Assessing Image Importance for 3D Reconstruction

Michael Hu, Julian Knodt, Justin Yan
Princeton University
35 Olden St, Princeton, NJ 08540
{myhu, jknodt, justiny}@princeton.edu

Abstract

Multi-image 3D reconstruction algorithms tend to become more accurate with additional image inputs. However, we do not understand which images have the largest impact reconstruction quality. Here, we propose a black-box framework to analyze the contribution of images towards a final 3D reconstruction. Using our framework, we demonstrate that the first few images of a reconstruction heavily impact reconstruction quality. Thus, in the low data regime, picking good initial images is of critical importance to obtaining a good 3D reconstruction. To this end, we train a classifier to help us determine where to take the next photo, given a representation of the current camera locations and reconstruction point cloud.

1. Introduction

Structure from Motion (SfM) algorithms generate 3D representations of objects from sets of images. Classical SfM pipelines follow three general steps for 3D reconstruction. First, locate keypoints in the images. This is often done through algorithms like SIFT [11], and is not considered part of the SfM algorithm itself. Next, match keypoints across images with each other, stitching images together in a way that will allow for 3D reconstruction. Last, from the matched keypoints, derive some representation of a 3D model–generally a point cloud.

One such SfM pipeline is Bundler [16], one of the original tools for SfM. In the Bundler paper, Snavely et. al. use Notre Dame as the reconstruction target, sampling images from the internet to reconstruct a visualization of the cathedral. For lesser-known monuments and more common objects, the set of available images is much less extensive. For smaller data sets, a complete and accurate reconstruction is not always possible. Thus, the question arises, how can more images be added to an existing data set to produce a better reconstruction? How will these images impact the reconstruction quality, and which images should we add in the first place?

To understand how individual image contributions affect quality of reconstruction, we define the *partial reconstruction loss*, a metric for how close a given reconstruction is to the ground-truth 3D representation. We then apply our definition of partial reconstruction loss to Bundler outputs, treating Bundler as a black box. Through our analysis, we observe that the initial images of a reconstruction heavily affect the partial reconstruction loss. We use this result to motivate our creation of a neural network classifier that predicts where the next photo in the reconstruction should come from. Our classifier predicts the approximate location of the image minimizing partial reconstruction loss with 40% accuracy.

2. Background

2.1. Preliminaries: Earth-Mover's Distance

Earth Mover's Distance (EMD) is the primary metric we use to compare reconstruction point clouds. EMD was applied early on to content-based image retrieval (CBIR) [1,13]. EMD measures the amount of work needed to transform one discrete distribution into another, where *distribution* here is defined as an assignment of mass or value to a set of points. The problem setting is analogous to filling a set of holes in the ground with dirt from a set of piles and minimizing the distance that each unit of dirt travels.

More formally, define a discrete distribution $\mathbf{X} = (\mathcal{X}, \mathbf{w}) \in \mathbf{D}^{m,k}$ to be a set of m tuples $(\mathbf{x}_i, w_i), 1 \leq i \leq m$ such that $x_i \in \mathbb{R}^k$ and $w_i \geq 0$. Let $\mathbf{X} = (\mathcal{X}, \mathbf{w}) \in \mathbf{D}^{k,m}$ and $\mathbf{Y} = (\mathcal{Y}, \mathbf{u}) \in \mathbf{D}^{k,n}$. A flow $\mathbf{F}$ is an $m \times n$ real matrix where $f_{i,j}$ is the amount of weight w_i associated with the vector $\mathbf{y}_j$.

