EECS 106B Project 3 - Multi-view 3D Reconstruction

Prabhman Dhaliwal Matthew Hallac Jaiveer Singh

***Abstract*—In this paper, we consider several approaches for reconstructing a static 3D structure of points given multiple views of the structure, and with potentially incomplete information about the structure's key features. Methods tested include simple Triangulation for Two-View Reconstruction, an iterative Factorization Algorithm for Multi-View Reconstruction, a SIFT-based feature extraction scheme, and a novel contribution of a technique to identify feature correspondences from wireframes.**

***Index Terms*—Camera Model - Essential Matrix - Calibration - SVD - Triangulation - Re-projection Error - Factorization Algorithm - Image Feature Matching - SIFT - SURF - ORB-RANSAC - View Correspondences - Wireframe Matching - Reconstruction - Features - Epipolar Constraint**

I. Methods

A. 3D Reconstruction

The following paragraphs review fundamental algorithms used to simultaneously estimate camera poses and the 3D positions of various features through Two-View and Multi-View reconstruction.

1) Eight-point Algorithm: The most basic algorithm for homography determination is the Eight-point Algorithm, classically covered in many textbooks. Essentially, if we can estimate the essential matrix E, then we can recover the rotation and translation matrices that relate the point correspondences for each feature. The Eight-Point Algorithm estimates this essential matrix from a pair of images, as described by algorithm 5.1 in *An Invitation to 3-D Vision*. For the sake of completeness, a brief reproduction is presented below:

- Formulate the epipolar constraint as a system of linear equations with known homogeneous coordinates of $(\tilde{x}_1^j, \tilde{y}_1^j), (\tilde{x}_2^j, \tilde{y}_2^j)$, allowing for the isolation of E^s, a flattened representation of the essential matrix.

$$\begin{bmatrix} \tilde{x}_2^j & \tilde{y}_2^j & 1 \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1^j \\ \tilde{y}_1^j \\ 1 \end{bmatrix} = 0 \quad (1)$$

becomes:

$$\mathbf{A}\mathbf{E}^s = 0 \quad (2)$$

Note that the convention used here and subsequently is that each superscript on the x and y terms represents the feature point index, while each subscript represents the image view index. E^s is:

$$\mathbf{E}^s = [e_{11}, e_{21}, e_{31}, e_{12}, e_{22}, e_{32}, e_{13}, e_{23}, e_{33}]^T \quad (3)$$

Each row A_i of the A matrix is:

$$A_i = \begin{bmatrix} \tilde{x}_1^i\tilde{x}_2^i, \tilde{x}_1^i\tilde{y}_2^i, \tilde{x}_1^i, \tilde{y}_1^i\tilde{x}_2^i, \tilde{y}_1^i\tilde{y}_2^i & \tilde{y}_1^i & \tilde{x}_2^i & \tilde{y}_2^i & 1 \end{bmatrix} \quad (4)$$

We wish to minimize this product relative to the two-norm, such that E^s is unitary.

- Project E onto the essential space and then compute the SVD of A. We can plug in this SVD into our minimum norm equation, and write out a modified objective function and constraint to optimize over.
- Using the modified SVD linear programming problem, we can now extract the rotation and translation matrices by reshaping E^s into a 3x3 matrix, and use the given set of 4 solutions (derived in the book). These solutions are:

$$\begin{aligned} (\widehat{\mathbf{t}}, \mathbf{R}) &= \left(\mathbf{U}\mathbf{R}_z\left(\tfrac{\pi}{2}\right)\mathbf{\Sigma}\mathbf{U}^T, \mathbf{U}\mathbf{R}_z^T\left(\tfrac{\pi}{2}\right)\mathbf{V}^T\right) \\ (\widehat{\mathbf{t}}, \mathbf{R}) &= \left(\mathbf{U}\mathbf{R}_z\left(\tfrac{\pi}{2}\right)\mathbf{\Sigma}\mathbf{U}^T, \mathbf{U}\mathbf{R}_z^T\left(-\tfrac{\pi}{2}\right)\mathbf{V}^T\right) \\ (\widehat{\mathbf{t}}, \mathbf{R}) &= \left(\mathbf{U}\mathbf{R}_z\left(-\tfrac{\pi}{2}\right)\mathbf{\Sigma}\mathbf{U}^T, \mathbf{U}\mathbf{R}_z^T\left(\tfrac{\pi}{2}\right)\mathbf{V}^T\right) \\ (\widehat{\mathbf{t}}, \mathbf{R}) &= \left(\mathbf{U}\mathbf{R}_z\left(-\tfrac{\pi}{2}\right)\mathbf{\Sigma}\mathbf{U}^T, \mathbf{U}\mathbf{R}_z^T\left(-\tfrac{\pi}{2}\right)\mathbf{V}^T\right) \end{aligned} \quad (5)$$

Each $\mathbf{\Sigma}$ is a diagonal matrix whose eigenvalues are 1, 1, 0. At this point, the desired essential matrix E is successfully reconstructed.

