# Optimal Compression of Point Clouds

Benjamin Robert Smith
*West Virginia University*, bbsmith1@mix.wvu.edu

# Optimal Compression of Point Clouds

Benjamin R. Smith

Thesis submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2019

Keywords: Computer Vision, Point Clouds, Structure From Motion

## Abstract

Optimal Compression of Point Clouds

Benjamin R. Smith

Image-based localization is a crucial step in many 3D computer vision applications, *e.g.*, self-driving cars, robotics, and augmented reality among others. Unfortunately, many image-based-localization applications require the storage of large scenes, and many camera pose estimators struggle to scale when the scene representation is large. To alleviate the aforementioned problems, many applications compress a scene representation by reducing the number of 3D points of a point cloud. The state-of-the-art compresses a scene representation by using a $K$-cover-based algorithm. While the state-of-the-art selects a subset of 3D points that maximizes the probability of accurately estimating the camera pose of a new image, the state-of-the-art does not guarantee an optimal compression and has parameters that are hard to tune. We propose to compress a scene representation by means of a constrained quadratic program that resembles a one-class support vector machine (SVM). Thanks to this resemblance, we derived a variant of the sequential minimal optimization, a widely adopted algorithm to train SVMs. The proposed method uses the points corresponding to the support vectors as the subset to represent a scene. Our experiments on publicly large-scale image-based localization show that our proposed approach delivers four times fewer failed localizations than that of the state-of-the-art while scaling on average two orders of magnitude more favorably.

## Acknowledgements

Firstly, I would like to thank my collaborators on this project, Dr. Victor Fragoso and Marcela Mera Trujillo. No amount of distance hindered Dr. Fragoso's ability or willingness to aid me with the completion of this project, being both my research advisor and committee chair. I would also like to thank Dr. Klinkhachorn and Dr. Nasrabadi for serving as instructors, committee members, and motivators to push me towards greater limits and accept nothing short of perfection. Without my committee's assistance and support, this Thesis would not have been possible. In addition, a thank you to all of the faculty and staff of the LANE department for everything they have taught me over the course of my five years as their student. Finally, I would like to thank my family for always being supportive of me and giving me the opportunity to pursue both a Bachelor's and Master's degree at West Virginia University.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **API** | application programming interface. 12 |
| **CDF** | cumulative distribution function. 31, 36 |
| **CDT** | coverage distinctiveness trade-off. 15, 29, 33, 38, 40, 53 |
| **QP** | quadratic program. 12, 15, 21, 22, 30 |
| **RANSAC** | random sample consensus. 8, 35 |
| **RBF** | radial basis function. 14, 19, 21, 24, 29, 33, 35, 54 |
| **SfM** | Structure from Motion. 1–3, 5, 6, 8, 13 |
| **SIFT** | scale-invariant feature transform. 10, 15 |
| **SMO** | sequential-minimal optimization. 2, 22–24, 35 |
| **SVM** | support vector machine. 2, 21, 22 |

# Chapter 1

# Introduction

Estimating the camera pose (*i.e.*, position and orientation) is a crucial step in many 3D computer vision applications, such as, self-driving cars [9, 11], robotics [6], and augmented reality [17, 30] among others. This is because these applications use camera poses and information from other sensors to understand where they are in an environment.

While many 3D computer vision systems successfully localize themselves in an environment, they struggle to scale well when the environment becomes very large [23]. Their struggle has various reasons. First, the memory and/or disk space requirements needed to store and represent the environment can be substantial. This is because the common scene representation can contain a collection of images, 3D points, and 2D image features with their respective feature descriptors (*e.g.*, SIFT [16]). Second, most of these computer vision systems use pose estimators that require longer time to operate when the representation of the scene is large. Although there exist efforts that increase efficiency of pose estimation (*e.g.*, [2, 3, 31, 32]), they still struggle when the scene representation is vast.

In order to improve the scalability of these computer vision systems, we aim to compress scene representations. In particular, we focus on compressing point clouds computed via Structure from Motion (SfM) pipelines. This is because SfM point clouds are the most common scene representation for visual-based localization.

The reader must recall that an SfM point cloud has a collection of 3D points describing the geometry of a scene. Every point in this representation (typically) has a set of 2D image features and their respective feature descriptors [15, 22]. The goal of this work is to carefully select a subset of 3D points from an SfM point cloud such that the selection provides enough information to accurately estimate a camera pose. Consequently, compressing an SfM point reduces the storage footprint because it prunes "unnecessary" points. This is useful especially for mobile agents (*e.g.*, mobile devices, robots, etc.) that have limited storage and need to self-localize in an environment.

The problem of compressing an SfM point cloud has been addressed by previous efforts [4, 15, 28]. The most common approach uses a $K$-cover-based methodology. This includes in particular the one that the state-of-the-art [4] adopts. In simple terms, the $K$-cover-based methodology aims to find a minimum subset of 3D points such that each database image sees at least $K$ points in the subset. While this approach effectively compresses an SfM point cloud, computing this subset is not optimal. Moreover, finding the right $K$ value in order to reduce the size of an SfM point cloud by a certain factor is not a trivial task.

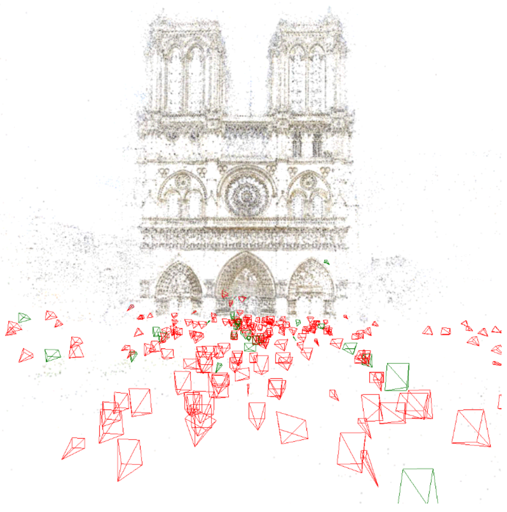
In this work, we introduce an approach that selects a subset of 3D points in an optimal fashion. To do so, we introduce an optimization problem that is convex and resembles the one-class support vector machine (SVM) [25]. Thanks to this resemblance, we derive an efficient solver based on the sequential-minimal optimization (SMO) [19]. The proposed approach aims to select points that cover sufficiently the surface to represent but at the same time present a visual distinctiveness. From the SVM perspective, the support vectors [5, 24] correspond to the 3D-point selection that best represent a scene. Moreover, the proposed approach has parameters that have intuitive meanings, which makes the parameter tuning simpler than that of the state-of-the art. Our experiments on publicly available large-scale image-based localization datasets demonstrate that our approach compresses an SfM point cloud efficiently. Additional experimental results show that the compressed point clouds computed with the proposed approach produces more accurate pose estimates than those computed using the compressed SfM point clouds produced by the state-of-the-art while also scaling favorably.

Figure 1.1 depicts how point cloud compression quality is measured. Given a point cloud, with a set of cameras viewing the points, randomly select cameras to be used as ‘query’ cameras (shown in green in 1.1a). Remove these cameras from the reconstruction (1.1b). Then, using the desired compression algorithm, compress the reconstruction to the desired compression factor (1.1c). Once compressed, attempt to add back and re-localize the ‘query’ cameras to the reconstruction (again shown in green in 1.1d). Failure to localize a camera, or localization with large error, indicates crucial points have been removed.

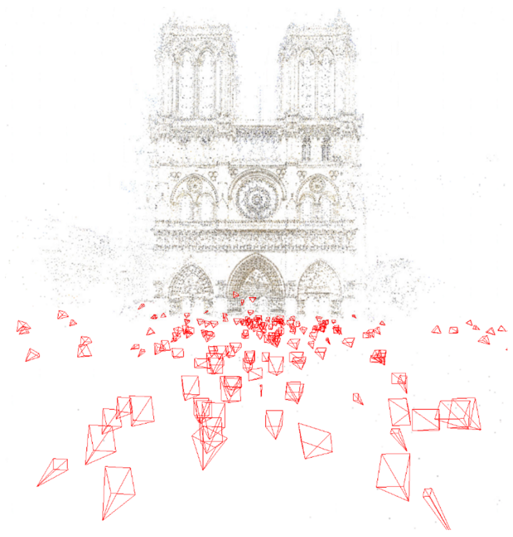
In sum, this work we present the following contributions:

1. Convex formulation for compressing SfM pipelines optimally, which is easy to tune; and

2. An efficient SMO-based constrained QP solver for compressing SfM pipelines.


(a) Query cameras (green), non-query cameras (red), and uncompressed point cloud.

(b) Query cameras removed.


(c) Compressed point cloud.

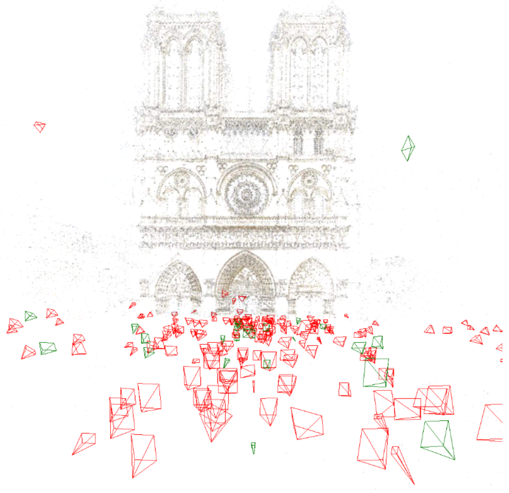

(d) Localized query camera (green) using compressed point cloud.

Figure 1.1: Starting with a point cloud: select query cameras to be localized (green), remove the query cameras, compress the reconstruction, and re-localize the query cameras.

# Chapter 2

# Related Work

The problem of reducing an SfM point cloud has been addressed by means of $K$-cover-inspired algorithms, mixed-integer quadratic program (QP) optimization methods, and deep-learning-based approaches. We cover related work that falls under these three different approaches.

## 2.1 $K$-cover-based Approaches

Li *et al.* [15] proposed a $K$-cover-inspired algorithm to select a minimum subset of 3D points such that each database image sees at least $K$ points in the subset. While this approach works well for reducing an SfM point cloud, computing the subset of 3D points is a challenging combinatorial problem. Li *et al.* [15] proposed to compute this minimum subset by incrementally building it. Their method uses a gain function that allows the algorithm to select points that contribute to the construction of the subset.

While the $K$-cover-inspired algorithm [15] reduces an SfM point cloud, it does not include information about the visual distinctiviness of each of the selected points. This aspect is important for image-based localization since visual features (*e.g.*, SIFT [16]) are crucial to establish 2D-to-3D correspondences which are the input for any pose estimator. To address this issue, Cao and Snavely [4] proposed an extension to the $K$-cover-inspired algorithm [15] that considers the coverage and the visual distinctiveness of the points. The coverage aspect imposes the constraint that the points in the subset are highly visible, *i.e.*, that a new camera observing the scene has a high probability of seeing most of the points in the subset. Moreover, their extension also includes a visual distinctiveness term that favors the selection of points that are easy to visually identify.

Although the state-of-the-art [4] effectively compresses an SfM point cloud, it has a few limitations that makes it non-optimal and hard to use. The first limitation is that the selected subset of points may not be optimal. This is because the state-of-the-art the $K$-cover-inspired algorithm is combinatorial. As such, it is hard to solve efficiently. The second limitation is that it is hard to find the parameter $K$ such that the state-of-the-art returns a compressed SfM point cloud by a certain factor. In contrast to the state-of-the-art, the proposed approach returns an optimal selection of points efficiently and has parameters that are easy to set given their intuitive meaning.

## 2.2 Mixed-integer-QP-based Approaches

An alternative approach to the $K$-cover-based algorithms is a formulation using mixed-integer programming. Park *et al.* [28] proposes a constrained quadratic program (QP) formulation mimicking the $K$-cover problem. This problem aims to compute a binary vector. The $i$-th entry of this vector is set to 1 when the $i$-th point is selected, and it is set 0 otherwise. While this approach ensures an optimal selection of points, solving the formulated constrained QP is not scalable and requires specialized mixed-integer solvers. This method struggles to scale due to the $n \times n$ matrix that encodes pairwise relationships among the points; $n$ is the number of points. Clearly, for large-scale datasets $n$ is large and the scalability of this method depends of the used solver. Although the proposed formulation also uses a constrained QP formulation, we propose an efficient solver that scales well. This is because the proposed QP formulation shares the structure of a one-class SVM [25] which can be solved efficiently using a variant of the sequential minimal optimization [19] (SMO) method.

## 2.3 Deep-learning-based Approaches

Recently, several approaches [12, 14, 13] based on deep learning [8] have been proposed to address image-based localization or pose estimation. These methods can be considered as compression approaches because the learned weights of the neural network encode the parameters of a scene. Kendall *et al.* proposed PoseNet [14], a convolutional neural network (CNN) that estimates camera poses for relocalization. Walch *et al.* [33] proposed to use a CNN in combination with LSTMs [12] to estimate camera poses. While deep-learning-based approaches have shown impressive results, they still require specialized equipment (*e.g.*, GPUs) to train them, and making them work on mobile devices can be challenging. The proposed compression method does not require specialized equipment, has an explainable or interpretable compression model, and allows estimators to compute accurate poses.

# Chapter 3

# Background

The following sections detail high-level information necessary to understand the principle components of the document, including but not limited to Structure-from-motion (SfM), three-dimensional reconstructions, and corresponding pipelines to create 3D reconstructions.

## 3.1 Structure from Motion

SfM estimates three-dimensional structures from two-dimensional images. These two-dimensional images may be taken from entirely different cameras (or perspectives), or from a single camera undergoing motion. The general idea of SfM is that a three dimensional structure is perceived from moving around it, viewing the object from multiple angles, to obtain a spatially coherent model. Internally, SfM algorithms must determine the pixel-wise correspondence between the images or motion signals and filter the poor correspondences via algorithms such as random sample consensus (RANSAC) [7]. The three dimensional points are then calculated based upon the matched features. The final result of this process is then a three-dimensional point cloud representing the structure (a 3D point cloud being a collection of data points (X, Y, Z) in a defined space).