A *valid* flow $\mathbf{F}$ must satisfy the following conditions:

$$f_{i,j} \geq 0 \tag{1}$$

$$\sum_{i=1}^{m} f_{i,j} \leq w_i \tag{2}$$

$$\sum_{j=1}^{n} f_{i,j} \leq u_i \tag{3}$$

$$\sum_{j=1}^{n}\sum_{i=1}^{m} f_{i,j} = \min\{\sum_{i=1}^{m} w_i, \sum_{j=1}^{n} u_j\}. \tag{4}$$

The work associated with $\mathbf{F}$ is

$$W(\mathbf{F}, \mathcal{X}, \mathcal{Y}) = \sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j} d_{i,j},$$

where $d_{i,j}$ is the distance between $\mathbf{x}_i$ and $\mathbf{y}_j$; we use the Euclidean metric, so $d_{i,j} = \|\mathbf{x}_i - \mathbf{y}_i\|_2$. The objective function we must solve to yield the EMD is

$$\min_{\mathbf{F}\in\mathcal{F}(\mathcal{X},\mathcal{Y})} \frac{W(\mathbf{F}, \mathcal{X}, \mathcal{Y})}{\min\{\sum_{i=1}^{m} w_i, \sum_{j=1}^{n} u_j\}}.$$

Minimizing $W(\mathbf{F}, \mathcal{X}, \mathcal{Y})$ is known as the *transportation problem*, and is $O(n^3 \log n)$ in its classical formulation as a min-cost network flow problem on a bipartite graph. However, a number of algorithms exist for approximating the solution in linear time. [5, 9, 15]

In our case, $\mathcal{X}$ and $\mathcal{Y}$ are reconstructed point clouds and $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^3$. We face a constrained version of EMD in which $w_i = u_j = 1, f_{i,j} \in \{0, 1\}, \forall i, j$. This reduces EMD to the *assignment* problem, in which we seek an optimal one-to-one matching between $\mathcal{X}$ and $\mathcal{Y}$. [1] To balance the number of points in the sets so that $m = n$, we pad the smaller set with the average point vector until the two sets are of equal size.

2.2. Related Work

2.2.1 Traditional 3D Reconstruction

Traditional Structure from Motion algorithms are based on non-linear least square optimizations of the distances between where the estimated 3D point would project onto the image plane and the actual point location in the image. This more general form of 3D reconstruction is known as bundle adjustment in reference to the "bundles of light" that are reflected from the expected location into the camera lens and the adjustment of those bundles. Bundler, in specific, seeks to iterate on `sba`, a sparse bundle adjustment algorithm in C/C++ [10]. `sba` and others like it focus on memory efficiency and and the specific nature of sparsity in the problem in order to make the problem tractable. The problem becomes increasingly intractable as the number of pictures as points are only visible in a small fraction of cameras, but the structure of the sparsity can be used to develop efficient algorithms.

2.2.2 Structure from Motion Optimizations

Due to the time-consuming nature of traditional SfM methods like Bundler, several optimizations to SfM algorithms have been proposed. [16]. Modern iterative methods have succeeded in reducing computation time while preserving reconstruction quality [17, 18]. Qu et al. [12] use PCA to estimate the overlap area between images, and select for reconstruction images that fall within a certain overlap range. Their method maintains queues of such images and applies depth-map fusion to create dense point clouds, while reducing reconstruction time by a factor of four. Yang et al. [19] estimate the camera-calibration contribution of each image with an engineered measure function. They then prune images whose contribution is below some threshold and whose removal does not significantly decrease the contribution of other images. Schönberger and Frahm [14] proposed an incremental SfM pipeline (COLMAP) with a geometric verification strategy that improves initialization and triangulation and an uncertainty-driven approach proposed by Haner et al. [3] for next-best scene selection. Their general-purpose SfM system is 50 times faster than Bundler and suffers only $1/2$ to $1/4$ the average reprojection error that Bundler does. VisualSFM [17] is a modern SfM pipeline that features slightly faster reconstruction times than COLMAP and on-par average pixel reproduction error.

2.2.3 Single Image 3D Reconstruction

Methods for 3D reconstruction from a single image have recently surged in popularity. These methods usually simplify the reconstruction problem by restricting the object to a certain class. Rather than reconstructing a 3D representation from existing data only, single-image 3D reconstruction tends to infer the true 3D structure from the image. Fan et. al (2017) train an encoder-decoder architecture to predict 3D point structure from a single image [2]. Henderson and Ferrari (2019) recently defined another encoder-decoder architecture to predict 3D reconstruction, with the output type being a mesh instead of a point cloud [4].