2) Two-View Triangulation Algorithm: Triangulation is the act of estimating the 3D position of a point based on the correspondence points in our 2D images, and assuming proper calibrations. The algorithm implemented in this part considers each pair of feature points in the images and outputs the 3D point that corresponds to those points. Since this is typically a projection onto an inner space, there will be re-projection error between the identified 3D points and the original 2D points considered in each image. In a similar vein to the earlier Eight Point Algorithm, we can solve the following system of equations using the SVD:

$$\lambda \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (6)$$

In this step, the individual entries p_{ab} are part of the $P = [R|T]$ matrix, and so solving for the p_ab allows the return of the R and T matrices encompassing the homography. All other details follow from the previous elaboration, and have been discussed at length in various classical textbooks.

3) Factorization Algorithm: We have discussed a process for recovering the R, T matrices corresponding to the transformations between points in two views, and we will now expand

on a process to do the same with more than two views. The factorization algorithm is meant to incorporate information from several views to better estimate a projection matrix Π_i, where the index i refers to index of the view. This algorithm is a natural extension in concept from the eight point algorithm, but the reality of the constraints on this problem mean that a bilinear optimization approach is necessary. Note that for the sake of notational convenience, we define $\alpha_j = 1/\lambda_1^j$. The algorithm we implemented is listed as follows:

- Compute R and T for the first two views of the m total views, as well as an initial guess for the locations of the features' 3D points, using our Eight-Point Algorithm and Triangulation.
- Compute the first-pass estimate of each α_j by assuming that the first view is the identity:

$$\alpha^j = -\frac{\left(\widehat{\mathbf{x}_2^j \mathbf{t}_2}\right)^T \widehat{\mathbf{x}_2^j} \mathbf{R}_2 \mathbf{x}_1^j}{\left\|\mathbf{x}_2^j \mathbf{t}_2\right\|^2} \tag{7}$$

- Since we can only recover our depth up to a scale, we decide to let $\alpha_1 = 1$ for the initialization step, and scale all other α_j terms accordingly.
- Now, we use the initialized values as our first attempt to resolve the bilinear optimization problem for α_j and R_i, T_i. The following steps are repeated until the average re-projection error across all feature points and all images is below a specific $\epsilon = 1e-5$, a value that was calibrated experimentally.
  - First, we need to create a new estimate for each R_i and T_i using the current set of α_j:

$$\begin{bmatrix} \mathbf{x}_1^{1^T} \otimes \widehat{\mathbf{x}_i^1} & \alpha^1 \widehat{\mathbf{x}_i^1} \\ \mathbf{x}_1^{2^T} \otimes \widehat{\mathbf{x}_i^2} & \alpha^2 \widehat{\mathbf{x}_i^2} \\ \vdots & \vdots \\ \mathbf{x}_1^{nT} \otimes \widehat{\mathbf{x}_i^n} & \alpha^n \widehat{\mathbf{x}_i^n} \end{bmatrix} \begin{bmatrix} \mathbf{R}_i^s \\ \mathbf{t}_i \end{bmatrix} = 0 \in \mathbb{R}^{3n} \tag{8}$$

    As established above, a straightforward application of the SVD produces a flattened representation of R_i and T_i, though an explicit normalization step is required to ensure the matrices are properly members of $SO(3)$.
  - Now that we have our estimated projection matrix for the current view, we can recalculate a^j in the same fashion explained in the initialization step. The only modification is that we will compute the average across all m images instead of merely the 2 initial images used in Triangulation.
  - Finally, with the latest j and R_i, T_i, we can calculate a new re-projection error averaged across every image and every feature point. This error is then checked against the main loop to determine completion.
- Once we have our final 3D points calculated and with an acceptably-low error, we simply return the 3D points along with each R_i, T_i.

B. Re-projection Error of Reconstruction

Since the re-projection error is essential to the termination of the Factorization Algorithm, it is worth elaborating on the exact process by which it is calculated. Recall that the fundamental idea of using the re-projection error is to incur cost when our final 3D points do not match the original 2D projections of the same points in each image view.

With the basic idea in mind, the actual algorithm is quite simple: iterate over each image and its reconstructed R_i, T_i; further iterate over each 3D feature point with its matching 2D point that was initially provided to the algorithm; re-project the 3D point onto the image plane using R_i and T_i; compute the l_2 norm between the re-projected point and the original 2D point, and accumulate in a running total. A simple average across all images and all features produces the final result.

