TOWARD EFFICIENT AND ROBUST LARGE-SCALE STRUCTURE-FROM-MOTION SYSTEMS

Jared S. Heinly

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2015

ABSTRACT

Jared S. Heinly: Toward Efficient and Robust Large-Scale Structure-from-Motion Systems
(Under the direction of Jan-Michael Frahm and Enrique Dunn)

The ever-increasing number of images that are uploaded and shared on the Internet has recently been leveraged by computer vision researchers to extract 3D information about the content seen in these images. One key mechanism to extract this information is structure-from-motion, which is the process of recovering the 3D geometry (structure) of a scene via a set of images from different viewpoints (camera motion). However, when dealing with crowdsourced datasets comprised of tens or hundreds of millions of images, the magnitude and diversity of the imagery poses challenges such as robustness, scalability, completeness, and correctness for existing structure-from-motion systems. This dissertation focuses on these challenges and demonstrates practical methods to address the problems of data association and verification within structure-from-motion systems.

Data association within structure-from-motion systems consists of the discovery of pairwise image overlap within the input dataset. In order to perform this discovery, previous systems assumed that information about every image in the input dataset could be stored in memory, which is prohibitive for large-scale photo collections. To address this issue, we propose a novel streaming-based framework for the discovery of related sets of images, and demonstrate our approach on a crowdsourced dataset containing 100 million images from all around the world. Results illustrate that our streaming-based approach does not compromise model completeness, but achieves unprecedented levels of efficiency and scalability.

The verification of individual data associations is difficult to perform during the process of structure-from-motion, as standard methods have limited scope when determining image overlap. Therefore, it is possible for erroneous associations to form, especially when there are symmetric, repetitive, or duplicate structures which can be incorrectly associated with each other. The consequences of these errors are incorrectly placed cameras and scene geometry within the 3D reconstruction. We present two methods that can detect these local inconsistencies and successfully resolve them into a globally consistent 3D model. In our evaluation, we show that our techniques are efficient, are robust to a variety of scenes, and outperform existing approaches.

ACKNOWLEDGEMENTS

I would like to thank my advisors, Jan-Michael Frahm and Enrique Dunn, for their input and guidance.

I would also like to thank my committee members, Alexander C. Berg, Marc Niethammer, and Sameer Agarwal, for their feedback and advice.

Additionally, I would like to thank my labmates, as their company and discussion made my time more fruitful and enjoyable: Akash Bapat, Sangwoo Cho, Pierre Fite-Georgel, Jeremy Graham, Yunchao Gong, Rohit Gupta, Shubham Gupta, Xufeng Han, Junpyo Hong, Yi-Hung Jen, Dinghuang Ji, Alex Keng, Hadi Kiapour, Hyo Jin Kim, Wei Liu, Jie Lu, Vicente Ordóñez-Román, David Perra, True Price, Rahul Raguram, Patrick Reynolds, Johannes Schönberger, Meng Tan, Joseph Tighe, Sirion Vittayakorn, Ke Wang, Yilin Wang, Yi Xu, Hongsheng Yang, and Enliang Zheng.

I would like to thank my parents, who encouraged me to explore, create, tinker, and learn.

Finally, I would like to thank my wife, Kelly, for her continued support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CCG	Camera connectivity graph
lcc	Local clustering coefficient
LS-SfM	Large-scale structure-from-motion
MST	Minimum spanning tree
PCOG	Point co-occurrence graph
RANSAC	Random sample consensus
SfM	Structure-from-motion
SIFT	Scale invariant feature transform (Lowe, 2004)
WS-SfM	World-scale structure-from-motion

LIST OF SYMBOLS

C	3D point co-occurence matrix
C_{ij}	Boolean indicating whether 3D points i and j were observed in the same camera
M	Number of images
m	Number of cameras
n	Number of points
O	Storage of 3D point observations for each camera
O_i	Set of 3D points observed by camera i
O_{ij}	Boolean indicating whether 3D point j is observed in camera i
T_i	Center position of camera i

CHAPTER 1: INTRODUCTION

As humans, visual content is one of the richest ways in which we can experience the world around us. As such, throughout our history, we have striven to capture and record our visual environment, whether through drawings in caves, a cartographic map carved in wood, or an oil painting on canvas. Nowadays, with the proliferation of digital cameras and camera-enabled phones, there is an incredible amount of imagery being captured and uploaded to the Internet. For instance, it is estimated that over 1.8 billion images and 430,000 hours of video are uploaded to Internet media storage sites daily (Meeker, 2014; YouTube, 2014).

Currently, much of this content is indexed and searched for using textual means. Users provide various query terms, and depending on the titles and tags associated with the content, a limited set of search results is returned. However, this content can also be indexed visually, where users can use an image as a query, and have the retrieval system return a set of images that are similar according to some visual mechanism.