3. Approach

Our approach: we define the partial reconstruction loss and characterize its behavior on Bundler outfiles as a function of k, the number of images. Through our analysis of this loss behavior, we motivate the use of our neural network classifier, which predicts where we should take the next image to minimize partial reconstruction loss.

3.1. Partial Reconstruction Loss

We investigate the effect that camera placement and the number of cameras has on reconstruction quality by considering *partial reconstructions*. A reconstruction is the point-cloud that results from running Bundler [16] on a particular image sequence. Each reconstruction is uniquely defined by the sequence of images used, and *not* by the underlying set of images. Order matters for Bundler reconstructions: the reconstruction resulting from sequence $1 \rightarrow 2 \rightarrow 3$ is in general not the same as the reconstruction resulting from sequence $1 \rightarrow 3 \rightarrow 2$.

We now define the *partial reconstruction loss*. Let S be a sequence of scene images. A partial reconstruction R_k of size k is a reconstruction that results from running Bundler on the first k images in sequence S. To compute EMD between R_k and G, we pad the smaller point set with its mean. Our intuition here is that if Bundler does not yet predict a point, then the mean is an unbiased estimator for that point. Let R'_k and G' be the padded point sets. We define $EMD(R'_k, G')$ as the *partial reconstruction loss*.

$$\begin{aligned}\text{Partial reconstruction loss} &:= L(R_k, G) \\ &= EMD(R'_k, G')\end{aligned}$$

If we define G to be the ground truth point set, the partial reconstruction loss $L(R_k, G)$ is a measure of how far away our current reconstruction R_k is from the true point set.

Due to computational complexity of calculating the EMD and limited compute time, we take G itself to be a partial reconstruction of size 90, and investigate the behavior of the partial reconstruction loss for G defined over different 90-image sequences. We chose to start with partial reconstructions with 10 images to minimize the chance of no matching points (and hence, a null reconstruction).

3.2. Calculating EMD

We wrote our own implementation of EMD. Calculating EMD between point clouds consists of 1) calculating a cost matrix of pairwise point distances and 2) solving a linear assignment problem (LAP). 1) is $O(n^2)$ in the number of points in the larger point cloud, and 2) is $O(n^3)$. Instead of using standard libraries, we attempt our own implementation of 1) using simple parallelization. For 2), we use a Python implementation[1] of the Jonker-Volgenant algorithm [6] shown to be faster[2] than several other methods, including `scipy.optimize.linear_sum_assignment`. We find that LAP is still too computationally expensive for distance matrices of larger than 20000×20000; since the provided `notredame.out` file in the NotreDame dataset [16] consists of over 127,000 points, we perform our EMD calculations against a partial 90-photo reconstruction one order of magnitude smaller (~ 16000 points) than the true full reconstruction. Such a calculation takes anywhere from 5 seconds to 30 minutes.

[1] https://github.com/src-d/lapjv

[2] https://github.com/cheind/py-lapsolver

3.3. Data Preprocessing

Our initial goal was to predict the location of the image that will minimize partial reconstruction loss. We now simplify this task into a classification problem. Instead, we train our network to predict a discretized camera angle with respect to the center of the point cloud. Concretely, let R_k be the current reconstruction point cloud, and let R_{k+1} be the reconstruction after adding one additional image c. We train our model to predict the camera angle corresponding to c^*, where $c^* := \arg\min_c L(R_{k+1}, G)$.

We performed several data preprocessing steps to transform the output of Bundler into data for our neural network. Bundler predicts camera locations $\in R^3$ and a point cloud $\in R^3$ representing the central object. Both the number of cameras and the number of points in the point cloud can vary. Thus, we require a way to summarize the data, such that the size of the input information to the neural net is the same.