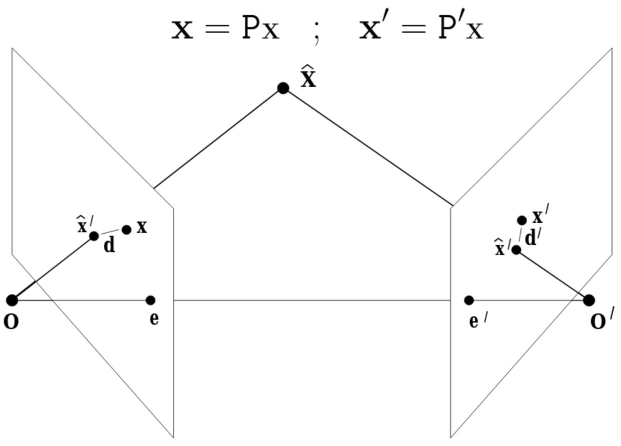


Figure 3.1: Epipolar triangulation. [10]

### 3.1.1 3D Reconstructions

In essence, three-dimensional reconstructions consist of points or meshes in a three-dimensional space representing a physical structure, *e.g.*, a building, with cameras viewing the structure. These reconstructions contain the position and orientation of each of the cameras that imaged the structure. Cameras represent their position and orientation within a $3 \times 4$ matrix. The reconstruction's points are where multiple images had matching descriptors that are triangulated into a three dimensional point, shown in Figure 3.1 and Equation 3.1. Given camera matrices $P$ and $P'$, the image points $x$ and $x'$, the 3D point $X$ is computed via minimizing the following cost function,

$$C(X) = d(x, \hat{x})^2 + d(x', \hat{x'})^2 \tag{3.1}$$

where $d$ is distance and $\hat{x}$ is the closest point on the epipole line to $x$ [10].

These reconstructions may either be created via monocular methods or binocular methods. In the former, only a single viewpoint is used to create the reconstruction, where shading is used to capture depth. In the latter, multiple images from different viewing angles are used to form the 3D shape. The latter is used in the basis of structure from motion.

#### 3.1.1.1 Descriptors & Matching

When a non-ordered set of images are presented to recreate a scene, spatial information is calculated between the images via feature extraction and matching. Features are identifiable points within the image, such as edges or corners of an object. Feature vectors, known as descriptors, describe the spatial information of these keypoints. When descriptors are (nearly) the same across images, there is a strong likelihood those image keypoints are depicting the same point in three dimensional space.

The descriptors used in our reconstructions are scale-invariant feature transform (SIFT) [16], which is a normalized 128-dimensional vector describing a keypoint. While understanding SIFT (or descriptors in general) is not directly pertinent towards point cloud optimization; we propose to use the distance between SIFT descriptors as one measure of the distinctiveness a point has towards the localization of the cameras that see it (see section 4.1 for greater detail on this point scoring technique).

#### 3.1.1.2 Examples

Two different three dimensional reconstruction examples are given, the Piccadilly Circus and the Notre Dame. Three images each are given as samples for these two structures, although it should be noted that the actual reconstructions are comprised of hundreds of images. Figure 3.2 depicts the three sample images of the corner of the Piccadilly Circus, and Figure 3.3 is the corresponding reconstruction of one of its structures (the full reconstruction contains many more buildings). In the reconstruction, the points are simply colored points in space, and the cameras are red pyramids, where the square face of the pyramid is the orientation of the camera's rotation matrix. It should be noticed that highly volatile objects within the images, such as the changing billboard, do not provide enough consistent features to create 3D points.

Figure 3.4 are three sample images of the Notre Dame, with its reconstruc-

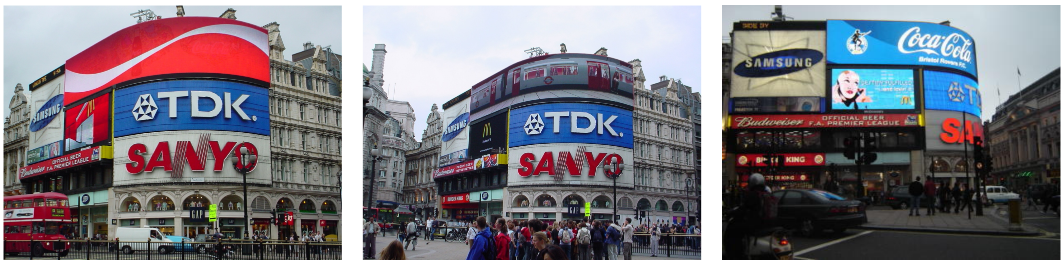


Figure 3.2: Three sample images of one of the structures in Piccadilly.

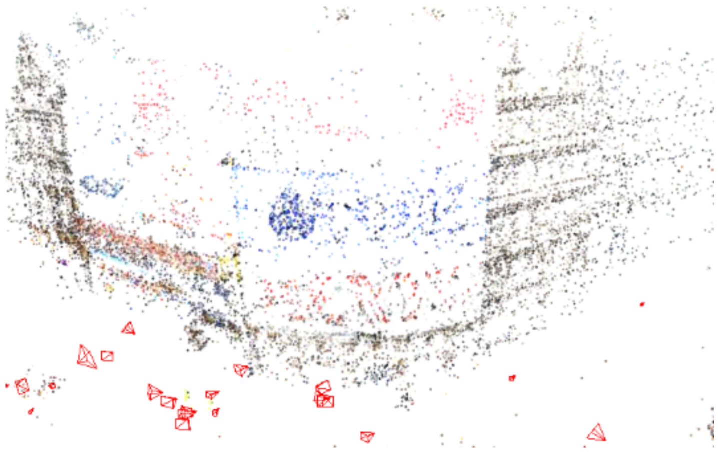


Figure 3.3: Piccadilly Reconstruction

tion shown in Figure 3.5. Once again, the reconstruction is robust to volatility, such as from pedestrians. The features the pedestrians provide do not stay consistent across multiple images, thus resulting in a lack of their presence in the reconstruction. This is desired for the purposes of our algorithm, as we will only be optimizing on the points and features relevant to the structure itself.

### 3.1.2 Pipelines

The process of creating these three-dimensional reconstructions has been streamlined into easy to use end-to-end pipelines. The primary library used in performing localization and point cloud optimization is Theia SfM [29]. Due to the original datasets (more detail in section 5.1) being created in Bundler [26], they are first converted into the realm of this library.

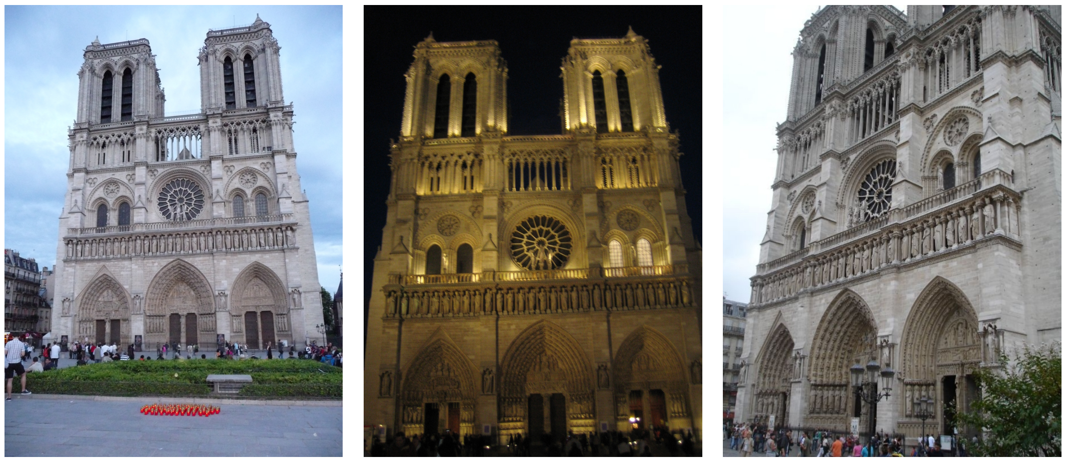


Figure 3.4: Three sample images of the Notre Dame.

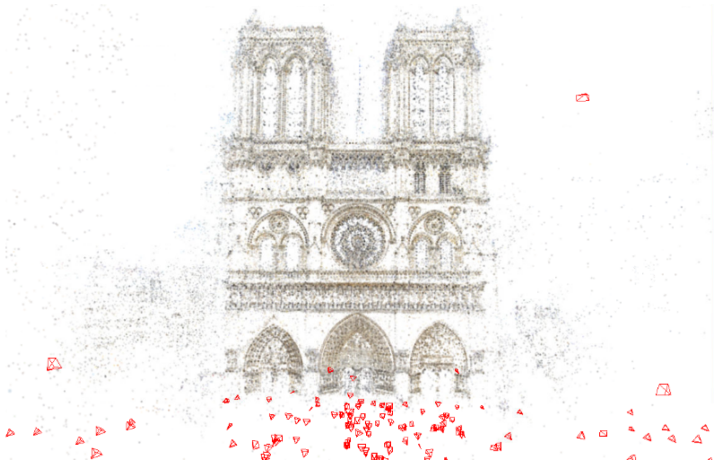


Figure 3.5: Notre Dame Reconstruction

#### 3.1.2.1 Theia SfM

As mentioned, Theia Structure from Motion is the primary library used for point cloud compression. The Theia application programming interface (API) offers an easy to use interface with algorithms implemented necessary for the goal of this project. The primary benefit to the use of Theia is that it is easily extendable for our re-localization based tasks. While the quadratic program (QP) solver (section 4.2) did not directly rely on Theia, the point scoring, minimization, and localization occurred within the Theia interface (see section 5.2 for this process). Additionally, Theia's OpenGL applications were used in visualization of the three-dimensional reconstructions, both before and after optimization occurred.

#### 3.1.2.2 Bundler

Bundler is another commonly used pipeline for performing SfM tasks. When Bundler creates a three dimensional scene, the output is written in "bundle files," which is simply the state of the scene. The file contains all camera information incrementally (including, but not limited to, rotation and translation information) and then all point information (including the coordinates and a list of all views the point is present in). For each of the datasets used, these "bundle files" are converted into Theia files, which all subsequent work is done upon.

# Chapter 4

# Algorithm

The sections below detail the process in which optimal point cloud compression occurs within the proposed algorithm, laid out in several main steps. First of all, since the 1DSfM [34] dataset (see section 5.1) is stored as "bundle files," they must be converted into Theia reconstructions and normalized. The Bundler structures tended to be heavily transitionally offset from the origin ((x=0 : y=0 : z=0 : w=1)), instead of centered about the origin. Luckily, Theia has a built in application for both conversion and normalization. Additionally, the feature and descriptor information must be re-populated for every point, as this is not contained within the Bundler files. If no feature is assigned for a point, it is removed from the reconstruction. Once converted, query cameras are randomly selected for localization and removed from the reconstruction. The first step of our algorithm then occurs, point scoring, to assign a score to every point in the reconstruction. This score may be computed in a variety of ways, see section 4.1 for more details on its computation. In essence, the purpose of the score is to serve as a visual distinctiveness measure for every point in the point cloud; aiming to keep the most visually distinct points.

Once this point score has been computed, the compression algorithm occurs. The compression algorithm has two main goals: minimize the coverage of a scene via a radial basis function (RBF) kernel (indirectly maximize the expected distance between points) and to maximize the visual distinctiveness of the scene (using the point scores). These are combined into one minimization (optimization) function which the constrained QP solver aims to solve, as explained further in section 4.2.

After the solver has chosen the optimal points to keep, the points not selected must be removed from the reconstruction, or in other words, the reconstruction is compressed. To gauge the quality of this compressed reconstruction, we re-add the previously removed query cameras and attempt to re-localize them. If successfully localized, the quality of localization is measured by positional and rotational distance from the ground truth reconstruction to the new localization. For further detail on this re-localization process, please see subsection 5.2.5

## 4.1 Point Scoring

The point scoring process is the first step of the proposed algorithm (aside from converting the "bundle files" to the proper TheiaSfM files). As previously mentioned, point scoring is a technique to assign a visual distinctiveness to every point in a reconstruction. This distinctiveness measure can be calculated in a variety of different ways, as the following sections (and subsections) show. The methods utilized to calculate this score are: descriptor distance (both normalized and non-normalized), the frequency of a point being seen by cameras, the frequency of a point being seen by cameras w.r.t. the best frequency, and a combination of both distance and frequency. The re-localization of cameras was performed with each of the above scoring techniques, while modifying a scalar controlling coverage distinctiveness trade-off (CDT).

### 4.1.1 Descriptor Distance

The first method of assigning a visual distinctiveness score is based upon SIFT descriptor distances. The concept is as follows: for a given point in the reconstruction, determine which images it is seen by, and obtain those descriptors. Then, determine the average distance between the descriptors. A higher distance will imply that the feature is slightly different among images, whereas a lower distance implies the feature is closer to being the exact same across images. In other words, this method is determining how different the feature is for the same point across its images. The formula for this score is represented by,

$$\text{score} = \frac{\sum_{i=1}^{n-1}(\sum_{j=i+1}^{n} \|\mathbf{d}_i - \mathbf{d}_j\|^2)}{S} \tag{4.1}$$

where $n$ is the number of images that see that point, $\mathbf{d_i}$ is the appropriate descriptor of image $i$, $\mathbf{d_j}$ is the appropriate descriptor of image $j$, and $S$ is the sum of the descriptor distance comparisons. This sum is calculated such that,

$$S = \frac{(n-1)n}{2} \tag{4.2}$$

where $n$ is the number of images that see the point. Note that this is the same as the formula of sum of numbers $[1, ..., n-1]$ as that is the total number of comparisons that occur in Equation 4.1.

#### 4.1.1.1 Non-Normalized

The non-normalized version of this descriptor distance remains as-is. Meaning, a lower score implies the features are similar, whereas a higher score implies the features are different. We would generally expect this to run counter-productive to the visual distinctiveness term. Since visual distinctiveness is being maximized, this non-normalized version will favor a point where its corresponding coordinate in images have different descriptors. Keep in mind the descriptors are similar enough to become a three-dimensional point to begin with. In other words, this will favor a point that is being seen from different angles (different descriptors).