The equation involved in this process is as follows (where z is our true measurement, and π is our measured response):

$$r = \sum_{i=1}^{n} \sum_{j=1}^{m} \left\| z_{ij} - \pi_i \left(f_j^{(1)} \right) \right\|^2 \tag{9}$$

C. Two-view Feature Matching

The two-view feature matcher used in our experiment was a FLANN matcher provided by OpenCV. The underlying feature set was derived using the SIFT feature extractor, and outliers were subsequently removed and the fundamental matrix calculated with a RANSAC-based approach. The choice to use SIFT for feature extraction was made because of the notable property that SIFT features are scale-invariant, and are thus able to be matched regardless of the image scale or the distance from the object being photographed. Another benefit of SIFT is that the descriptors are rotation-invariant, because the underlying representation orients descriptors with gradients pointing in a common direction. SIFT is very accurate, but can be slow. However, for this experiment, we chose not to weight runtime as heavily as an on-line vision system might.

The choice to use fundamental matrix estimation over homography estimation was made experimentally. Based on a dataset consisting of several images of a toy dollhouse and a separate dataset of synthetic city blocks, we found that using the fundamental matrix returned both more feature points and more accurate feature points. However, there is a degenerate case where the homography estimation approach would make more sense. Specifically, if it is impossible to obtain well-conditioned points that are not co-planar, then a homography reconstruction approach is necessary.

In order to boost the performance of this strategy, and to reduce false positives during the SIFT feature extraction step, we incorporated the criterion of Lowe's ratio test (Lowe, 1999). Essentially, this ratio test considers the relative strength of the first- and second-best matches to each descriptor, and ensures that only descriptors with a sufficiently low ratio are accepted. The intuition behind this approach is that a descriptor that is so vague as to match several candidates will have a ratio close to 1 between its first- and second-

best matches, and so should be rejected. We chose to set the threshold parameter of the ratio test to 0.7.

The parameters we had to set for RANSAC were the RANSAC re-projection threshold and the confidence level. The RANSAC re-projection threshold was chosen to be 2.0. This value represents the threshold for error when running RANSAC and what the function will define as an outlier versus an inlier. Increasing this value increases the number of points returned, but also increases the amount of error that will be allowed, and so we have maintained a very conservative value. Confidence was set to 0.995, requiring an extremely high certainty that the final matched point is valid, consistent with our desire for an error-free output. Of course, the natural consequence of these decisions is that we may be so unnaturally restrictive that the returned features are few in number, and thus that the re-projection performs poorly anyways. However, our experimentation has validated the use of these parameters across the two datasets considered.

D. Feature Tracks and Multi-view Correspondences

For tracking features across multi-view correspondences, we followed the below elementary algorithm:

1) Use SIFT to extract the features from the first image
2) Initialize a Feature Track for each point extracted, with each Feature Track's first image to be image 1 and its first keypoint to be the detected point.
3) For each image i from image 2 to the end:
   a) Extract the features from image i
   b) Use the two-view feature matcher to match features from image $i-1$ to the features of image i
   c) For each match:
      i) Check if there exists a Feature Track with th last image equal to $i-1$ and a last keypoir equal to the matched keypoint for the $i-1$. I so, add the match to the Feature Track.
4) Check each resulting Feature Track to see if it properl spans across every view considered. If this is the case add each keypoint in the Feature Track to the final lis of points.
5) Return the final composed list of points.

E. Wireframe Correspondences

The pseudocode for our wireframe method is outlined in thi section. Given 2 images, a junction map, the point identifiers and a line map:

- Take each junction and compute a descriptor using th SIFT feature extractor
- Match descriptors using FLANN-based K-Nearest Neigh bors
- Use Lowe's ratio test to reduce the number of points and eliminate bad correspondences.
- For a fixed number of iterations:
  1) Pick two correspondences randomly.
  2) Check if there is a line between each pair in each of the images.
  3) If there is a line in one image and not in another, increment a counter attached to each of the correspondences, indicating that there is a disparity.
- Remove correspondences which have a disparity counter value above a certain threshold, preserving the others.
- Return the final set of junctions as the keypoints to be used in the reconstruction algorithm.

In choosing two correspondences randomly and testing for disparities, we emulate the behavior of Random Sample Consensus (RANSAC). This ensures that, if there is a line between two points, but only one of them is wrong, only that wrong point is removed. As a consequence of our 'overly-choosy' approach, this method would not work well in sparse wireframes, but can accurately and precisely remove false matches in densely connected wireframes.

II. Experimental Results

A. 3.1: Two-view 3D Reconstruction using Given Matched Points

Figures 1, 2, and 3 show the 3D reconstructed points from our two-view algorithm on a synthetic skyscraper dataset, presented across several perspectives. Figures 4 and 5 show how our estimated values for the re-projected points match up to the actual 2D points for each view. The average re-projection error was on the order of $1e-7$, as expected given the synthetic data used in this experiment. Experimentation here involved identifying a low enough threshold that produces virtually perfect results without requiring more than a minimal runtime, though preference was given to accuracy over speed.