To compute the inputs x_i to our neural network, we first derived a representation of the point cloud by taking the eigenvectors of the point cloud after performing PCA. This summarizes the point cloud of indeterminate size into three vectors $\in R^3$, or 9 values total. (Step 1.) Next, we radially discretized the camera locations into sectors of 22.5° in a circle around the generated point cloud. We take the mean of the point cloud to be the origin, and compute the relative angle of the camera to the origin using its X and Y coordinates. We then place the camera into one of 16 unique buckets $b_1, ..., b_{16}$. We define the final camera location vector to be the bit vector $x_i^{(2)} := [\mathbb{1}_{\{b_1>0\}} ... \mathbb{1}_{\{b_{16}>0\}}]$. Essentially, $x_i^{(2)}$ expresses whether each of the 16 sectors is covered by a camera. (Step 2.)

We now have our inputs x_i to the neural net. Concretely, our post-processed data contains tuples (x_i, y_i). $x_i := [x_i^{(1)} x_i^{(2)}]$, where $x_i^{(1)} \in R^9$ and $x_i^{(2)} \in R^{16}$. $x_i^{(1)}$ contains the three eigenvectors in R^3 derived from PCA, and $x_i^{(2)}$ is defined as above. $x_i \in X \subset R^{25}$.

Finally, we compute the y_i's to correspond to each x_i. For each point cloud R_k, we compute $L(R_{k+1}, G)$ for all remaining cameras not yet included in the sequence. We take $c^* := \arg\min_c L(R_{k+1}, G)$ and compute c^*'s camera bucket. y_i is then a one-hot vector with a one in the index corresponding to c^*'s camera bucket. $y_i \in Y \subset R^{16}$.

We computed our training set from image sets of at most 12 images. Thus, we expect the camera locations to be relatively sparse, more significantly that they are not significantly close to each and to be taken from all angles around

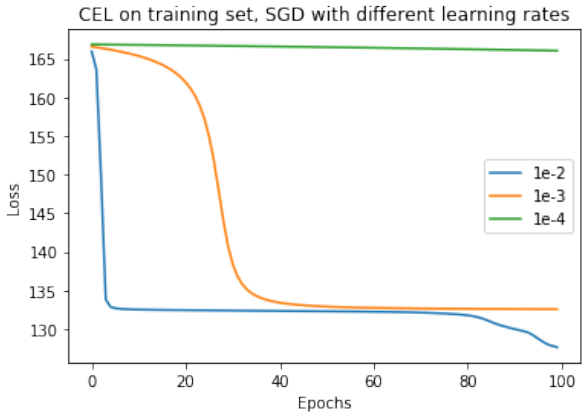
the object, allowing the camera location vector $x_i^{(2)}$ to not be all ones. We assume the eigenvectors from PCA capture the general shape of the point cloud.

3.4. Model Architecture and Training Approach

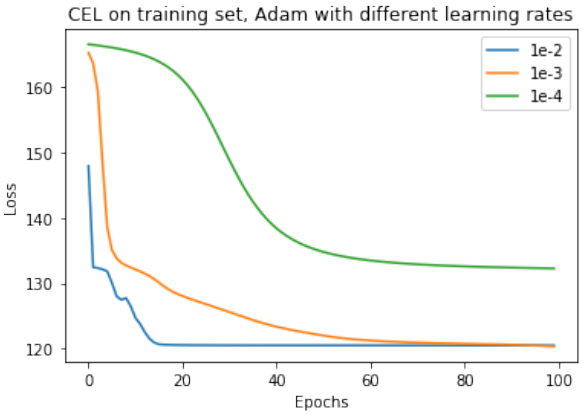
Our model is a neural network classifier learning $f : X \rightarrow Y$. The neural network has a structure of (25, 20, 16), where the fully-connected hidden layer has 20 nodes. We use ReLU activations and pass final model output values through a softmax to obtain a probability distribution on the next bucket. Last, we take the cross entropy loss between the predicted distribution and the ground truth label.