#### 4.1.1.2 Normalized

The normalized version of descriptor distance adds a normalization function to the distances, in the form of a negative exponentiation. The scores become within the range [0, 1], where a score of '1' would be the exact same descriptors, and conversely a score of '0' would be vastly different descriptors. The function used for this normalization is,

$$\text{score} = \exp\left(-\frac{d^2}{2}\right) \tag{4.3}$$

where $d$ is the average distance as determined in Equation 4.1. This score will end up maximizing in favor of descriptors that are exactly the same, opposite of the non-normalized version.

### 4.1.2 Frequency

The next method of assigning a score to every point is based upon the frequency of that point being seen across images. For example, a higher score will be assigned to a point that is seen from 50 cameras than a point seen from 10 cameras. We use two different measures for this frequency. The first is simply the percentage of cameras the point is seen in. The second is the number of cameras the point is seen in w.r.t. the point with the highest frequency.

#### 4.1.2.1 Number of Images

This score, as mentioned, is the percentage of a cameras that a particular point is seen in. A point that is seen from 0 cameras (technically not possible, it would not exist otherwise), would be assigned a score of '0'. A point that is seen from every camera would be assigned a score of '1'. This is represented by,

$$\text{score} = \frac{n_i}{N} \tag{4.4}$$

where the number of cameras seeing a point $i$ is $n_i$, and $N$ is the number of cameras. Thus, this method of point scoring will cause points seen in numerous images to be more favored.

#### 4.1.2.2 Best Point

The best point score assigns the score value of the current point w.r.t. the frequency of the best point. For example, if the best point is seen within 50 views, it will be assigned a score of '1'. All other scores will be the proportion of cameras viewing it, out of 50. This may be modeled by,

$$\text{score} = \frac{n_i}{\max\{n_1, ..., n_P\}} \tag{4.5}$$

where the number of cameras seeing a point $i$ is $n_i$ and $P$ is the number of points.

### 4.1.3 Combination

The combination based score is computed as a weighted sum between distance and frequency. The weight is a changeable parameter, but for the purpose of our results, we chose 0.5. This score is simply,

$$\text{score} = w \cdot d + (1 - w) \cdot f \tag{4.6}$$

where $w$ is the weight, $d$ is the distance score computed in Equation 4.3, and $f$ is the frequency score computed in Equation 4.5.

## 4.2 Constrained QP Solver

The proposed problem reduces a point cloud considering two terms: spatial coverage of the scene and visual distinctiveness of each of the 3D points. Intuitively, we aim to select points that are far away from each other such that they cover most of the scene and that have good visual distinctiveness. While these terms are also considered by the state of the art [4], the proposed approach introduces a novel convex optimization problem that can be solved efficiently. Unlike the state of the art which builds on the $K$-cover problem, the proposed formulation aims to learn a sparse discrete probability distribution over the set of points. This learned distribution has non-zero values on the selected points and zero on the points that are discarded.

Mathematically, we aim to learn a sparse distribution $\boldsymbol{\alpha}$ over the $m$ 3D points of the input point cloud. To formulate this problem mathematically, we need to define two terms that measure the spatial coverage and visual distinctiveness of each of the points as a function of $\boldsymbol{\alpha}$.

In order to measure the coverage of the scene, we propose to use the following term:

$$C = \boldsymbol{\alpha}^{\intercal} K \boldsymbol{\alpha} \tag{4.7}$$

where the entries of the matrix $K \in \mathbb{R}^{m \times m}$ are

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \tag{4.8}$$

$\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^3$ are two point positions, and $\sigma$ is a parameter that controls when two points are considered close enough. Because the matrix $K$ is an RBF kernel [24], the coverage term $C$ ranges between 0 and 1. It decreases when two points are far away and increases when two points are close to each other. To fulfill the goal of keeping points that are far apart of each other in order to cover most of the scene, then we need to minimize $C$.

The term $C$ can be seen as the expected coverage score via the RBF kernel, *i.e.*,

$$\begin{aligned}
\mathbb{E}\left[k(\mathbf{x}_i, \mathbf{x}_j)\right] &= \sum_{i,j} k(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i, \mathbf{x}_j) \\
&= \sum_{i,j} k(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i) p(\mathbf{x}_j) \\
&= \sum_{i,j} k(\mathbf{x}_i, \mathbf{x}_j) \alpha_i \alpha_j \\
&= \boldsymbol{\alpha}^{\intercal} K \boldsymbol{\alpha} = C,
\end{aligned} \tag{4.9}$$

where $\mathbb{E}[\cdot]$ is the expectation operator, and $p(\mathbf{x}_i, \mathbf{x}_j) = p(\mathbf{x}_i)p(\mathbf{x}_j)$ is the joint probability encoding the chances that the pair $(\mathbf{x}_i, \mathbf{x}_j)$ of points is selected. By minimizing $C$ over $\boldsymbol{\alpha}$, we are indirectly maximizing the expected distance between the pairs of points in the point cloud. Thus, by minimizing $C$ we enforce the algorithm to learn a distribution $\boldsymbol{\alpha}$ that aims to maximize the expected pairwise distance between points in the input point cloud. Since the algorithm learns a sparse distribution $\boldsymbol{\alpha}$, the algorithm will learn to select only a handful of points.

To model the visual distinctiveness, we propose to use the following term:

$$D = \mathbf{d}^{\intercal}\boldsymbol{\alpha}, \tag{4.10}$$

where $\mathbf{d} \in \mathbb{R}^m$ is a vector holding a visual distinctiveness score for every point in the input point cloud. Our goal is to keep the most visual distinctive points. Consequently, we need to maximize this term.

The term $D$ also can be interpreted as the expected visual distinctiveness of the selected points, *i.e.*,

$$\mathbb{E}[d_i] = \sum_i d_i p(\mathbf{x}_i) = \sum_i d_i \alpha_i = \mathbf{d}^{\intercal}\boldsymbol{\alpha} = D, \tag{4.11}$$

where $d_i$ is the visual distinctiveness score for the $i$-th point, and $\alpha_i = p(\mathbf{x}_i)$ is the probability of selecting the $i$-th point.

To put everything together, we want to minimize $C$ (*i.e.*, maximize the expected coverage) and maximize $D$ (*i.e.*, the expected visual distinctiveness). We propose the following cost function that fulfills our goals:

$$J = C - \tau D = \boldsymbol{\alpha}^{\intercal}K\boldsymbol{\alpha} - \tau\mathbf{d}^{\intercal}\boldsymbol{\alpha}, \tag{4.12}$$

where $\tau$ is a scalar that controls the trade-off between spatial coverage term $C$ and the visual distinctiveness $D$. By minimizing $J$ over $\boldsymbol{\alpha}$, we minimize $C$ and maximize $D$. When visual distinctiveness is more important, then $\tau$ must be high. On the other hand, when the coverage term is more important, then $\tau$ must be near zero.

To learn the sparse distribution $\boldsymbol{\alpha}$ that minimizes $J$, we propose the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha} - \tau \mathbf{d}^\intercal \boldsymbol{\alpha} \\ \text{subject to} \quad & \sum_i^m \alpha_i = 1, \\ & 0 \le \alpha_i \le \frac{1}{\nu m}; \quad i = 1, \ldots, m, \end{aligned} \tag{4.13}$$

where $\nu \in (0, 1]$ is a scalar that controls the sparsity of the distribution $\boldsymbol{\alpha}$. This parameter thus controls the compression rate. This is because when $\nu = 1$, then $\boldsymbol{\alpha}$ becomes a uniform distribution, which is equivalent to no compression. On the other hand, when $\nu < 1$, then we allow the algorithm to put more mass on a few points. In this case, this is equivalent to select only a few points which reduces the size of a point cloud.

The problem proposed in Eq. (4.13) is a QP. As such, the proposed problem is convex. This is because the RBF kernel matrix $K$ is positive-semi-definite matrix [24], and the proposed problem has linear equality and inequality constraints. Any convex solver (*e.g.*, Newton-like solvers [1]) can be used to find $\boldsymbol{\alpha}$. However, it is well known that these methods do not scale well when the number of data points is large. Nevertheless, we can exploit the intimate relationship that this problem has with one-class SVM to derive efficient solvers.

### 4.2.1 Relation with One-class SVMs.

The proposed problem in Eq. (4.13) has a direct relationship to one-class classifiers. We can obtain the exact one-class SVM dual formulation [25] by setting $\tau = 0$. With this setting, we omit the linear term $D$ and only keep the coverage term $C$. This reveals that the proposed approach with this setting reduces a point cloud by keeping the support vectors. Recall that the support vectors have a corresponding non-zero entry in $\boldsymbol{\alpha}$, and are the points that allow an SVM to define a decision boundary for recognition.

While this relationship provides insight as to how the proposed algorithm operates, it also enables an opportunity to derive an efficient solver. This is because efficient and scalable solvers that train an SVM exist, *e.g.*, the SMO [19]. Unfortunately, we cannot directly use the one-class SVM SMO solver for the proposed problem. There are two reasons that limit the SMO solver for our proposed problem. The first one is that the original one-class-SVM SMO solver does not consider a linear term; in our case the distinctiveness term $D$. The second reason is that the SMO uses the SVM decision rule to determine efficient variable updates. Our proposed problem is not a classification one. As such, our problem does not have a decision rule. Nevertheless, as we discuss in the next section, it is still possible to derive an SMO-like solver that can efficiently solve the proposed problem.

### 4.2.2 Efficient SMO-like Solver

Inspired by the SMO solver, we aim to formulate the simplest sub-problem that we can sequentially solve at a time. By solving these sub-problems sequentially, we can find the solution for the proposed problem. As shown by Platt [19], the simplest problem that we can solve in an SVM involves a quadratic program with only two variables. The advantage of solving a QP problem with two variables is that we can solve it analytically. Consequently, we avoid expensive matrix operations (*e.g.*, matrix inversions and multiplications) which are fundamental operations in Newton-based optimization methods.

To obtain the simplest QP problem with two variables, we need to algebraically manipulate Eq. (4.12). Recall that the goal is to obtain a cost function $J'$ that focuses on only two variables: $\alpha_i$ and $\alpha_j$ which are the $i$-th and $j$-th entries of $\boldsymbol{\alpha}$, respectively, and $i \neq j$. After algebraic manipulations, we obtain the following $J'$:

$$\begin{aligned} J'(i,j) &= \alpha_i^2 + 2\alpha_i\alpha_j K_{ij} + \alpha_j^2 \\ &+ 2\alpha_i \sum_{l\neq i, l\neq j} \alpha_j K_{il} + 2\alpha_j \sum_{l\neq i, l\neq j} \alpha_l K_{jl} \\ &- \tau d_i \alpha_i - \tau d_j \alpha_j + g\left(\{\alpha_t : t \neq i,j\}\right) \\ &= J, \end{aligned} \tag{4.14}$$

where $g(\cdot)$ is a function including all the remaining entries $\{\alpha_t : t \neq i,j\}$ in $\boldsymbol{\alpha}$. For the full derivation of $J'$, we refer the reader to the supp. material.

The original problem shown in Eq. (4.13) aims to learn a probability distribution $\boldsymbol{\alpha}$. Since $J'$ focuses on only two variables, we need to update the equality constraints when we only optimize for the two entries $\alpha_i$ and $\alpha_j$. To do this, we need to ensure that the sum of all the entries in $\boldsymbol{\alpha}$ equals to one. At the same time we still need to ensure the inequality constraints shown in Eq. (4.13). After considering these aspects, we obtain the following the problem:

$$\begin{aligned} \underset{\alpha_i,\alpha_j}{\text{minimize}} \quad & J'(i,j) \\ \text{subject to} \quad & \alpha_i + \alpha_j = \Delta, \\ & 0 \leq \alpha_i \leq \frac{1}{\nu m}; \quad i = 1,\ldots,m, \end{aligned} \tag{4.15}$$

where $\Delta$ is the joint probability mass between $\alpha_i$ and $\alpha_j$. This means that we can minimize $J'$ as long as we maintain the probability mass $\Delta$ between $\alpha_i$ and $\alpha_j$ constant. Similar to the SMO, this constraint imposes a solution over a line $\alpha_i + \alpha_j = \Delta$ and keeps a valid sum: $\sum_i \alpha_i = 1$. This is because the solver assumes that the starting probability distribution $\boldsymbol{\alpha}$ is feasible:, *i.e.*, it sums up to one and satisfies the inequality constraints.