An additional way to index and represent the visual content of the world is through geometric constraints and 3D geometry. Here, imagery is associated and organized based on the fact that it depicts the same underlying physical object or scene. With this constraint, a 3D model of an object or scene can be reconstructed, and the positions of the cameras that imaged it can be recovered. One of the primary methods to recover this geometry is termed structure-from-motion (SfM), in which the motion of a camera allows both its position and the underlying scene's structure to be determined.

In order to perform structure-from-motion, the relationships between the input images must first be determined. When the imagery is obtained from a video sequence, the determination of the relationships becomes much simpler as one can make assumptions about the valid range of motion that can occur between successive video frames. However, when the imagery comes from Internet photo collections, there is typically no order to the images, and thus one must devise ways to handle these unordered collections. Many applications of SfM have been made to both ordered and unordered imagery domains, but in this work, we focus primarily on applications that deal with unordered, crowdsourced collections of images (such as those found on Flickr[1] or Panoramio[2]).

While structure-from-motion has been studied heavily and improved upon over the years (Snavely et al., 2006; Agarwal et al., 2009; Frahm et al., 2010; Crandall et al., 2011; Klingner et al., 2013; Wu, 2013; Wilson and Snavely, 2014), it still remains a viable research area, and one that motivates many useful applications. For instance, when applied to community-driven Internet photo collections, SfM provides one mechanism to organize and browse these photos (Schaffalitzky and Zisserman, 2002; Snavely et al., 2006, 2007, 2008a; Raguram et al., 2011). For commercially-captured datasets, SfM can be used for 3D mapping data or navigation (Klingner et al., 2013; Frahm et al., 2013; Ardeshir et al., 2014). In these applications mentioned so far, SfM is used to capture the static (non-moving) parts of the environment. However, SfM can be used to enable the capture of temporal changes within an environment (Park et al., 2010; Matzen and Snavely, 2014; Ji et al., 2014; Zheng et al., 2014b). Furthermore, SfM enables many additional applications and research topics, such as simultaneous localization and mapping (SLAM) (Klein and Murray, 2007; Davison et al., 2007), dense reconstruction (Goesele et al., 2007; Furukawa et al., 2010; Furukawa and Ponce, 2010; Zheng et al., 2012), image localization (Sattler et al., 2011; Li et al., 2012; Bansal et al., 2012; Cao and Snavely, 2013), and many more.

In all of the aforementioned applications, efficiency in processing remains an important goal; specifically, it is important for methods to have the ability to handle large amounts of imagery. Two motivating factors of this are the now-ubiquitous presence of camera devices and the ever-increasing number of photos and videos that are uploaded to the Internet (Meeker, 2014). In order to find the relationships between the images in a given dataset of size m, naïve methods would

[1] https://www.flickr.com/

[2] http://www.panoramio.com/

perform exhaustive pairwise verifications, yielding a computational complexity of $O(m^2)$. This quickly becomes intractable for datasets of even a few thousand images. To combat this, many methods create a database storing summary information for each image, and then perform retrieval against this database. However, this requires $O(m)$ storage, which in itself becomes problematic when dealing with datasets of millions of images. Therefore, depending on the target dataset size, applications must make careful design decisions to make their processing tractable.

In addition to issues arising from the scale of the dataset, there are also challenges related to content and organization of the imagery in the dataset. For instance, in an Internet photo collection, the images are typically unordered, so there are often no priors about which images will be related. Also, in many cases, a high fraction of an Internet photo collection's images will be unusable for reconstruction as they represent unique or transient scenes, and thus lack the multiple observations necessary for SfM (Frahm et al., 2010; Heinly et al., 2015). Therefore, in order to determine the relationships between images in a given dataset, methods must discover similar images, but be robust to a large presence of unrelated imagery. Furthermore, the dataset's imagery may contain watermarks, text, or other commonly-occurring patterns which may confuse the discovery of relationships between the images. In addition to this, the imagery may view scenes that contain repetitive or duplicate structure, leading to ambiguous viewpoint determination or reconstruction artifacts. Given the presence of these challenges, applications must take care when discovering the relationships between images, or provide a method of disambiguation when ambiguities arise.

Summarizing these challenges is the problem of data association, which, in the context of structure-from-motion, is the issue of determining the sets of images that are related to each other through geometric means. This work focuses squarely on this problem in that it tackles both efficient discovery of related images, as well as robust disambiguation when misregistrations occur.

1.1 Thesis Statement

The efficiency of large-scale structure-from-motion systems can be increased by employing a streaming paradigm for data association, and their robustness improved by post-processing the structure-from-motion result to remove artifacts caused by duplicate structure within a scene.