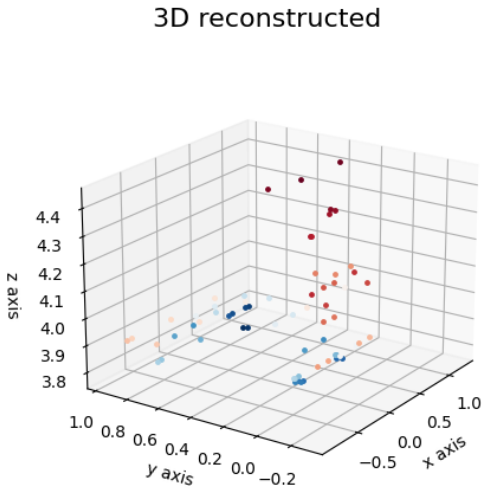

Fig. 1. 3.1: View 1 of 3D point reconstruction

B. Multi-view 3D Reconstruction using Given Matched Points

The generated 3D points using 4 input views of the skyscraper dataset are visualized in 3D in figures 6, 7 and 8.

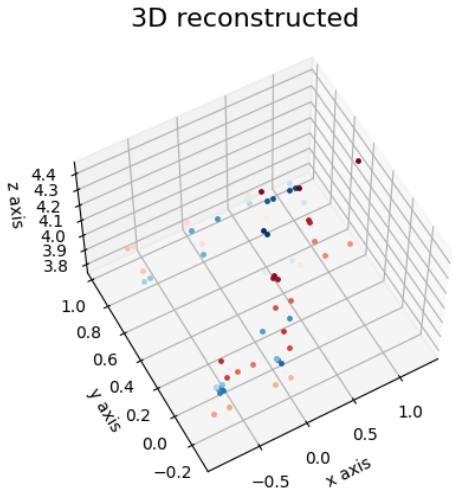

Fig. 2. 3.1: View 2 of 3D point reconstruction

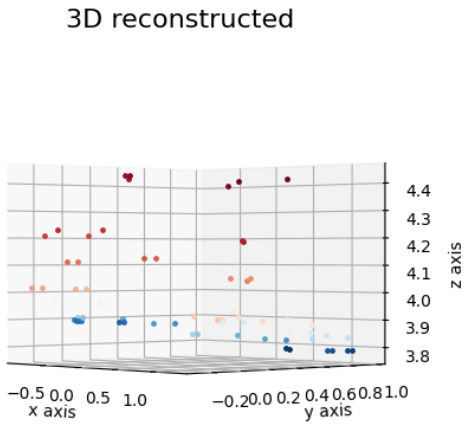

Fig. 3. 3.1: View 3 of 3D point reconstruction

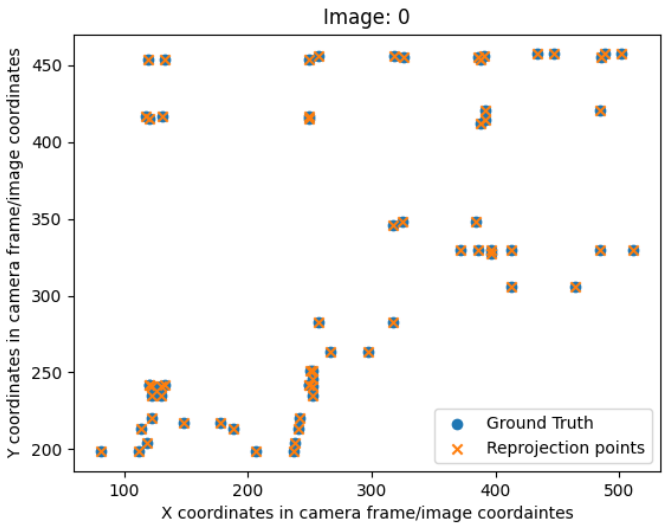

Fig. 4. 3.1: Estimated points vs true points

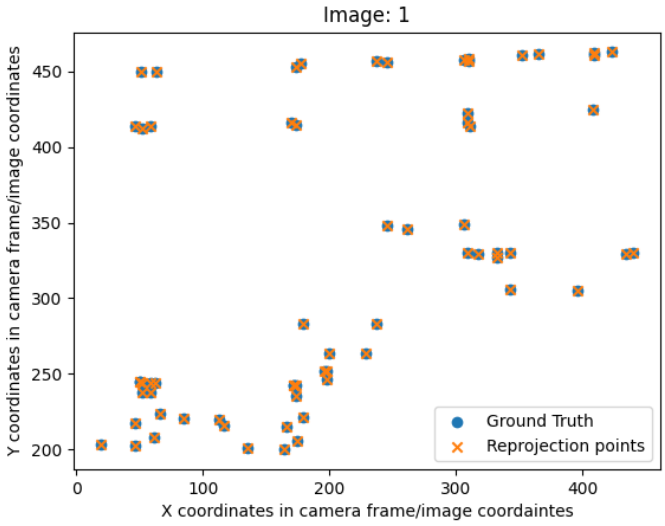

Fig. 5. 3.1: Estimated points vs true points

The individual images of re-projected points overlapped with the ground truth are given in 9, 10, 11 and 12.