We chose to use a single hidden layer because our current architecture can approximate any function with one hidden layer [8]. Thus, adding additional layers is theoretically unnecessary. To train the neural network, we tried two different optimizers with different learning rates: SGD with momentum and Adam. By examining the loss plots, we observed that Adam better minimizes loss on the training set, without sacrificing performance on a held-out validation set (see Figures 1, 2 on page 5).

In the end, we trained our classifier for 30 epochs using Adam with a learning rate of 0.001. In terms of training and testing data, we collected 73 data points from 4 sets of images taken on 4 objects. We split the 73 data points into a 60-point training set and a 13-point validation set. Next, we collected 40 data points from a 5th set of images to serve as a testing set.



(a) SGD with different learning rates.



(b) Adam with different learning rates.

Figure 1: Loss plots for SGD with momentum and Adam

4. Results

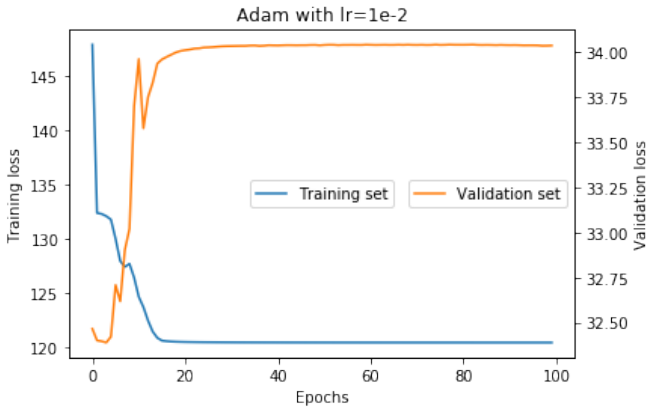
4.1. Analyzing Partial Reconstruction Loss

Our main result is that the choice of initial cameras has a longstanding impact, usually determining the behavior of partial reconstruction loss for many cameras into the future. Figure 3 shows the EMD behavior of partial reconstructions as a function of the number of images used. R4 lacks an initial 10-image reconstruction due to a failure to match points, which occasionally occurs due to the diversity of dataset image perspectives. Since the EMD does not decrease monotonically, we see that the addition of certain images decreases the reconstruction quality. For instance, we see an over two-fold spike in EMD for R1 when adding 10 images to the 70-image partial reconstruction. This motivates the need for dataset pruning strategies that remove images detrimental to camera calibration and post estimation. We would then hope to see close to monotonically-decreasing loss with increasing number of images.

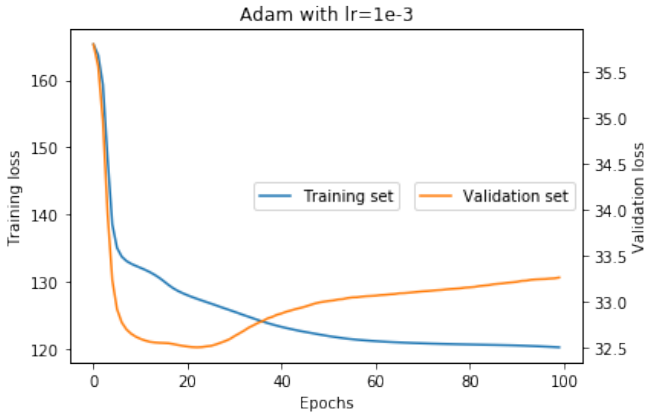
Perhaps most importantly, we note that loss for each reconstruction permutation is relatively constant for the first 40 to 60 camera additions, after which it converges to zero. This implies that the choice of initial cameras more or less determines the loss behavior until we reach a critical number of cameras, at which point the loss converges. Note here that we are using the same set of cameras across all reconstructions R1-4, and simply changing the order. Depending on the initialization set, we could suffer a loss of 14 for the first 50 camera additions (R4) or a loss of 3 (R1, R3).

