Since the cameraman's route is roughly rectangle, the 3D reconstruction would also be rectangle. Also, artifacts like chairs are only recorded in the front direction since we didn't change the orientation of the camera on those artifacts while walking. Therefore these artifacts shouldn't have feature points detected at the back of them. However, other special artifacts like the corner of the front desk and the desktop at the front should have a rough 3D model since rotation of the camera captures both the back of the desktop and the side of the desktop. Last but not least, we will also analyze the noise level's effect of such a camera, considering the iphoneXs is not the camera with the highest resolution and therefore the 3D reconstructed result would be influenced by the noise.

Conclusion

Based on this project proposal, we will build a 3D reconstruction with camera motions from a video. We finished the first milestone by detecting and matching features in consecutive image frames. The features seem to be detected and matched well. For the remaining milestones, we will use the damped Newton matrix factorization method to find the structure matrix and motion matrix and adapt the bundle adjustment for a more accurate 3D reconstruction. At the end, we will apply the whole process to a video. This proposal shows a good guideline for our overall project.

References

[1] Szeliski, Richard (2022). Computer Vision: Algorithms and Applications, (Second Edition).

[2] Matthew Brown, Richard Szeliski, and Simon Winder (June 2005). Multi-image matching using multi-scale oriented patches. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005), volume I, pages 510-517.

[3] Buchanan, A. and Fitzgibbon, A. (2005). Damped Newton algorithms for matrix factorization with missing data. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 316–322.

[4] Triggs, B., P.F. McLauchlan, R.I. Hartley and A. Fitzgibbon (2000). Bundle adjustment – A modern synthesis, In Vision Algorithms: Theory and Practice, volume 1883 of Lecture Notes in Computer Science. Pages 298- 372. Springer-Verlag. https://www.asprs.org/a/publications/proceedings/baltimore09/0066.pdf

[5] Feature Detection Lab: https://weinman.cs.grinnell.edu/courses/CSC262/2022F/labs/feature-detection.html

[6] Stereo Disparity Lab: https://weinman.cs.grinnell.edu/courses/CSC262/2022F/labs/stereo-disparity.html

[7] Structure from Motion Lab: https://weinman.cs.grinnell.edu/courses/CSC262/2022F/labs/structure-from-motion.html

[8] Middlebury Dataset: https://vision.middlebury.edu/mview/data/

[9] MatLab Documentation for the VideoReader object: https://www.mathworks.com/help/matlab/ref/videoreader.html