C. Multi-view Reconstruction using Detected Features and Keypoints

Multi-view reconstruction using SIFT-generated features on the toy dollhouse dataset produced 3D points shown in Figures 13, 14, and 15. For reference, the matching images taken before and after RANSAC's elimination are provided in Figures 16 and 17 respectively.

D. Wireframe Matching

A selection of visualizations of our novel wireframe-handling algorithm are provided in 18 and 19.

III. Discussion

A. Feature Extraction Accuracy

In our experimentation, we found that SIFT finds relatively noisy features. Though the features appear quite sophisticated to the human eye, the lack of pixel accuracy means that the reconstruction is very vulnerable to slight deviations caused by noise. The features are generally well distributed around the object of interest, though our combination of parameters means that many points not directly in the focus of each view are discarded. As a consequence, SIFT could be most useful in situations where the images are clearly capturing the target object, and in which roughly the same portion of the object is captured in each consecutive frame. Scenes in which the target

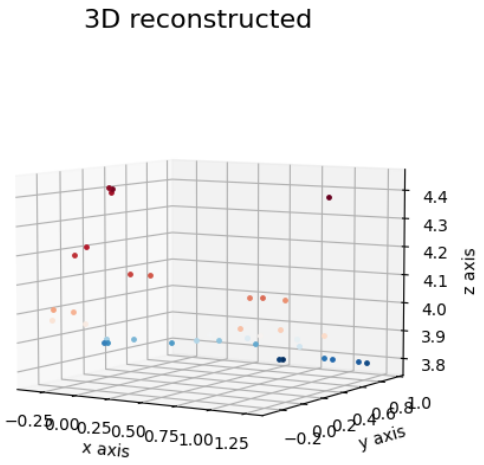

Fig. 6. 3.2: View 1 of 3D point reconstruction

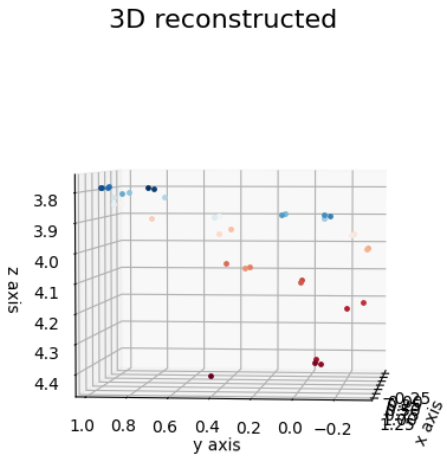

Fig. 8. 3.2: View 3 of 3D point reconstruction

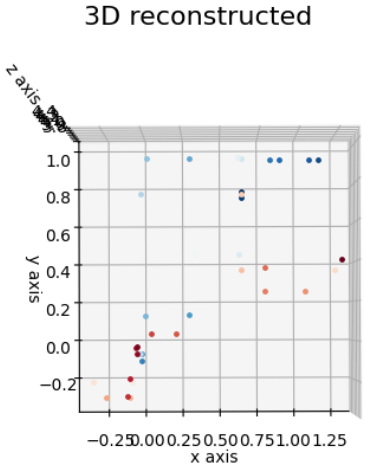

Fig. 7. 3.2: View 2 of 3D point reconstruction

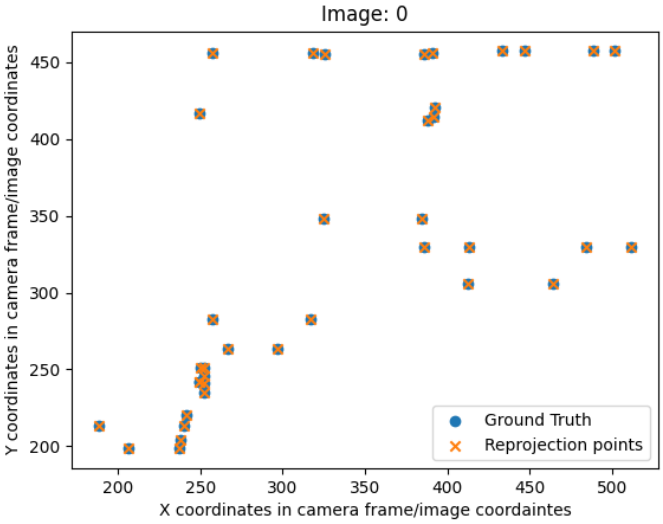

Fig. 9. 3.2: Estimated points vs true points

structure rotates through an angle larger than 180 degrees, for example, will necessarily cause SIFT to fail, since no points from the backside of the object will be matched to the front. SIFT also fails to account for higher level structure, which methods such as wireframe matching utilize.