Thus, choosing the right cameras is critically important to a good partial reconstruction. This motivates our construction of the neural network classifier. Given a point cloud and camera orientations, we wish to decide which camera to include in the reconstruction, such that the partial reconstruction loss will decrease. With a perfect classifier, we could avoid cases where we sample a bad initial set of cameras, leading to high loss.

With regards to other implementation details, we find that our parallelized implementation of SciPy's `cdist` function offers better performance than the library implementation, as shown in Figure 5. All exhibit quadratic runtime, as expected, but our float32 method has the smallest constant factor. We opted to use our float32 method over



(a) Adam with learning rate 0.01.



(b) Adam with learning rate 0.001.

Figure 2: Adam with learning rate 1e-3 obtains a smaller loss on the validation set. Thus, we chose to use Adam with 1e-3 as our final optimizer, stopping training at 30 epochs to help with generalization.

both other ones due to superior memory and timing performance.

Figure 6 shows the computation time needed to generate R1, R2, R3, and R4. R2 shows the most irregular behavior; the large spike in runtime just past the addition of the 60th image is unobserved in any other reconstruction. However, there is no observable correlation between this anomaly and the EMD at the same point in reconstruction.

4.2. Classifier results

Our model achieves 40% accuracy on the testing set of 40 data points. For comparison, since there are 16 possible camera sectors, we would expect a completely random classifier to be correct around 6.25% of of the time. Thus, the classifier performs better than random chance, but more work is required to increase the accuracy of the classifier

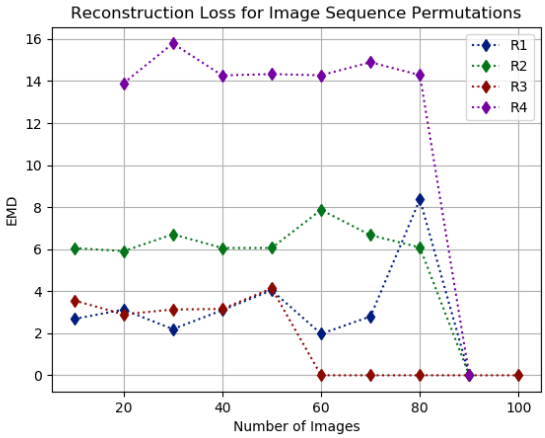


Figure 3: EMD for four permutations of the Notre Dame input image sequence. Note that we do not see the loss decrease monotonically; rather, the first few images seem to largely determine the behavior of the reconstruction up until nearly all the images have been processed.

such that it is usable in practice. We suspect that the model requires more training data, as we only trained the model on 60 samples.

A bottleneck here is that we currently preprocess and produce all data for the model ourselves. We take by hand around 15 pictures of an object to compose an image set, and we preprocess this image set into around 20 or data points. To our knowledge, there does not exist a large-scale dataset for our use case.

An accurate prediction made by our classifier is shown in Figure 7. For accurate predictions, our classifier tends to recommend taking the next picture at an angle not represented in the reconstruction.

5. Discussion

By plotting and analyzing the reconstruction loss from the Notre Dame data set, we can see that our initial assumption that certain images contribute more to the final reconstruction is validated, in that the loss did not dramatically go down until nearly all images were included in the final reconstruction. Interestingly, we also found that the contribution of more images did not always lead to better reconstructions by our EMD metric, as the EMD did not monotonically decrease with respect to the number of images. In order to understand why it might be the case that the reconstruction is getting worse, under the assumption that EMD is a good metric for comparing reconstruction quality, it is necessary to look at the Bundler algorithm to see where discrepancies may occur. Bundler uses each image and keypoints within that image as generated by SIFT