Similar to the SMO algorithm, the proposed algorithm needs to perform iteration to select a pair of variables $\alpha_i$ and $\alpha_j$, and solve the problem shown in Eq. (4.15). To solve this simplified problem analytically, we can leverage the equality constraint to set $\alpha_j = \Delta - \alpha_i$. Via substitution, we obtain the following simplified cost function

$$\begin{aligned} J'(\alpha_i) = \alpha_i^2 &+ 2\alpha_i \left(\Delta - \alpha_i\right) K_{ij} + \left(\Delta - \alpha_i\right)^2 \\ &+ 2\alpha_i \sum_{l \neq i, l \neq j} \alpha_l K_{il} \\ &+ 2\left(\Delta - \alpha_i\right) \sum_{l \neq i, l \neq j} \alpha_l K_{jl} \\ &- \tau\left(d_i \alpha_i + d_j \left(\Delta - \alpha_i\right)\right) + g\left(\{\alpha_t : t \neq i, j\}\right) \end{aligned} \tag{4.16}$$

Given that $J'(\alpha_i)$ is a function of a single variable, we can obtain the optimal $\alpha_i^\star$ analytically by solving $\frac{\partial J'}{\partial \alpha_i} = 0$. The analytic solution for this relationship is:

$$\alpha_i^\star = \frac{1}{2}\left(\frac{T}{\left(2\left(1 - K_{ij}\right) + \Delta\right)}\right), \tag{4.17}$$

where

$$T = \tau * \left(c_i - c_j\right) - 2 \sum_{l \neq i, l \neq j} a_l K_{il} + 2 \sum_{l \neq i, l \neq j} a_l K_{jl}. \tag{4.18}$$

Basically we want to maximize the coverage and distinctiveness at the same time. In other words, we want to select points that are far from each other but those points must be easy to identify. Inspired by the SMO [19] algorithm, the solver aims to solve this problem by solving simpler problem. Consider the derivation of the sub-problem in terms of $\alpha_1$ and $\alpha_2$ for optimization, $k(x_i, x_j) = k(x_j, x_i)$ because is a RBF kernel and a shorthand of $k_i j = k(x_i, x_j)$. Therefore, the coverage term is the sum of the all elements in the following matrix:

$$\begin{bmatrix} \alpha_1^2 & \alpha_1\alpha_2 k_{12} & \dots & \alpha_1\alpha_m k_{1m} \\ \alpha_2\alpha_1 k_{21} & \alpha_2^2 & \dots & \alpha_2\alpha_m k_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_m\alpha_1 k_{m1} & \alpha_m\alpha_2 k_{m2} & \dots & \alpha_m^2 \end{bmatrix}$$

where $\alpha_i$ is the $i^{th}$ alpha entry for the $i^{th}$ point. The coverage term can be rewritten as follows:

$$\begin{aligned}\text{coverage} &= \sum_i \alpha_i^2 + 2\sum_{i=1}^{m}\sum_{j=i+1}^{m}\alpha_i\alpha_j k_{ij} \qquad (4.19)\\ &= \alpha_1^2 + 2\alpha_1\alpha_2 k_{12} + \alpha_2^2 + 2\alpha_1\sum_{j=3}^{m}\alpha_j k_{1j} \qquad (4.20)\\ &\quad + 2\alpha_2\sum_{j=3}^{m}\alpha_j k_{2j} + \phi\end{aligned}$$

where $\phi = \sum_{i=3}\sum_{j=3}\alpha_i\alpha_j k_{ij}$. In terms of $\alpha_1$ and $\alpha_2$, the distinctiveness term is:

$$\begin{aligned}\text{distinctiveness} &= \sum_i C_i\alpha_i \qquad (4.21)\\ &= C_1\alpha_1 + C_2\alpha_2 + \gamma \qquad (4.22)\end{aligned}$$

where $\gamma = \sum_{i=3}^{m} C_i\alpha_i$. We can discard $\phi$ and $\gamma$ since is independent of $\alpha_1$ and $\alpha_2$.

Thus the total cost function to minimize emphasizing the two $\alpha_1$ and $\alpha_2$ is

$$\begin{aligned}Z(\alpha_1,\alpha_2) &= \alpha_1^2 + 2\alpha_1\alpha_2 k_{12} + \alpha_2^2 + 2\alpha_1\sum_{j=3}\alpha_j k_{1j} \qquad (4.23)\\ &\quad + 2\alpha_2\sum_{j=3}\alpha_j k_{2j} - \tau C_1\alpha_1 - \tau C_2\alpha_2\end{aligned}$$

Then, the subproblem to solve is

$$\begin{aligned}\underset{\alpha}{\text{minimize}} \quad & Z(\alpha_1,\alpha_2)\\ \text{subject to} \quad & \alpha_1 + \alpha_2 = \Delta, \qquad (4.24)\\ & 0 \le \alpha_i \le \frac{1}{\nu m}; \quad i = 1, 2.\end{aligned}$$

Using the linear constraint to get $\alpha_2$ as a function of $\alpha_1$ (i.e $\alpha_2 = \Delta - \alpha_1$). We use this relationship to get the cost function of the subproblem as a function of $\alpha_1$ only.

$$\begin{aligned}Z(\alpha_1) &= \alpha_1^2 + 2\alpha_1(\Delta - \alpha_1)k_{12} + (\Delta - \alpha_1)^2 \qquad (4.25)\\ &\quad + 2\alpha_1\sum_{j=3} a_j k_{1j} + 2(\Delta - \alpha_1)\sum_{j=3} a_j k_{2j}\\ &\quad - \tau C_1\alpha_1 - \tau C_2(\Delta - \alpha_1)\end{aligned}$$

Solving the problem at the optimal point and then solving for $\alpha_1$, we get:

$$\begin{aligned}\frac{\partial Z}{\partial \alpha_1} = 2\alpha_1 + 2(\Delta - 2\alpha_1)k_{12} - 2(\Delta - \alpha_1) \\ + 2\sum_{j=3} a_j k_{1j} - 2\sum_{j=3} a_j k_{2j} \\ - \tau C_1 + \tau C_2\end{aligned} \tag{4.26}$$

$$\alpha_{1\text{optimal}} = 0.5\left(\frac{T}{2(1-k_{12})} + \Delta\right) \tag{4.27}$$

where $T = \tau(C_1 - C_2) - 2\sum_{j=3} a_j k_{1j} + 2\sum_{j=3} a_j k_{2j}$.

Since we are solving a box constrained problem we have that

$$\alpha_1 = \max\left(0, \min\left(\min\left(\frac{1}{\nu m}, \Delta\right), \alpha_{1\text{optimal}}\right)\right) \tag{4.28}$$

Consequently, $\alpha_2 = \Delta - \alpha_1$.

### 4.2.3 The Complete Algorithm

From the above mentioned, the whole proposed method algorithm is as follow:

**Initialization of proposed method algorithm** [20] For the fraction of the compression factor $\nu$, the values for $\alpha_i$ where $i$ is in that fraction are:

$$\alpha_i = \begin{cases} \frac{1}{\nu m} & \text{if } \nu m \text{ is integer} \\ (0, \frac{1}{\nu m}) & \text{Otherwise} \end{cases} \tag{4.29}$$

Otherwise, the values for $\alpha_i = 0$

**Algorithm 1:** Simple-SMO algorithm

```
Input       : m-by-4 matrix $\mathbf{M}$ - 3D point and their aggregated confidence
              score
Output      : a vector of alphas, $\boldsymbol{\alpha}$
Parameters: Compression factor $\nu$, Coverage and Distinctiveness trade-off
              factor $\tau$, Sigma value for RBF
Upper_bound = $\frac{1}{\nu m}$
Initialization for $\alpha_i$
Main function:
for (i = 1 ... number of iterations) do
    Generate pair $(\alpha_1, \alpha_2)$ randomly;
    function Solver for $\alpha_1$ $(\alpha_1, \alpha_2)$;
    function Update Alpha pair $(\alpha_{1\text{optimal}})$;
end
Function Solve for $\alpha_1$ $(\alpha_1, \alpha_2)$;
if $(\alpha_1 = \alpha_2)$ then
    return;
end
$\Delta = \alpha_1 + \alpha_2$;
```

# Chapter 5

# Experiments

## 5.1 Datasets

In order to measure the quality of our proposed algorithm compared to other baseline algorithms, we use the 1DSfM datasets [34]. These datasets each contain a 3D reconstruction, including the camera positions and orientations, as well as the original images used to create said reconstruction.

### 5.1.1 1DSfM

1DSfM presents fourteen large-scale reconstructions that includes the images used to compute the reconstructions, 2-view matches, and epipolar geometry. This information is present in the form of "bundle files" which must first be converted into a format usable by TheiaSfM. This is done by an application that comes existing with Theia out of the box. Additionally, the feature/descriptor information of each point must be repopulated. Using a reprojection error threshold of 8 pixels, each point was assigned the proper feature for each view it was seen within. Any tracks that were never assigned a feature were removed from the reconstruction

The fourteen models contain within the dataset are of varying large-scale sizes, real world structures, and features. It is important to realize that the Bundler model coming with the models is not an "exact" ground truth. As in, it should not be considered a perfect 1-to-1 reconstruction of these real world objects. However, it is sufficient to use as a general ground truth for our optimization purposes. Please see section 7.1 for information on the models, such as the number of images and the number of points, for both query and validation tests.

## 5.2 Integration with Theia & Testing

The algorithm must be tested against the baseline algorithms to measure its performance. The entire testing procedure is described in the following steps:

1. Convert the 1DSfM "bundle files" to a Theia reconstruction.
2. Normalize the reconstruction about the origin, and repopulate features.
3. Randomly select 10% of the cameras to be removed, and remove them.
4. Perform point scoring on the reconstruction (or perform any other necessary information retrieval for running baselines).
5. Select the points to be kept and removed from the reconstruction.
6. Compress the reconstruction by removing undesired points.
7. Perform localization; add back the previously removed cameras. This process returns the success metrics.

### 5.2.1 Query View Removal

As mentioned, 10% of views are removed, optimization occurs, then those 10% of views are attempted to be re-localized. We measure the re-localization performance when varying point scoring techniques, RBF sigma, and CDT values to obtain the optimal parameters. Upon testing, a different subset of views need to be used as to not "learn to the test set." To account for such, a subset of 10% are chosen for query and another 10% are chosen for validation, with absolutely no overlap between the two sets. The query set is used to see which techniques are best, while the validation set is used to compare to the other baseline algorithms (which are also using validation). The proper view set (query or validation) is then removed, and saved into their own respective Theia files.

### 5.2.2 Point Scoring

As section 4.1 discussed all the various techniques, an application was added to Theia's interface to perform this scoring. The desired technique may be selected, and point scoring occurs, ultimately outputting a file of every point such that it reads, ($x_i$, $y_i$, $z_i$, $score_i$), where $i$ is the point in question.

### 5.2.3 Point Selection

The point selection process for our algorithm is the QP solver (section 4.2). The format for the output file must be parse-able such that it contains the points to be removed (or kept) in a text file, one point index per row. An example of point selection is shown in Figure 5.1, where the 20% of points to keep are shown in green and the remaining points to remove shown in black.

### 5.2.4 Point Removal

An application was added to Theia such that when presented with a list of points, they are removed from the reconstruction. The resulting minimized reconstruction is saved into a separate file, and does not overwrite the non-compressed pruned reconstruction.

### 5.2.5 Localization Metrics

There are three primary metrics by which the success of the algorithm is measured, compared to the baselines. These metrics are: failure rate upon re-

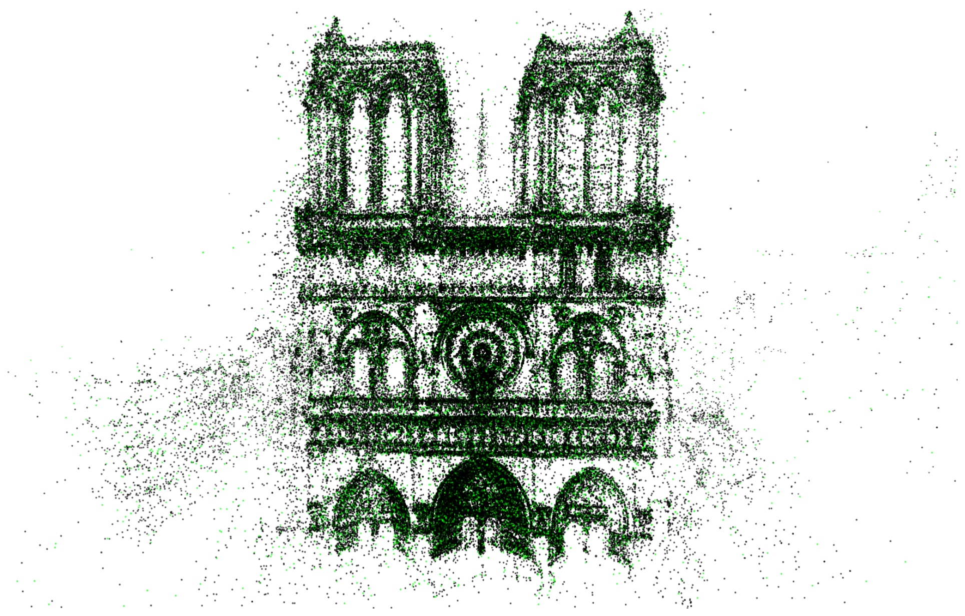


Figure 5.1: Example of point selection on the Notre Dame. Points to keep shown in green, points to remove shown in black.

localization, the positional distance from re-localization to ground-truth, and rotational distance from re-localization to ground-truth. The positional and rotational measures are graphed as a cumulative distribution function (CDF) of error vs. the fraction of cameras re-localized. The CDF allows seeing clear distance outliers upon re-localization.

#### 5.2.5.1 Re-Localization Failure Rate

This metric is fairly self-explanatory, it is simply the percentage of cameras that were able to be re-localized after compression occurred. One important thing to note about this is that when a camera **could not** be localized, its positional and rotational distance will be not be recorded for the CDF plots. This was done because there is no "maximum" distance value that could be applied. For example, if every camera fails to be localized (which should be a clear indicator the point selection was terrible), it will record no positional or rotational distances. Thus, a general, but not strict rule, is that this should be used as the primary indicator for localization success, and distances as "tie breakers."

#### 5.2.5.2 Positional Distance

The positional distance metric is calculated as the L2 (Euclidean) distance between two points. Please remember all coordinate points should be on the same homogeneous scale when computing distance (the fourth value for 3D points). The formula is simply,

$$\text{distance} = \sqrt{\sum_{i=1}^{3}(p_i - q_i)^2} \tag{5.1}$$

where **p** and **q** are three dimensional points.

#### 5.2.5.3 Rotational Distance

The rotational distance may be computed one of two ways. Rotation matrices have the property that they are orthogonal matrices with determinant 1. Meaning: for any $n$-dimensional rotation matrix $R$ in $\mathbb{R}^n$, $R^\intercal = R^{-1}$. Thus, if the re-localized rotation matrix is exactly equal to the ground truth rotation matrix,

$$R_{gt}R_{rl}^\intercal = I \tag{5.2}$$

where $R_{gt}$ is the ground truth matrix, $R_{rl}$ is the re-localized matrix, and $I$ is the identity matrix. We can compute the rotational distance then as the norm of this multiplication minus to the identity matrix, or in other words, the distance from being the identity matrix. This resultant formula is,

$$\text{distance} = ||I - R_{gt}R_{rl}^\intercal|| \tag{5.3}$$

which is the first rotational metric. The second metric is to convert the matrices into quaternions, the dot product of which corresponds to the cosine of the half angle between them. More specifically:

$$\text{distance} = 2 * \arccos\left(|Q_{gt} \cdot Q_{rl}|\right) \tag{5.4}$$

where the distance is in radians, $Q_{gt}$ is the ground truth quaternion, and $Q_{rl}$ is the re-localized quaternion. This second metric (converted to degrees) was used in our experiments, however an option exists to choose either.