B. Feature Matching and RANSAC Accuracy

After full consideration, utilizing the combination of RANSAC and the Lowe's ratio criterion enabled us to effectively eliminate false positives and thus obtain valid 3D points. We found the best ratio to be 0.7 between the first nearest neighbor and the next. A comparison of the effect of running RANSAC and Lowe's on the features can be seen between the before (Figure 17) and after (16). Clearly shown are the removals of many extraneous features. The failure cases exhibited a tendency to cluster around areas rife with noise in the underlying image itself. For example, examining the ground near the house in the referenced figures shows that our techniques struggled to remove from these 'clusters' near where the black and grey strips meet, especially near the red and blue patches on the ground. The relative change in intensity is narrow in those locations, but the patterns are broadly unique across the image. As a result, the features are unambiguous but with considerable variation between where the actual feature is determined inside a small window. Numerically, there were approximately 50 matches in Two-View reconstruction, with only about 20 preserved across Multi-View reconstruction.

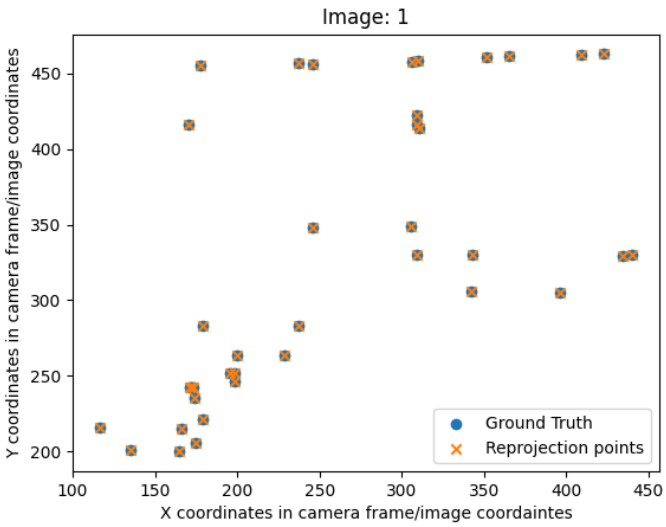

Fig. 10. 3.2: Estimated points vs true points

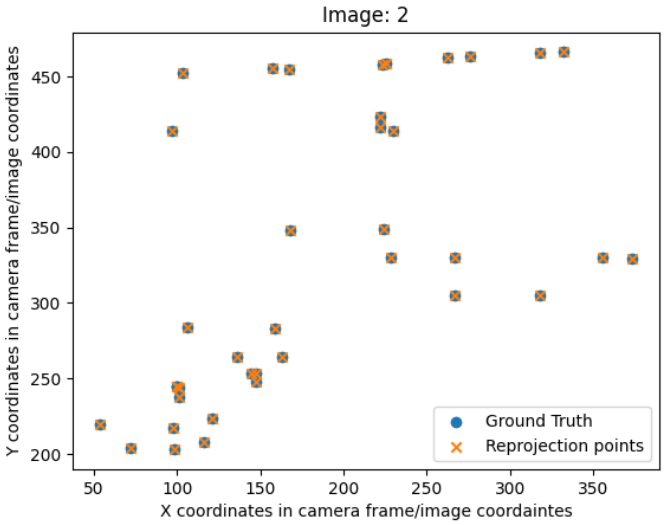

Fig. 11. 3.2: Estimated points vs true points

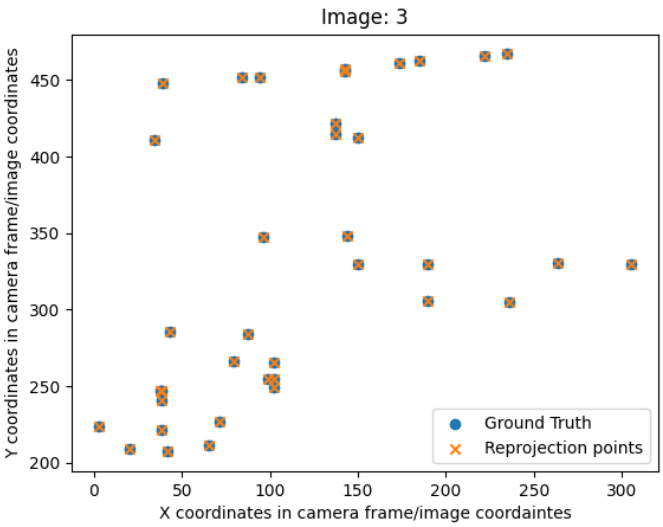

Fig. 12. 3.2: Estimated points vs true points

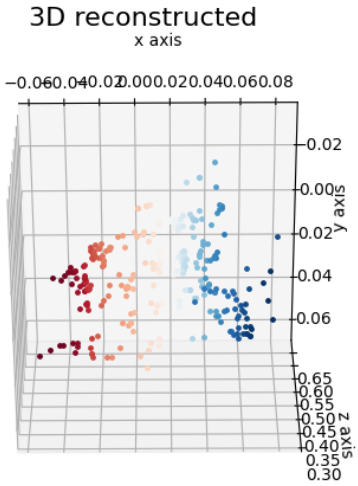

Fig. 13. 3.3: 3D reconstruction view 1

C. *Quality of Reconstruction*

The reconstruction algorithm for Two-View reconstruction had a remarkably good reproduction of the 3D points from the 2D images. Our average error was near 0, on the order of $1e-7$, and from the plots it is evident that the orange re-projected points are virtually identical to the given blue points. The Factorization Algorithm for Multi-View reconstruction achieved a similarly impressive result, with average error on the order of $1e-6$. Again, the points are virtually exact matches in each of the 4 camera views considered.

D. *Iteration in the Factorization Algorithm*

An essential component of the Factorization Algorithm is that it iteratively improves the estimates for both the depths of 3D points and the underlying homographies of each camera view in a bilinear fashion. This iteration is critical, because the initialized values for 3D points only considered information from the first two views provided. As a result, a progressively improved set of R_i and T_i could then be re-deployed to improve the depth-analogues α_j, rapidly reducing total re-projection error in the synthetic data case. However, this iteration does come with a vulnerability when utilizing imperfect feature matches, such as those from SIFT. Because there is some inherent uncertainty in the exactness of the matches, repeated iteration is not necessarily guaranteed to converge to a low-error final result; instead, the values tended to oscillate between two points on a local minimum. To compensate for this, we implemented a maximum iteration count as a fallback