1.2 Outline of Contributions

This dissertation contains several significant contributions that advanced the state of the art in large-scale structure-from-motion systems, and builds upon published works which support these claims (Heinly et al., 2015, 2014a,b). These contributions include:

Streaming Paradigm for Structure-from-Motion Preprocessing: We propose a new paradigm for preparing large-scale datasets for use in structure-from-motion. This paradigm is a streaming framework in which images are read only once in a sequential fashion from disk, and useful information about those images is retained in memory for only as long as it is useful. The outputs of this processing are sets of connected components (related images) that are ready for incremental reconstruction. This work is described in Heinly et al. (2015), and detailed in Chapter 3.

Structure-from-Motion Post-Processing to Correct for Duplicate Structure: We propose novel methods to post-process a structure-from-motion result to detect and correct for the presence of artifacts caused by duplicate structure within the imaged scene. This work is described in Heinly et al. (2014a) and detailed in Chapter 4.

Efficient Processing for Duplicate Structure Correction: Given the insights gained from the prior contribution, we propose a more efficient post-processing method to perform disambiguation when errors arise due to duplicate structure. This work is described in Heinly et al. (2014b) and detailed in Chapter 5.

Each of these contributions addresses the issue of data association within the context of structure-from-motion. The streaming paradigm of Chapter 3 shows how the complex relationships between images can be efficiently discovered for world-scale datasets, demonstrating results on a 100 million image dataset (Shamma, 2014; Thomee et al., 2015). The disambiguation method of Chapter 4 shows how misregistrations due to duplicate scene structure can be robustly detected and corrected. Then, the alternate disambiguation method of Chapter 5 shows how the same process can be modified with a focus on efficiency and runtime. Finally, Chapter 6 reflects on these contributions, and proposes both practical modifications as well as research directions to continue to advance the state of the art in large-scale data association for structure-from-motion.

CHAPTER 2: RELATED WORK

Structure-from-motion and its applications have been studied in much of the computer vision literature. The following sections provide an overview of standard techniques in structure-from-motion as well as recent advances in the field.

2.1 Overview of a Structure-from-Motion System

When dealing with a large-scale structure-from-motion system, it is important to understand the role of each component in the pipeline.

Generally speaking, a structure-from-motion pipeline goal is to discover relationships, whether they are sets of images sharing common feature observations, or images that have been assigned relative 3D poses. In the beginning of the pipeline, local, keypoint-based features are extracted from the images. These features are then used to find candidate feature matches between pairs of images, which are then verified using geometric constraints. By employing this geometric verification, verified image pairs can be combined into larger sets of related images (connected components), which are then passed to structure-from-motion in order to be reconstructed. This reconstruction process recovers not only the relative camera geometry (relative camera poses, which include orientation and position information), but also a set of 3D points which correspond to the shared 2D point observations between the images.

The following sections provide a more detailed overview of the main components that comprise a structure-from-motion system.

2.1.1 Input

This work focuses on unorganized, crowdsourced, photo collections. Therefore, the input to the structure-from-motion system is simply a set of images.

Certain prior information about these images may be known. For instance, if the image was downloaded from an online photo-sharing website, there may be textual information associated with the image, such as a title or keyword tags. Additionally, the photo may have been geotagged, indicating the geospatial location at which the photo was captured. The photo may itself contain meta information (such as EXIF data for JPEG images). This meta information may indicate the date and time the photo was taken, or the type of camera, lens, focal length, exposure, or other settings used to capture the image.

Many of these types of prior information can be used to derive relationships between images. For example, images that all contain a common landmark name in their title or tags may depict that common landmark. More promising, however, are images with a similar geotagged location. These photos may all have been taken from a common position viewing a common scene. Finally, images taken around a similar time with the same camera and lens configuration may be from the same photographer, yielding a higher probability that they show the same scene.

Unfortunately, this additional prior information about the images may be noisy or nonexistent. For instance, only 15-25% of the images that we download for large-scale photo collections (millions of images) typically have geotags provided. Additionally, these geotags may be inaccurate by tens or hundreds of meters (Zamir et al., 2014), especially when images are taken indoors. In the case of textual tags, the same word may refer to multiple locations, a word may be generic and not refer to a specific place or content, or a word may have different spellings or misspellings (Zhu et al., 2010). Therefore, while this additional prior information can provide some indication as to the relationship between a subset of the images, we ignore it for our purposes and focus purely on visual information provided in the images.

2.1.2 Feature Extraction

Given a set of input images, we now seek to construct a representation for the content contained within each image. The typical strategy for this is to compute a set of local features for each image, where a feature consists of two parts: the keypoint and the descriptor. The keypoint is a 2D location within the image, and the descriptor is a numeric representation of the visual content surrounding that location.