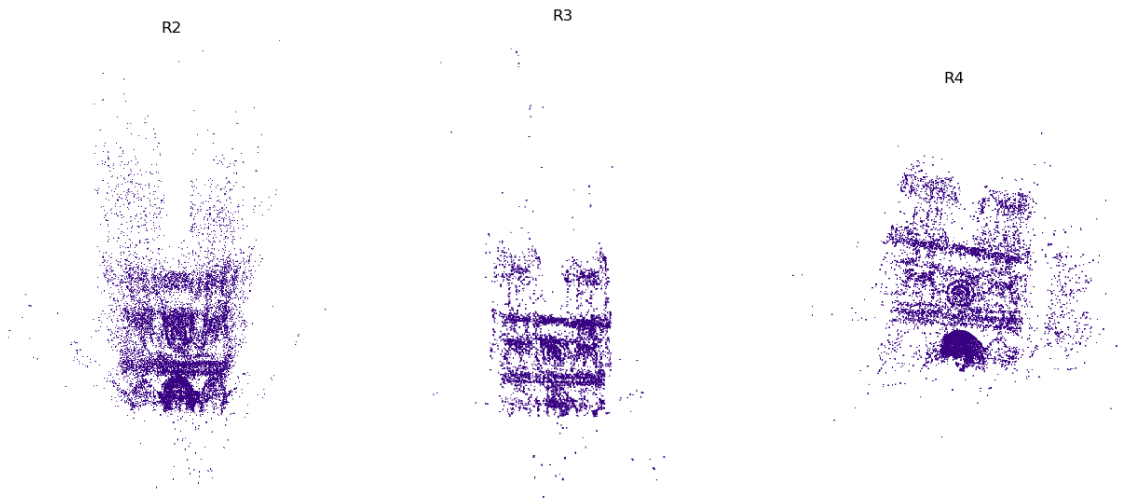


Figure 4: Reference reconstructions using first 90 images of permuted image sequences.

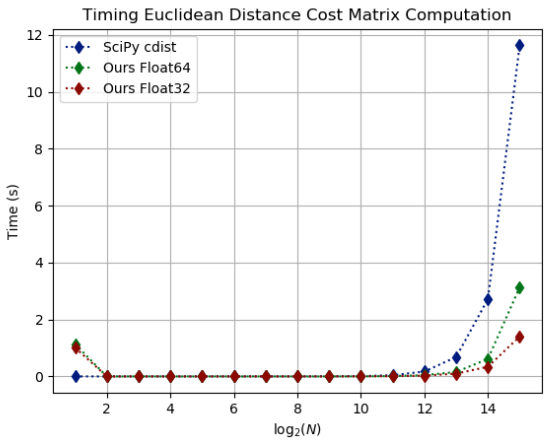


Figure 5: Experiments performed on a Google Colab machine with a single core hyper threaded Xeon Processor @2.3Ghz (No Turbo Boost), 45MB Cache and 25 GB RAM. All three methods perform nearly identically for point clouds of fewer than 2^{12} points, but our parallelized implementation with Numba [7] begins to show a large constant-time reduction in computation time past this point.

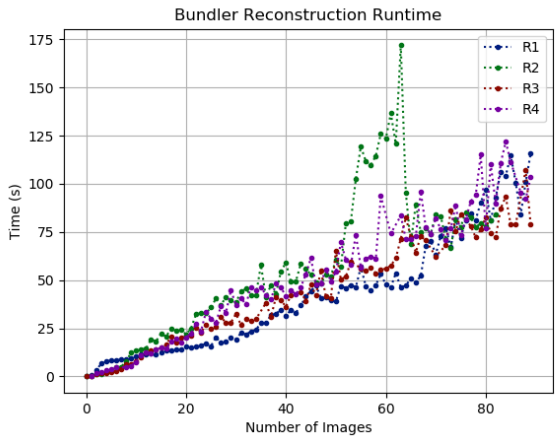


Figure 6: Shows the time taken to add the i-th image in sequence. Since the plot is linear, the reconstruction algorithm is quadratic in the total number of images. Experiments performed on an i7-7700HQ @2.8GHz with 16 GB RAM.

in order to compute matches between points. Then, sets of related keypoints are matched across pairs of images in order to form tracks of multiple keypoints across multiple images. Then, new camera parameters and locations are estimated per track, as a non-linear least squares optimization problem. In the case that there are few cameras, it is more likely that the reconstruction converged to a bad local minima, as it uses prior estimated cameras in order to estimate the location of the next camera match.