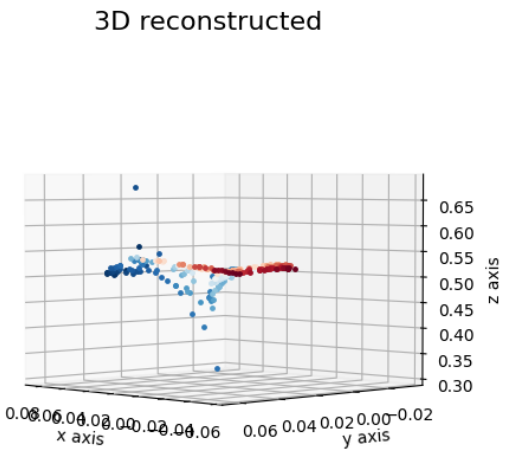

Fig. 14. 3.3: 3D reconstruction view 2

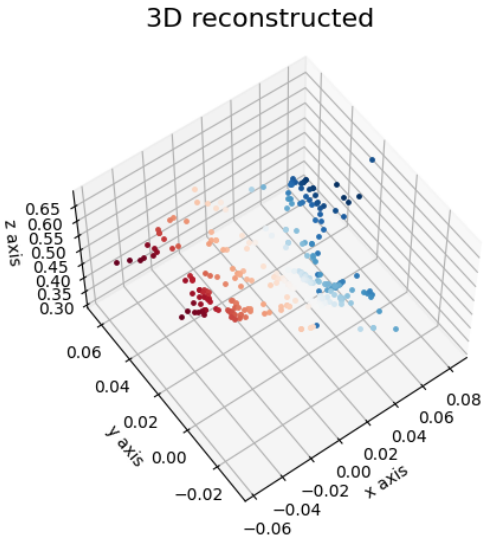

Fig. 15. 3.3: 3D reconstruction view 3

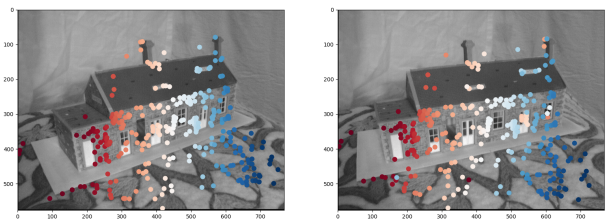

Fig. 16. 3.3: Matching Before RANSAC

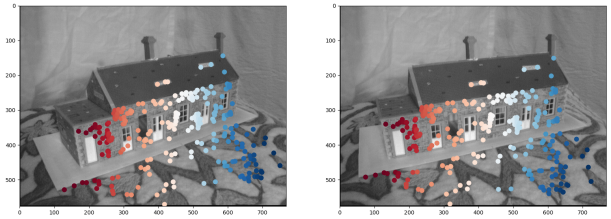

Fig. 17. 3.3: Matching After RANSAC

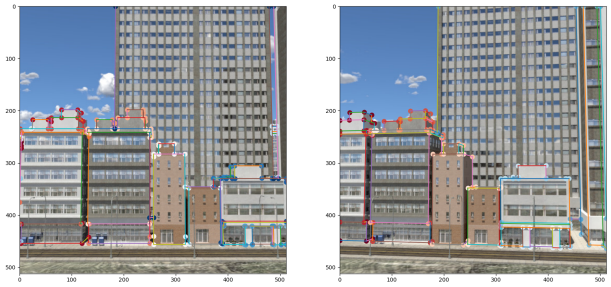

Fig. 18. 3.4: Wireframe Lines through Novel Method

to avoid infinite looping, which effectively eliminated the problem to the largest degree possible.

E. SIFT vs. Ground Truth Reconstruction

Unsurprisingly, the reconstruction accuracy was significantly stronger using the ground truth features than with SIFT's reconstructions. Though the comparison is not quite fair due to SIFT's better performance on 'real-looking' images like the toy house, coupled with the fact that the synthetic data was limited to the skyscraper dataset, we are convinced

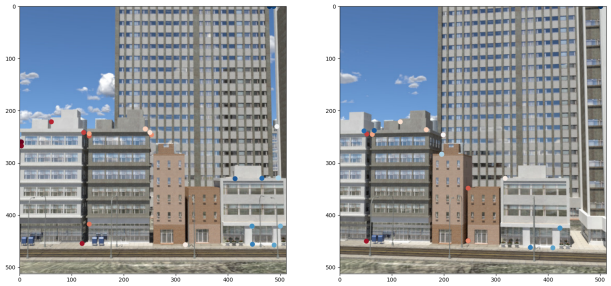

Fig. 19. 3.4: Wireframe Points through Novel Method

in our conclusion that the methods expressed in this paper work best when the underlying features can be treated as exact. Intuitively, some sort of regularization term might be necessary to adapt these techniques to work well with imperfect features.

F. SIFT vs. Learning-based Feature Extractors

Our testing revealed that the significant downside of SIFT is its failure against repeated patterns in an image. Particularly evident in this case was that SIFT often mistook windows for adjacent, or even quite distance, windows of the same skyscraper on the synthetic dataset. By contrast, the learning-based approach accurately provided excellent features on a building-specific level, and should largely be preferred if made available. Of course, the natural constraint is that a learning-based approach requires a sophisticated model capable of understanding the underlying image shown in the views. For complex or irregular shapes like the dollhouse, such a method might struggle and thus perform quite poorly.

G. Wireframe Correspondence Challenges