## 5.3 Results

### 5.3.1 Query

The views for each dataset (section 5.1) were broken into three sets each, as described in subsection 5.2.1: 10% for query, 10% for validation, and the remainder 80%, with no overlap between the sets. For the query set, the point scoring technique (section 4.1), RBF sigma, and CDT were empirically modified to choose the optimal values. In order to accomplish this, first the RBF sigma value was modified with a CDT of 0, to isolate itself from the different scoring techniques (a CDT of 0 does not use distinctiveness as shown in Equation 4.12). The RBF sigma is modified from 1.0 to 10.0 in 0.5 increments. Once the RBF is chosen and held constant, the CDT is modified from 0.2 to 2.0 in 0.2 increments for each of the scoring techniques. The best combination of these values is then chosen, as determined by the metrics in subsection 5.2.5. This chosen set of parameters is then used for the validation set of views, for the actual testing and comparison against the other baseline methods.

The chosen parameters for each model are given in Table 5.1, and the broken down counts are given in Table 5.2 through Table 5.4.

A breakdown of the query failure rates, with respect to the different scoring techniques may be seen in Table 7.2 in section 7.2. Across the datasets, 658 cameras were queried for localization for each scoring method, with each camera under going five trials of localization. Distance without normalization failed 1.19%, distance normalize failed 1.08%, frequency based on number of images failed 34.50%, frequency based on best point failed 81.44%, combination failed 1.69%, and only RBF (no distinctiveness) failed 1.16%. The

| **Model** | **RBF Sigma** | **CDT** | **Score** |
|---|---|---|---|
| Alamo | 1 | 0.4 | Combo |
| Ellis Island | 1 | 0 | (any) |
| Gendarmenmarkt | 4 | 1.8 | Distance Norm |
| Madrid Metropolis | 5 | 1.8 | Combo |
| Notre Dame | 8.5 | 1.0 | Combo |
| NYC Library | 2.5 | 0.4 | Distance Norm |
| Piazza del Popolo | 1.5 | 1.0 | Combo |
| Piccadilly | 2 | 0.4 | Distance Norm |
| Roman Forum | 9 | 1.2 | Distance Norm |
| Tower of London | 4 | 0.6 | Distance Norm |
| Trafalgar | 3 | 0.2 | Distance Norm |
| Union Square | 9.5 | 1.6 | Distance Norm |
| Vienna Cathedral | 3 | 1.2 | Combo |
| Yorkminster | 7.5 | 1.4 | Distance Norm |

Table 5.1: Chosen parameters based on query results.

| **Score** | **Distance Non-Norm** | **Distance Norm** | **Freq. Num. Imgs.** | **Freq. Best Pt.** | **Combo** | **any (0 cdt)** |
|---|---|---|---|---|---|---|
| **Count** | 0 | 8 | 0 | 0 | 5 | 1 |

Table 5.2: Count of scores being best among the datasets.

| **Sigma** | **1** | **1.5** | **2** | **2.5** | **3** | **3.5** | **4** | **4.5** | **5** | **5.5** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Count** | 2 | 1 | 1 | 1 | 2 | 0 | 2 | 0 | 1 | 0 |

| **Sigma** | **6** | **6.5** | **7** | **7.5** | **8** | **8.5** | **9** | **9.5** | **10** |
|---|---|---|---|---|---|---|---|---|---|
| **Count** | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

Table 5.3: Count of RBF sigmas being best among the datasets.

| **CDT** | **0** | **0.2** | **0.4** | **0.6** | **0.8** | **1.0** | **1.2** | **1.4** | **1.6** | **1.8** | **2.0** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Count** | 1 | 1 | 3 | 1 | 0 | 2 | 2 | 1 | 1 | 2 | 0 |

Table 5.4: Count of CDT sigmas being best among the datasets.

frequency based scoring metrics failed drastically more often than the other three scoring methods, with the normalized distance metric performing the best.

### 5.3.2 Validation & Method Comparisons

The validation set of views are used in comparison between the different baseline methods, once the optimal parameters for our algorithm have been chosen. The methods used are:

1. "Ground Truth." The selected views are removed, **no compression occurs**, and the selected views are relocalized. The purpose behind not compressing is that some views may be challenging, or even impossible, to localize due to us having to manually repopulate the features per track. Some dataset models may be 'bad' or too small, with very few features assigned. Thus it provides a good metric of what the (general) upper limit should be, aside from the inherent randomness of RANSAC based localization.

2. "Random (5)." This method is simply the random minimization of the reconstruction five times, plotting their concatenated errors. Essentially, the purpose of this method is to verify our algorithm has achieved the goal of meaningfully compressing a point cloud (by simply beating this method).

3. "Minimal Scene." As mentioned in chapter 2, this is an extension of the $K$-cover-inspired algorithm that considers coverage and distinctiveness of the points [4].

4. "Simple-SMO (Ours)." Our algorithm as described in chapter 4 aims to compressed by means of an RBF kernel and visual distinctiveness, solved in SMO-like means.

For all methods, five trials occur for each localization step. Meaning, each camera is attempted to be re-localized five times for a given compression. This was done to alleviate a 'poor' localization at the hands of randomness, and to achieve more accurate results.

Figure 5.2 shows the timing to select points of our algorithm (Simple-SMO) vs. Minimal Scene. The plot's timing y-axis (seconds) is shown on the logarithmic scale, with a trend line fit to the points. An important aspect to note about these plots is that it only includes the point selection time of both algorithms, for the proper parameters (it does not include the time needed to obtain the files to run the algorithms). For example, in the Minimal Scene algorithm, the compression factor is non-deterministic. As such, the algorithm was ran numerous times, varying the $K$-cover value until the desired compression was met; only the time of the proper compression factor was plotted. The plot also does not account for the fact Minimal Scene failed to meet the compression factor for some datasets (The $K$-cover should have been higher, resulting in more time to select points). The exact timings for each dataset, and the number of points, are shown in their corresponding section. It may be noted that the "Ground Truth" and the "Random (5)" were not timed, as the time "Ground Truth" has no compression and the "Random (5)" will be the insignificant amount of time to randomly generate numbers.

The following subsections discuss the differences between the baselines applied to the different models. As mentioned, in subsection 5.2.5, the localization metrics used are: re-localization failure rate, positional distance, and rotational distance where CDFs were computed for the distances. These more clearly illustrate how many of the cameras are 'difficult' to localize, by seeing at which data fraction the error increases. An important aspect to note about these errors, is that upon a failed localization, no error is added to the graphs (as explained in subsubsection 5.2.5.1), as there is no good metric for a 'maximum' error to give. Thus, it is possible a high fail rate may *appear* to be better graph wise, but that is simply because no error was assigned. Mean-

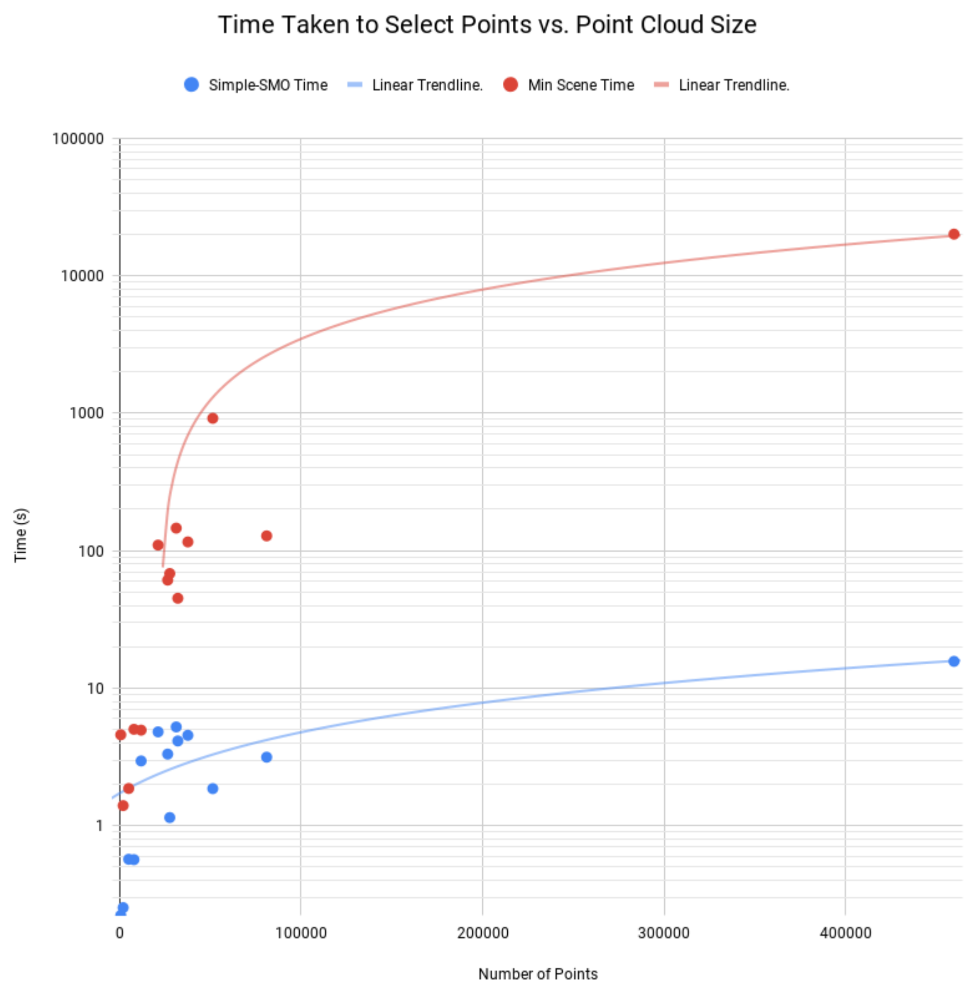


Figure 5.2: Time taken to select points vs. the size of a point cloud.

ing, instead of localizing poorly and being an outlier, it does not affect the cumulative distribution graph at all. Additionally, as a reiteration, the failure rate should be examined prior to the positional and rotational distances. The inability to localize a camera is worse than a successful localization with some error associated with it.

Table 7.3 in section 7.3 shows the number of localization failures for each dataset for each algorithm in one concise table. Our method failed 0.76% of localizations, while the Minimal Scene algorithm failed 4.8 times as many localizations (3.65% failure).

#### 5.3.2.1 Alamo

Table 5.5 and Figure 5.3 detail the specifics of the Alamo dataset. As previously mentioned, Ground Truth and the average of Random (5) will always take a relatively insignificant amount of time and were not measured. Due to Minimal Scene's non-deterministic compression algorithm, the number of points used in the compression will vary slightly off from the desired 20%, while remaining within 1% of the desired value (unless the dataset entirely fails to meet compression *i.e.*, that is the maximum obtainable compression factor for the dataset). As it may be seen in the CDT of rotational errors, our algorithm remained close to the (general) ground truth max, outperforming the other two baseline methods; while also being significantly faster than Minimal Scene.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 5.2632 | 5.2632 | 5.2632 | 5.2632 |
| **Number of Points** | 31249 | 6270 | 6250 | 6249 |
| **Time to Select Points for Compression (s)** | N/A | 146.14 | 5.23 | N/A |

Table 5.5: Alamo validation failure rate and time.

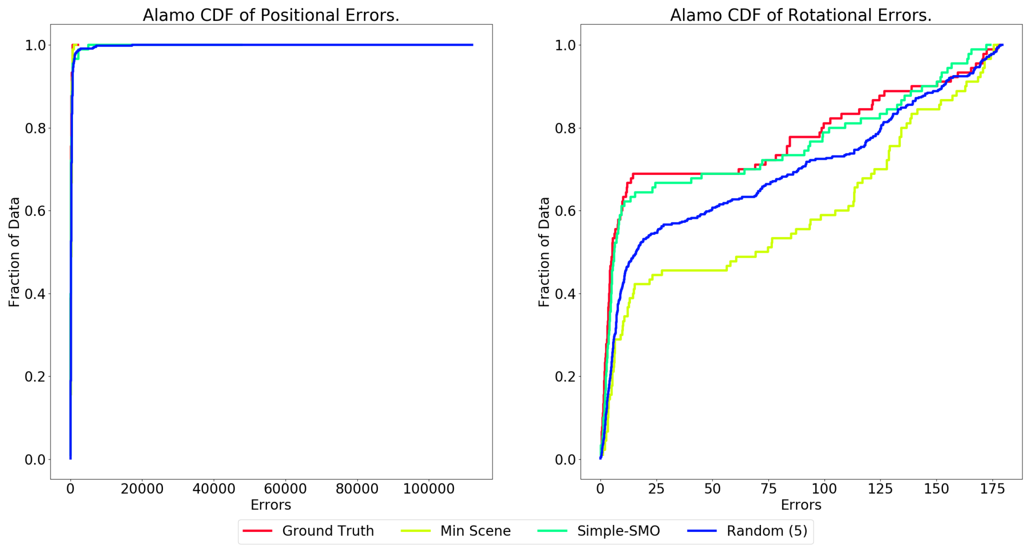


Figure 5.3: Alamo validation error CDF.

#### 5.3.2.2 Ellis Island

For the Ellis Island model, the Minimal Scene algorithm failed to meet desired compression, with it maxing out at 84 points selected. Our algorithm once again outperformed the others on the rotational front, but had an outlier with regards to position. This model was small scale and only had 3 cameras used for validation queries.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 0 | 0 | 0 |
| **Number of Points** | 2009 | 84* | 402 | 401 |
| **Time to Select Points for Compression (s)** | N/A | 1.40 | 0.25 | N/A |

*Failed to meet desired compression

Table 5.6: Ellis Island validation failure rate and time.