In addition to the assumption that there are a few key cameras in each reconstruction, we also found that distinct cameras could not be especially far apart as Bundler discarded any pair of images that had strictly less than 20 matching keypoints. It was especially noticable in the case of smaller data sets that many of the pictures had to be taken from the same angle, otherwise Bundler would simply not output any matches for comparing small numbers of im-

(a) 1st image used in reconstruction.

(b) 2nd image used in reconstruction.

(c) Image in the sector predicted by classifier.

Figure 7: Performed a 3D reconstruction on images (a) and (b) using Bundler. Using the point cloud output by Bundler, our classifier correctly predicted the sector of the minimizing image. (c) is a representative image of the sector. The first two images were taken from the side, and the predicted image angle faces the front.

ages. Thus, we found it necessary that images be "strung together", i.e. that a sufficient number of keypoints are shared between many images.

6. Conclusion

Our definition of partial reconstruction loss can be applied to any SfM or multi-image 3D reconstructon algorithm that generates a point cloud. In assessing partial reconstruction losses on Bundler, a popular SfM algorithm, we determined that the initial set of images has a long-lasting impact on the partial reconstruction loss. Thus, choosing the right images for a partial reconstruction is vitally important. To this end, we trained a neural network classifier that predicts where the camera that best minimizes partial reconstruction loss lies. The classifier obtains 40% accuracy, which is better than random assignment.

7. Future Work

Based on our current findings, there are multiple directions for future work. For example, one could diversify the set of characteristics used in order to predict the location of the next camera, as right now there are additional parameters we do not use, such as focal length. Another way to make the classifier perform better would be to look at other techniques, such as random forests or SVM.

An alternative approach might be to pursue the creation of a more deterministic SfM algorithm, before performing our analysis on camera importance. Bundler produces non-deterministic output, and thus it is hard to accurately say which pictures will lead to the best output. Thus, is there some way the Bundler algorithm can be modified to allow for deterministic output and not depend on the ordering of the input images? This would allow one to more easily correlate input images to their importance of construction.

References

[1] Scott Cohen. *Finding Color and Shape Patterns in Images*. Number 1620. Stanford University, Department of Computer Science, 1999.

[2] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. *CoRR*, abs/1612.00603, 2016.

[3] Sebastian Haner and Anders Heyden. Covariance Propagation and Next Best View Planning for 3D Reconstruction. In *European Conference on Computer Vision*, pages 545–556. Springer, 2012.

[4] Paul Henderson and Vittorio Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *CoRR*, abs/1901.06447, 2019.

[5] Frederick S Hillier and Gerald J Lieberman. *Introduction to Mathematical Programming*. McGraw-Hill, 1995.

[6] Roy Jonker and Anton Volgenant. A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems. *Computing*, 38(4):325–340, 1987.

[7] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A LLVM-Based Python JIT Compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery.

[8] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861 – 867, 1993.

[9] Haibin Ling and Kazunori Okada. An Efficient Earth Mover's Distance Algorithm for Robust Histogram Comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):840–853, 2007.

[10] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

[11] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, Sep. 1999.

[12] Yufu Qu, Jianyu Huang, and Xuan Zhang. Rapid 3D Reconstruction for Image Sequence Acquired from UAV Camera. *Sensors*, 18(1):225, 2018.

[13] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[14] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[15] Sameer Shirdhonkar and David W Jacobs. Approximate Earth Mover's Distance in Linear Time. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[16] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[17] Changchang Wu. Towards Linear-Time Incremental Structure from Motion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 127–134. IEEE, 2013.

[18] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore Bundle Adjustment. In *CVPR 2011*, pages 3057–3064. IEEE, 2011.

[19] Chao Yang, Fugen Zhou, and Xiangzhi Bai. 3D Reconstruction Through Measure Based Image Selection. In *2013 Ninth International Conference on Computational Intelligence and Security*, pages 377–381. IEEE, 2013.