When implementing the wireframe correspondences, a number of challenges were encountered. First of all, when matching descriptors, we found that Lowe's ratio test was returning an insufficient number of kept points. Further exploration revealed that this was due to the sparsity of points in the initial wireframe. This effect was worsened by the fact that the back corners of the buildings were obscured in straight-on views, effectively halving our available points. Next, checking if each pair of correspondences were connected in each image took an absurd amount of runtime, and was not effective enough to justify the time cost. One method we devised to account for this was to employ a technique similar to RANSAC. In this method, we pick two random correspondences, and check for a line between them in both views. If there is an inconsistency, both of the pairs are marked with an inconsistency counter, and outliers are removed by consensus. This method has the advantage of being able to set a certain number of iterations. Ultimately, however, this line-matching approach still struggled from the underlying point sparsity, and so it was discarded in favor of the purely point-based matching scheme defined above.

IV. Bibliography

References

[1] Ma, Yi, et al. An Invitation to 3-D Vision: from Images to Geometric Models. Springer, 2006.

[2] Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints." International Journal of Computer Vision, vol. 60, no. 2, 2004, pp. 91–110., doi:10.1023/b:visi.0000029664.99615.94.

V. Appendix

A. Future Improvements

Though the several typos indicated on Piazza were appropriately remedied, the lack of consistent notation in the project specification produced a significant amount of confusion. An example of this was in the case of depth factor calculation, where an ambiguity in the explanation for the nested-loop structure suggested a functionally-impossible approach for refining estimates.

B. Code Repository

GitHub Link

C. Bonus

The learning goals for this assignment were to learn about feature extraction and analysis in computer vision using foundational algorithms and feature extraction methods. This assignment was effective in that end, showing us how these algorithms are used and how they extract meaningful features out of images, and how these extracted features can be used for analysis. Having the opportunity to write out the algorithms covered in our textbook was particularly effective as a learning exercise.