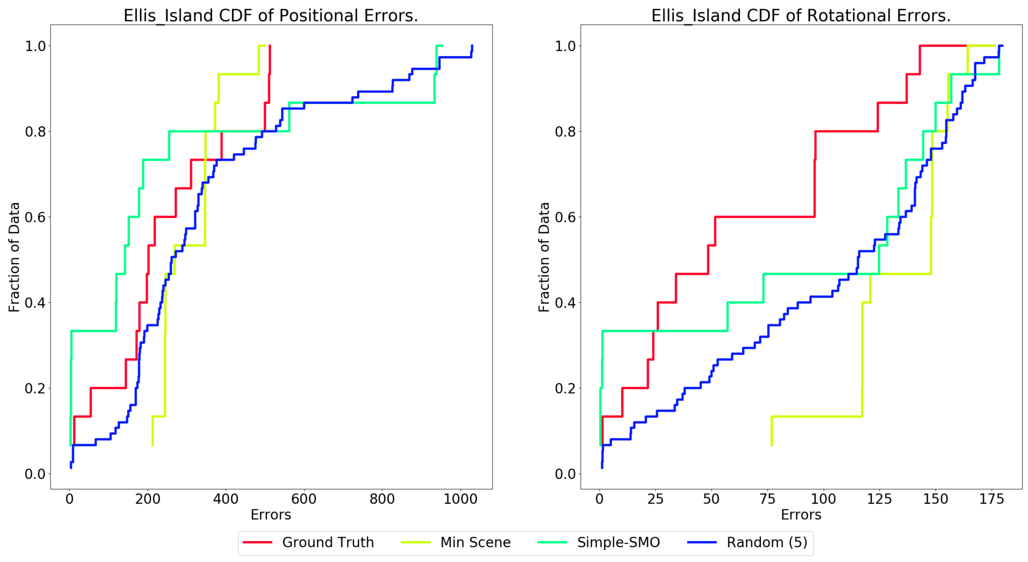


Figure 5.4: Ellis Island validation error CDF.

#### 5.3.2.3 Gendarmenmarkt

With regards to the rotational CDT, the Minimal Scene and Random algorithms were right in line with one another. Our algorithm was more accurate for the majority fraction of data. All methods came together for a similar large error at the final cameras, suggesting those cameras were particularly hard to localize.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 0 | 0 | 0 |
| **Number of Points** | 27683 | 5557 | 5537 | 5536 |
| **Time to Select Points for Compression (s)** | N/A | 68.31 | 1.15 | N/A |

Table 5.7: Gendarmenmarkt validation failure rate and time.

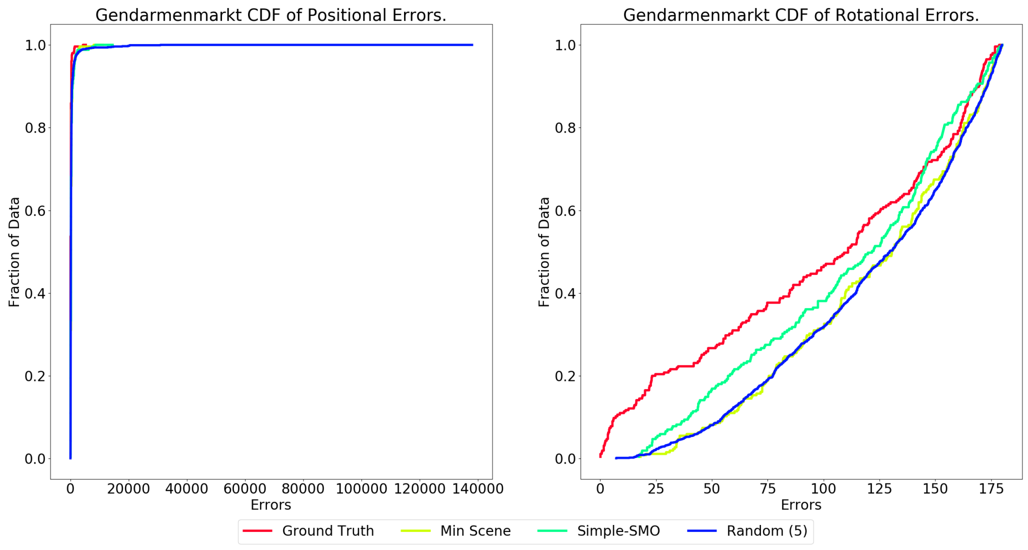


Figure 5.5: Gendarmenmarkt validation error CDF.

#### 5.3.2.4 Madrid Metropolis

Our proposed algorithm for this model both failed less localizations and was less error prone than the other methods. Minimal Scene failed twice as many localizations as ours, and was only slightly better than the randomization method. There were 19 query cameras for this, with random minimization failing 1 or 2, on each of its 5 trials.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 5.2632 | 10.5263 | 5.2632 | 7.3684 |
| **Number of Points** | 21245 | 4260 | 4249 | 4249 |
| **Time to Select Points for Compression (s)** | N/A | 109.97 | 4.81 | N/A |

Table 5.8: Madrid Metropolis validation failure rate and time.

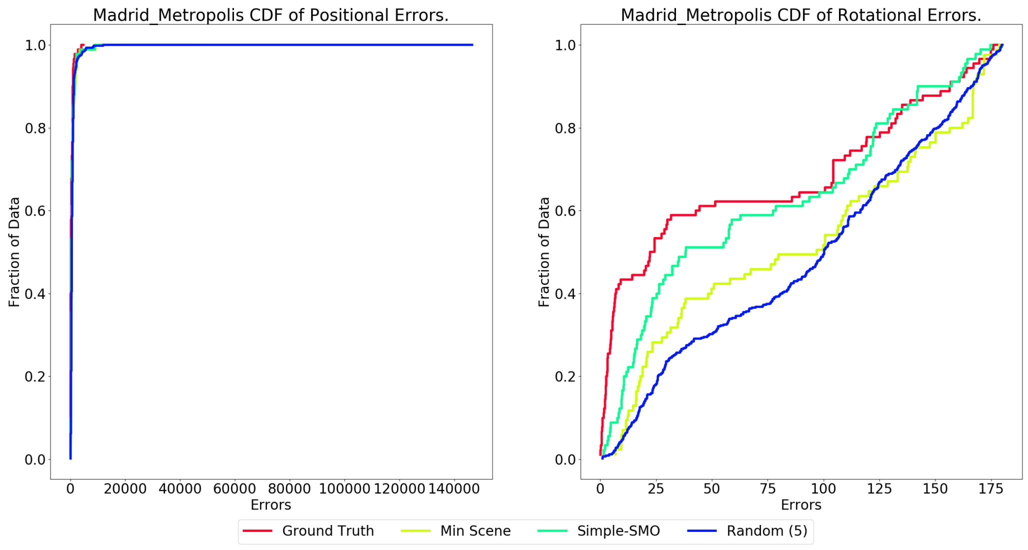


Figure 5.6: Madrid Metropolis validation error CDF.

#### 5.3.2.5 Notre Dame

All models performed similarly on this dataset, most likely due to the proportion of cameras to points (hundreds of points for each camera). Thus, even with compression a large number of points remained to perform localization. For all methods, 75% of the data had a rotation error of less than 2 degrees. There were, however, large outliers w.r.t. the positional distance. The primary difference between our method and Minimal Scene comes from the time aspect. It took our algorithm 15 seconds to select points for minimization, but took Minimal Scene over 5.5 hours.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 0 | 0 | 0 |
| **Number of Points** | 459555 | 91481 | 91911 | 91911 |
| **Time to Select Points for Compression (s)** | N/A | 20112.20 | 15.69 | N/A |

Table 5.9: Notre Dame validation failure rate and time.

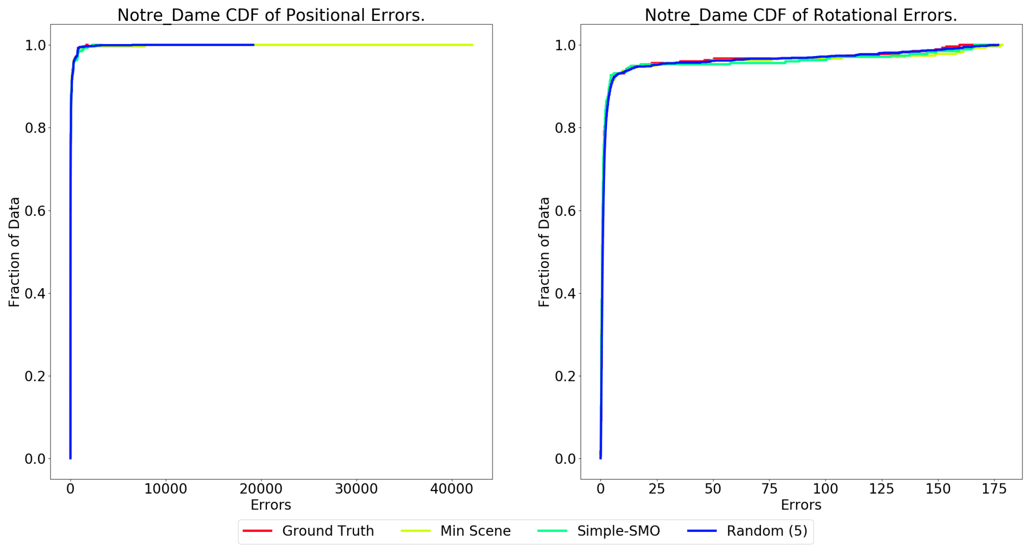


Figure 5.7: Notre Dame validation error CDF.

#### 5.3.2.6 NYC Library

The NYC Library model featured relatively high errors across the board for rotation and position. With only 30% of the data added, the Ground Truth already had an error of 90 degrees. The only difference in models with respect to the CDF comes with a small fraction of the data, where Random is the fastest to accrue error. Additionally, Minimal Scene did not meet compression for this model and likely, as a result of this, features a localization failure rate of 20%.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 20 | 0 | 0 |
| **Number of Points** | 11903 | 1158* | 2381 | 2380 |
| **Time to Select Points for Compression (s)** | N/A | 4.95 | 2.96 | N/A |

*Failed to meet desired compression

Table 5.10: NYC Library validation failure rate and time.

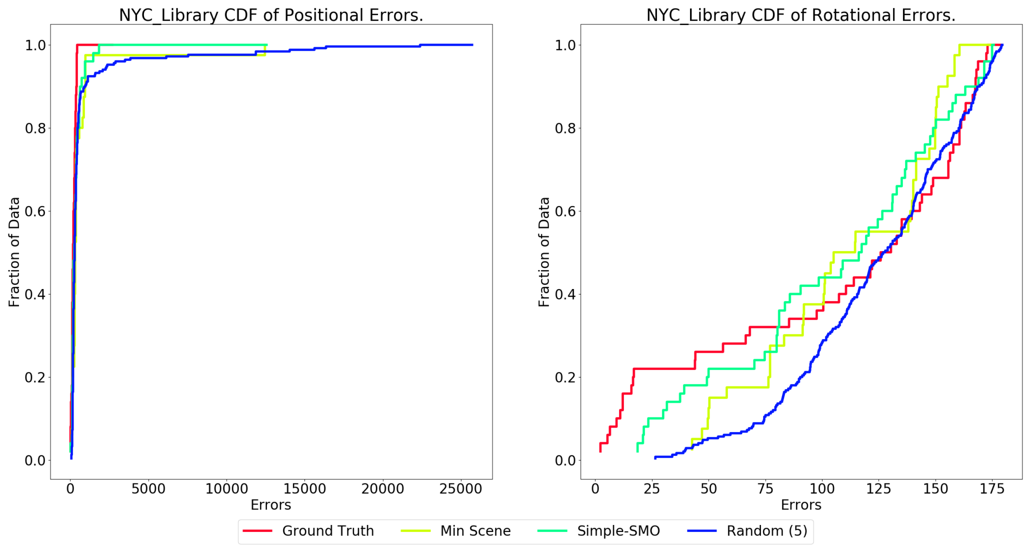


Figure 5.8: NYC Library validation error CDF.

#### 5.3.2.7 Piazza del Popolo

Once again, Minimal Scene was slightly below the desired compression factor, and featured the largest rotational error. Our proposed algorithm performed similarly to the Ground Truth in both distance metrics.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 0 | 0 | 0 |
| **Number of Points** | 7959 | 1203* | 1592 | 1591 |
| **Time to Select Points for Compression (s)** | N/A | 5.03 | 0.57 | N/A |

*Failed to meet desired compression

Table 5.11: Piazza del Popolo validation failure rate and time.

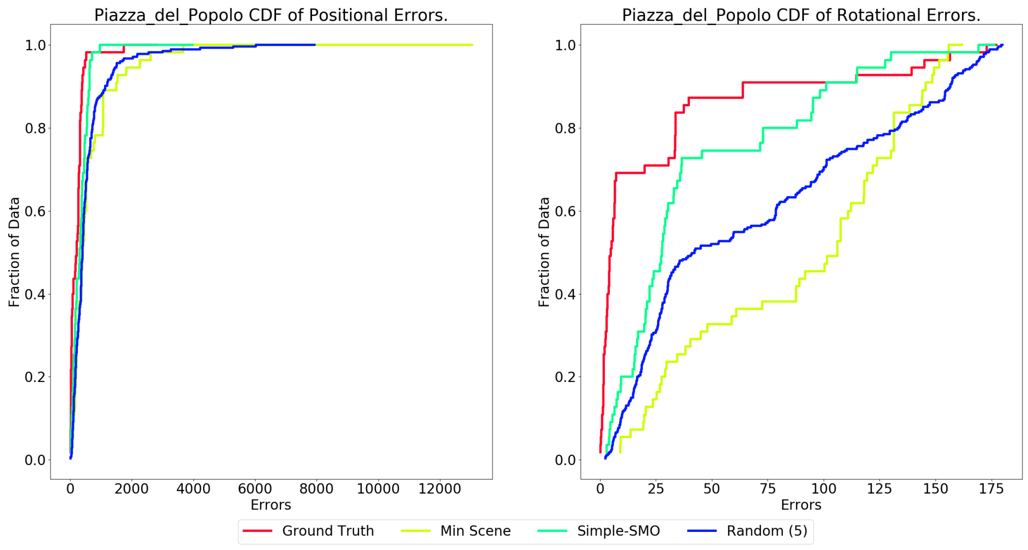


Figure 5.9: Piazza del Popolo validation error CDF.

#### 5.3.2.8 Piccadilly

Again, our proposed algorithm outperformed the other two on the rotational metric, but had an outlier on the positional metric. Minimal Scene performed similar to the Random algorithm, however it failed 3 of the 88 queries, while the other algorithms failed 0.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 3.4091 | 0 | 0 |
| **Number of Points** | 26536 | 5283 | 5308 | 5307 |
| **Time to Select Points for Compression (s)** | N/A | 61.18 | 3.32 | N/A |

Table 5.12: Piccadilly validation failure rate and time.

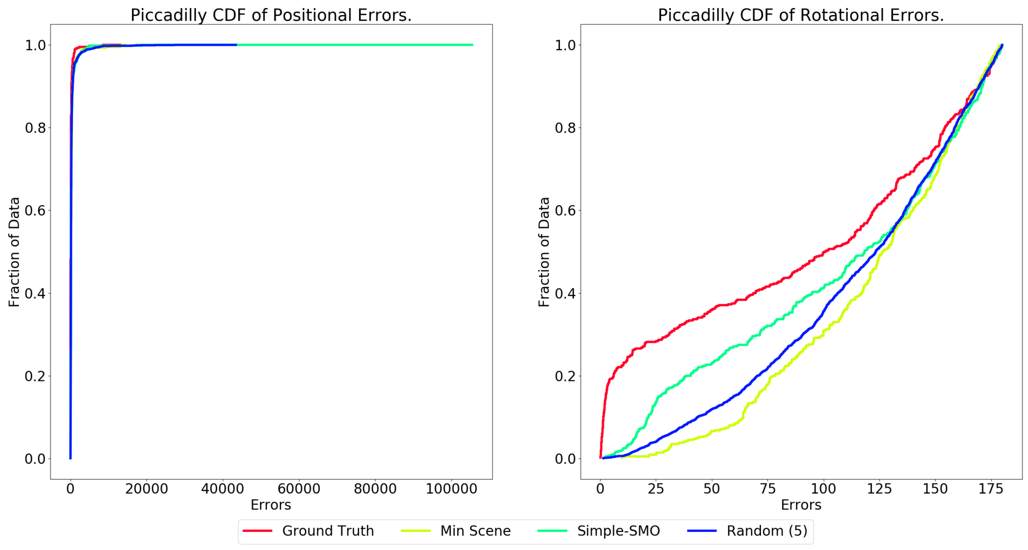


Figure 5.10: Piccadilly validation error CDF.

#### 5.3.2.9 Roman Forum

All algorithms for the Roman Forum had varying failure rates. The Ground Truth failed 2 of the 83 queries, ours failed 3, Minimal Scene failed 4, and Random failed a varying amount. In terms of error, our algorithm stayed close to the Ground Truth for the entirety of the queries, while Random and Minimal Scene were slightly more error prone.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 2.4096 | 4.8193 | 3.6145 | 2.6506 |
| **Number of Points** | 81040 | 16364 | 16208 | 16208 |
| **Time to Select Points for Compression (s)** | N/A | 128.35 | 3.15 | N/A |

Table 5.13: Roman Forum validation failure rate and time.

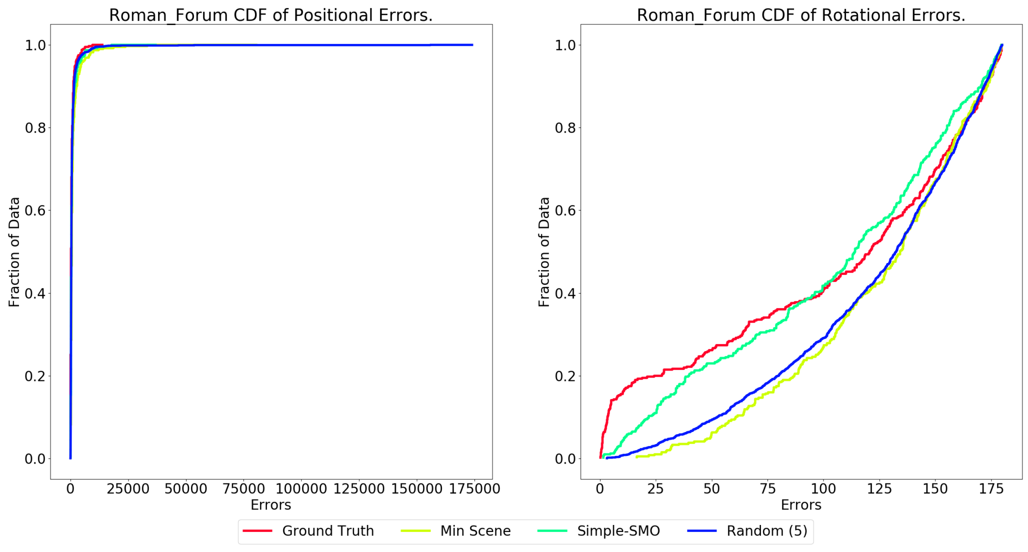


Figure 5.11: Roman Forum validation error CDF.

#### 5.3.2.10 Tower of London

The Random algorithm failed 1 camera on 4 of its 5 attempts, while the other algorithms did not fail any of the localizations. Once again, our algorithm is 2nd only to the Ground Truth, with Minimal Scene and Random performing similar (aside from Random's failed localization).

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 0 | 0 | 3.6364 |
| **Number of Points** | 32119 | 6438 | 6424 | 6423 |
| **Time to Select Points for Compression (s)** | N/A | 45.14 | 4.13 | N/A |

Table 5.14: Tower of London validation failure rate and time.

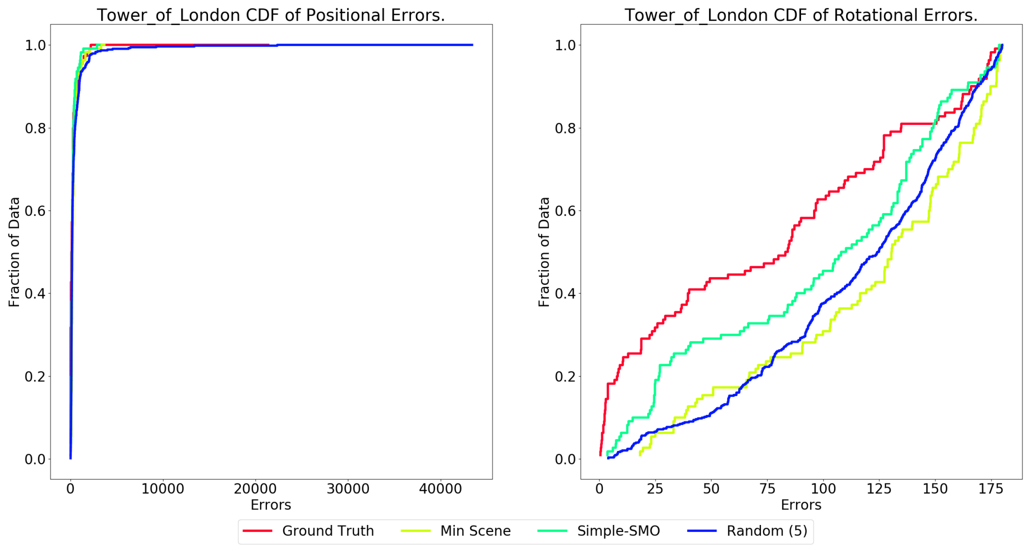


Figure 5.12: Tower of London validation error CDF.

#### 5.3.2.11 Trafalgar

Both the Ground Truth and Simple-SMO algorithms performed similarly on this model, beating both Minimal Scene and Random compression. Considering the Ground Truth failed a localization, but Simple-SMO did not, at least one of the points that camera saw must have been 'bad'. As in, the reconstruction had the camera see a point that it should not have, and the removal of that point ended up *helping* localization.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0.3846 | 0.3846 | 0 | 0.4615 |
| **Number of Points** | 51373 | 10358 | 10275 | 10274 |
| **Time to Select Points for Compression (s)** | N/A | 919.93 | 1.86 | N/A |

Table 5.15: Trafalgar validation failure rate and time.

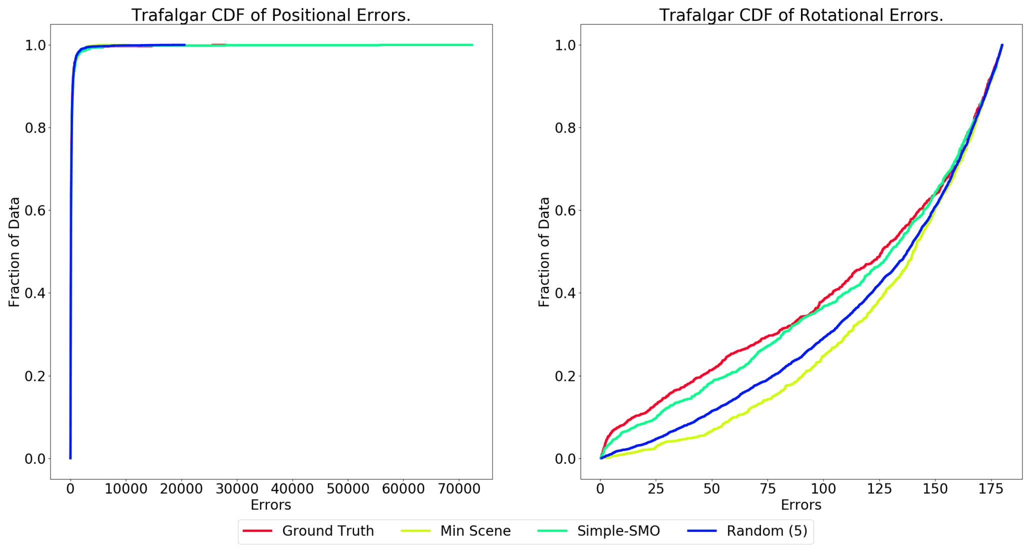


Figure 5.13: Trafalgar validation error CDF.

#### 5.3.2.12 Union Square

This algorithms overall performed poorly on this model. The original reconstruction only contained 696 three-dimensional points (which is not inherently bad), but led to overall difficulty in compression. Minimal Scene was only able to choose 53 points, which ultimately led to it failing all 11 of the query localizations. Random failed approximately 1/4 of the localizations, which could explain why its CDF **appears** better than Ground Truths (there should be a greater fraction of data with high errors that entirely failed). The same reasoning as in Trafalgar may be used for why Ground Truth failed a localization while Simple-SMO did not. Regardless of reasoning, our algorithm did out perform the others on this model.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 9.0909 | 100 | 0 | 25.4545 |
| **Number of Points** | 696 | 53* | 140 | 139 |
| **Time to Select Points for Compression (s)** | N/A | 4.58 | 0.22 | N/A |

*Failed to meet desired compression

Table 5.16: Union Square validation failure rate and time.

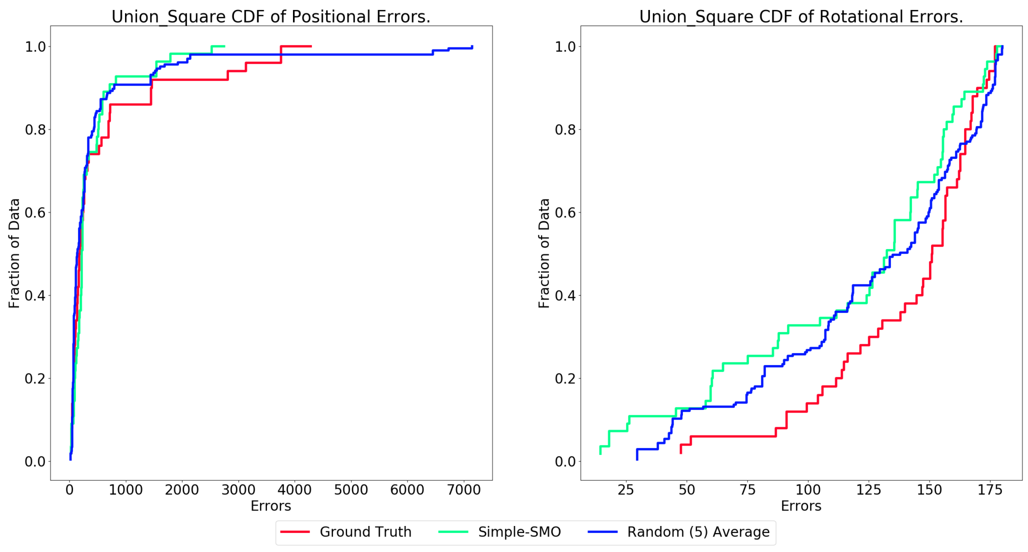


Figure 5.14: Union Square validation error CDF.

### 5.3.2.13 Vienna Cathedral

All datasets had similar positional errors, but varying rotational errors. The Ground Truth had an error of approximately 2 degrees for 40% of the dataset, before succumbing to higher errors. For approximately half the data, Simple-SMO had less error than Minimal Scene and Random. At that 50% data fraction mark, all algorithms had similar errors.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 0 | 0 | 0 |
| **Number of Points** | 37653 | 7568 | 7531 | 7530 |
| **Time to Select Points for Compression (s)** | N/A | 115.97 | 4.54 | N/A |

Table 5.17: Vienna Cathedral validation failure rate and time.

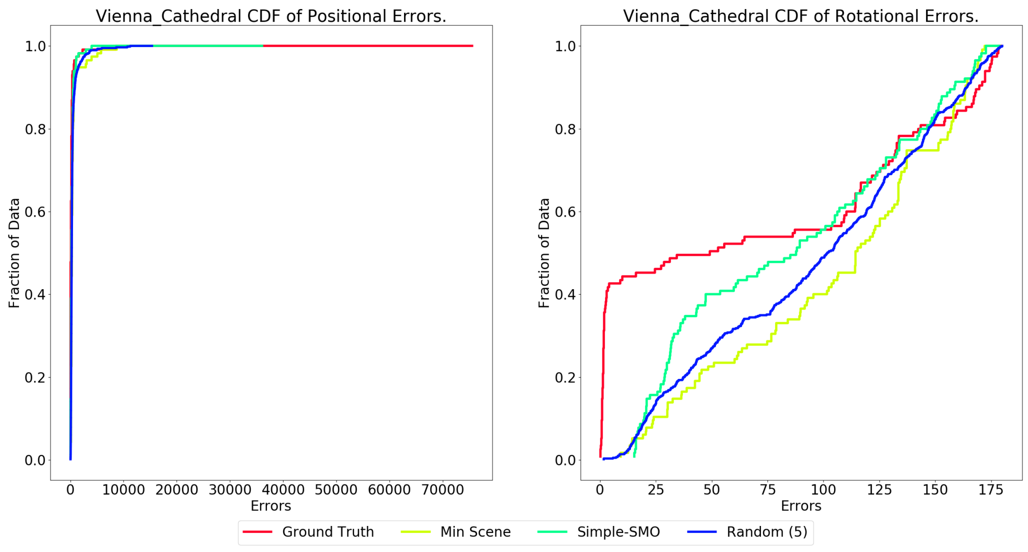


Figure 5.15: Vienna Cathedral validation error CDF.

#### 5.3.2.14 Yorkminster

Once again, Minimal Scene failed to meet desired compression. However, despite this its positional distance was just as good as all other algorithms, but its rotational distance suffered. Our algorithm stayed close to the Ground Truth the entire time, with only one large deviation at the 35% rotation mark. Overall, our algorithm out performed the other two on this model.

| | **Ground Truth** | **Min Scene** | **Simple-SMO** | **Random (5) Average** |
|---|---|---|---|---|
| **Failure Rate (%)** | 0 | 0 | 0 | 0 |
| **Number of Points** | 5076 | 88* | 1016 | 1015 |
| **Time to Select Points for Compression (s)** | N/A | 1.87 | 0.57 | N/A |

*Failed to meet desired compression

Table 5.18: Yorkminster validation failure rate and time.

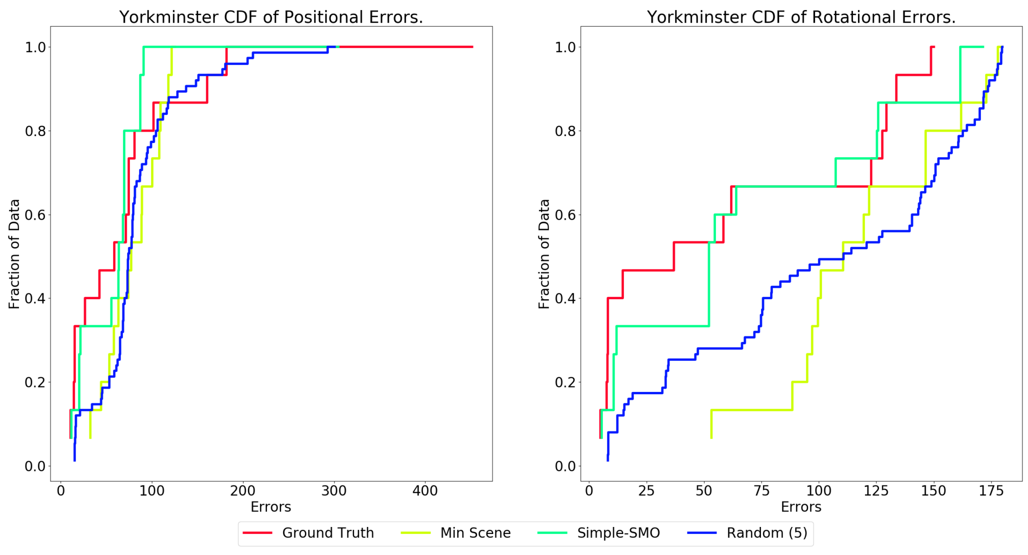


Figure 5.16: Yorkminster validation error CDF.

# Chapter 6

# Conclusions

Our proposed algorithm not only fails fewer localizations compared to the Minimal Scene and Random (5) algorithms, it is also more accurate and faster. The Random algorithm failed 188% more localizations than ours, Minimal Scene failed 480% more, while our algorithm failed no more than the Ground Truth. Minimal Scene was on average 144.48 times slower than our algorithm at selecting the points to localize (see Figure 5.2 for time plot). Across all tested datasets, our algorithm nearly always outperformed Random (5) and Minimal Scene. There were a few instances of the CDT plots which one of the other algorithms 'caught up to' ours (such as nearing 100% data fraction), but at no point was our algorithm significantly surpassed. Numerous CDT's give clear indication of superiority, such as Alamo and Roman Forum. Overall, the algorithms performed the best on Notre Dame and the worst on Union Square, which suggests feature population played a drastic role in the results. (Union Square had 7,742 of 8,456 points removed due to all the cameras that saw those points failing to repopulate features within 8 pixels, while Notre Dame had only 2 of 530,774 removed. It is likely the remainder of Union Square's points were not assigned all features for all cameras).

As per the scoring methods the two frequency based methods performed poorly compared to the other methods. As detailed in subsection 5.3.1, best point frequency failed up to 81.44% of localizations while normalized distance failed only 1.08%. Since normalized distance outperformed only RBF (no distinctiveness), it is safe to assume that the coverage term did in fact improve performance in that case.

## 6.1 Future Work

The next step of continuing this research task is to test the algorithms on lower compression factors than 20%; for example 5%, 1%, or 0.1%. As a greater and greater amount of points are removed, the difficulty to localize increases. Our hypothesis is that our algorithm will outperform by an even greater margin at these highly compressed reconstructions. Another possibility for the future is to devise new scoring metrics, or tune the combination scoring metric (currently set at 50% normalized distance, 50% best point frequency).

# Chapter 7

# Appendices

The following sections contain supplemental information towards the project.

## 7.1 Appendix A - Dataset Info

| **Dataset** | **Converted Recon with Empty Views Removed** | | **Pruned - Remove Query 10%** | | **Pruned - Remove Validation 10%** | |
|---|---|---|---|---|---|---|
| | **Num Images** | **Num Points** | **Num Images** | **Num Points** | **Num Images** | **Num Points** |
| **Alamo** | 192 | 32962 | 173 | 30307 | 173 | 31249 |
| **Ellis Island** | 38 | 2230 | 35 | 2164 | 35 | 2009 |
| **Gendarmenmarkt** | 519 | 30918 | 468 | 27872 | 468 | 27683 |
| **Madrid Metropolis** | 192 | 22808 | 173 | 20231 | 173 | 21245 |
| **Notre Dame** | 553 | 530774 | 498 | 460851 | 498 | 459555 |
| **NYC Library** | 103 | 12514 | 93 | 10420 | 93 | 11903 |
| **Piazza del Popolo** | 111 | 8389 | 100 | 7552 | 100 | 7959 |
| **Piccadilly** | 880 | 28274 | 792 | 26623 | 792 | 26536 |
| **Roman Forum** | 831 | 86837 | 748 | 80178 | 748 | 81040 |
| **Tower of London** | 228 | 33944 | 206 | 32387 | 206 | 32119 |
| **Trafalgar** | 2604 | 54486 | 2344 | 51383 | 2344 | 51373 |
| **Union Square** | 111 | 714 | 100 | 671 | 100 | 696 |
| **Vienna Cathedral** | 233 | 42197 | 210 | 39013 | 210 | 37653 |
| **Yorkminster** | 38 | 5096 | 35 | 4508 | 35 | 5076 |

Table 7.1: Dataset information for converted, query, and validation reconstructions.

## 7.2 Appendix B - Query Results

| **Dataset** | **Num. Query Images** | **Average Number of Query Cameras Failed to Localize** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Distance Non-Norm** | **Distance Norm** | **Freq. Num. Images** | **Freq. Best Point.** | **Combo** | **Only RBF** |
| **Alamo** | 19 | 0.00 | 0.00 | 14.10 | 14.00 | 0.00 | 0.00 |
| **Ellis_Island** | 3 | 0.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| **Gendarmenmarkt** | 51 | 0.00 | 0.00 | 12.00 | 42.40 | 0.00 | 0.00 |
| **Madrid_Metropolis** | 19 | 0.00 | 0.70 | 12.30 | 13.00 | 0.10 | 0.00 |
| **Notre_Dame** | 55 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **NYC_Library** | 10 | 0.00 | 0.00 | 7.80 | 8.30 | 0.00 | 0.00 |
| **Piazza_del_Popolo** | 11 | 0.00 | 0.00 | 7.00 | 6.90 | 0.00 | 0.00 |
| **Piccadilly** | 88 | 1.10 | 1.00 | 72.60 | 88.00 | 0.70 | 1.11 |
| **Roman_Forum** | 83 | 2.80 | 3.00 | 48.00 | 78.90 | 0.70 | 2.68 |
| **Tower_of_London** | 22 | 0.80 | 0.40 | 18.60 | 21.40 | 1.30 | 0.74 |
| **Trafalgar** | 260 | 1.00 | 1.90 | 7.70 | 228.70 | 0.20 | 1.11 |
| **Union_Square** | 11 | 1.40 | 0.10 | 3.30 | 10.30 | 8.10 | 1.47 |
| **Vienna_Cathedral** | 23 | 0.70 | 0.00 | 22.60 | 23.00 | 0.00 | 0.53 |
| **Yorkminster** | 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **TOTAL (COUNT)** | 658 | 7.80 | 7.10 | 227.00 | 535.90 | 11.10 | 7.63 |
| **TOTAL (PERCENT)** | 100 | 1.19 | 1.08 | 34.50 | 81.44 | 1.69 | 1.16 |

Table 7.2: Number of failed cameras by scoring method by dataset.

## 7.3 Appendix C - Validation Results

| **Dataset** | **Num. Query Images** | **Average Number of Query Cameras Failed to Localize** | | | |
|---|---|---|---|---|---|
| | | **Ground Truth** | **Simple-SMO** | **Minimal Scene** | **Random 5** |
| **Alamo** | 19 | 1.00 | 1.00 | 1.00 | 1.00 |
| **Ellis_Island** | 3 | 0.00 | 0.00 | 0.00 | 0.00 |
| **Gendarmenmarkt** | 51 | 0.00 | 0.00 | 0.00 | 0.00 |
| **Madrid_Metropolis** | 19 | 1.00 | 1.00 | 2.00 | 1.40 |
| **Notre_Dame** | 55 | 0.00 | 0.00 | 0.00 | 0.00 |
| **NYC_Library** | 10 | 0.00 | 0.00 | 2.00 | 0.00 |
| **Piazza_del_Popolo** | 11 | 0.00 | 0.00 | 0.00 | 0.00 |
| **Piccadilly** | 88 | 0.00 | 0.00 | 3.00 | 0.00 |
| **Roman_Forum** | 83 | 2.00 | 3.00 | 4.00 | 2.20 |
| **Tower_of_London** | 22 | 0.00 | 0.00 | 0.00 | 0.80 |
| **Trafalgar** | 260 | 1.00 | 0.00 | 1.00 | 1.20 |
| **Union_Square** | 11 | 1.00 | 0.00 | 11.00 | 2.80 |
| **Vienna_Cathedral** | 23 | 0.00 | 0.00 | 0.00 | 0.00 |
| **Yorkminster** | 3 | 0.00 | 0.00 | 0.00 | 0.00 |
| **TOTAL (COUNT)** | 658 | 6.00 | 5.00 | 24.00 | 9.40 |
| **TOTAL (PERCENT)** | 100 | 0.91 | 0.76 | 3.65 | 1.43 |

Table 7.3: Number of failed cameras by algorithm by dataset.

# References

[1] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004. 21

[2] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler. Hybrid camera pose estimation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018. 1

[3] F. Camposeco, T. Sattler, and M. Pollefeys. Minimal solvers for generalized pose and scale estimation from two rays and one point. In *Proc. of the European Conf. on Computer Vision*, 2016. 1

[4] S. Cao and N. Snavely. Minimal scene descriptions from structure from motion models. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2014. 2, 5, 6, 18, 35

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 2

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007. 1

[7] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 8

[8] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. 7

[9] C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, P. Furgale, T. Sattler, and M. Pollefeys. 3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing*, 68:14–27, 2017. 1

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, New York, NY, USA, 2 edition, 2003. vi, 9

[11] G. Hee Lee, F. Faundorfer, and M. Pollefeys. Motion estimation for self-driving cars with a generalized camera. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2013. 1

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 7

[13] A. Kendall, R. Cipolla, et al. Geometric loss functions for camera pose regression with deep learning. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017. 7

[14] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proc. of the IEEE International Conf. on Computer Vision*, 2015. 7

[15] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *Proc. of the European Conf. on Computer Vision*, 2010. 2, 5

[16] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the IEEE International Conf. on Computer vision*, 1999. 1, 5, 10

[17] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-dof localization on mobile devices. In *Proc. of the European Conf. on computer vision*, 2014. 1

[18] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[19] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998. 2, 7, 22, 24

[20] J. C. Platt, J. Shawe-Taylor, A. J. Smola, R. C. Williamson, et al. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001. 26

[21] T. Sattler, M. Havlena, F. Radenovic, K. Schindler, and M. Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition. In *Proc. of the IEEE International Conf. on Computer Vision*, 2015.

[22] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *Proc. of the IEEE International Conf. on Computer Vision*, 2011. 2

[23] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla. Are large-scale 3d models really necessary for accurate visual localization? In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017. 1

[24] B. Schölkopf, A. J. Smola, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002. 2, 19, 21

[25] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Proc. of the Conf. on Advances in Neural Information Processing Systems*, pages 582–588, 2000. 2, 7, 21

[26] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 835–846, New York, NY, USA, 2006. ACM. 11

[27] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, July 2006.

[28] H. Soo Park, Y. Wang, E. Nurvitadhi, J. C. Hoe, Y. Sheikh, and M. Chen. 3d point cloud reduction using mixed-integer quadratic programming. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, 2013. 2, 6

[29] C. Sweeney. Theia multiview geometry library: Tutorial & reference. `http://theia-sfm.org`. 11

[30] C. Sweeney, J. Flynn, B. Nuernberger, M. Turk, and T. Höllerer. Efficient computation of absolute pose for gravity-aware augmented reality. In *Proc. of the IEEE International Symposium on Mixed and Augmented Reality*, 2015. 1

[31] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. gdls: A scalable solution to the generalized pose and scale problem. In *Proc. of the European Conf. on Computer Vision*, 2014. 1

[32] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. Large scale sfm with the distributed camera model. In *Proc. of the IEEE International Conf. on 3D Vision*, 2016. 1

[33] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *Proc. of the IEEE International Conf. on Computer Vision*, 2017. 7

[34] K. Wilson and N. Snavely. Robust global translations with 1dsfm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 14, 28

[35] W. Zhang and J. Kosecka. Image based localization in urban environments. In *3DPVT*, volume 6, pages 33–40. Citeseer, 2006.