In order to determine keypoint locations, a feature detector is employed, which seeks out salient points within an image. In other words, a detector should identify specific locations within an image which could be easily and repeatably identified in other images viewing that same location in the scene. Additionally, a detector may extract local scale and orientation information for each detected keypoint in order to allow for changes in the field-of-view, resolution, and orientation of the overall image, as well as minor changes in viewpoint. To attempt to handle larger changes in viewpoint, a detector may also incorporate affine information, in which the local elliptical shape of the 2D image location is estimated (Matas et al., 2002; Mikolajczyk and Schmid, 2004).

Two common classes of feature detectors are corner (Harris and Stephens, 1988; Rosten and Drummond, 2006) and blob-style (Lowe, 2004; Bay et al., 2008) detectors. A corner detector attempts to find locations where the gradient in the image is at a local maximum with respect to two orthogonal directions (for efficiency, these are typically chosen to be the x and y axes of the image, which correspond to its rows and columns of pixels). In contrast, a blob-style detector attempts to find circular regions or "blobs" that have high contrast with their surroundings (such as a dark circle on a light background).

Once a detector has identified the set of keypoint locations for each image, a feature descriptor extracts their local representations. At a minimum, a descriptor could be the raw pixel values from a patch surrounding the keypoint. However, to be more robust to changes in illumination, image quality, viewpoint, *etc.*, descriptors may compute and describe the histograms, gradients, or intensity change of the pixels around the keypoint. In order to incorporate the optional scale, orientation, or affine information from the detector, the pixel patch can be warped to a canonical configuration before the descriptor is extracted. Alternatively, the sampling pattern employed by the descriptor can be warped so that a copy of the underlying 2D pixel data does not need to be made.

When extracting the set of descriptors for an image, a descriptor may employ a binary (Calonder et al., 2010; Rublee et al., 2011; Leutenegger et al., 2011; Alahi et al., 2012) or real-valued representation (Lowe, 2004; Bay et al., 2008; Tola et al., 2010). Typically, a binary representation is formed by the concatenation of a sequence of pairwise pixel intensity comparisons, whereas a real-valued representation can represent the values in a histogram or the gradients in the image.

Many of the above detectors and descriptors have manually created patterns, in that the sampling pattern that they employ has been chosen by the implementer. However, several recent works have looked at applying machine learning to the process, in order to create features that are more robust and discriminative (Rosten et al., 2010a; Brown et al., 2010; Rublee et al., 2011; Trzcinski and Lepetit, 2012). Here, these approaches typically leverage a large training set of known correspondences between images, and use these to train their final representations.

2.1.3 Pairwise Feature Matching

Given a set of features (each with its own descriptor) for each image, the next step is to propose candidate pairs of matching features between a pair of images. This is typically accomplished by evaluating a distance function between each candidate pair of descriptors (one from each image), and selecting those pairs of descriptors with the highest similarity. For real-valued descriptors, the Euclidean or cosine (dot-product) distance are popular choices, whereas binary descriptors employ the efficient Hamming distance.

Given a set of distances between the descriptors, each descriptor in the first image is paired with its best matching descriptor from the second set. This forms a set of candidate feature matches. However, many of the candidate matches will typically be erroneous, and as such, one or more filtering stages are applied. For instance, one filtering criteria enforces that a pair be selected only if its descriptors are each other's best matches. Another criteria enforces that the ratio between the first and second best matching descriptors is below a certain threshold (Lowe, 2004). Finally, a third criteria enforces a fixed threshold on the maximum dissimilarity between the descriptors. These filtering criteria can be applied in any combination, as well as with others, with the goal of reducing the fraction of erroneous candidate matches from this stage.

As opposed to the above strategy, certain applications may store a list of candidate matches per feature, or may perform feature matching within a single image prior to matching to other images. This is typically useful in attempting to detect repetition or symmetry within images (Jiang et al., 2011; Ceylan et al., 2014; Kushnir and Shimshoni, 2014), though it is typically avoided because of the additional computational burden.

As in feature extraction, several recent works have looked at applying machine learning to feature matching (Žbontar and LeCun, 2015; Han et al., 2015). Here, a classifier or convolutional neural network is trained to recognize when two local image representations are significantly similar. While these approaches can yield accurate results, they can be computationally limiting when efficiency is key.

2.1.4 Pairwise Geometric Verification

Given a set of candidate feature matches between a pair of images, we now seek to identify a potential subset of those matches that conforms to a consistent camera transformation. Here, the intuition is that if the two images depict a common object or scene, then there will be a subset of the candidate matches that lie within the common content and are consistent with the camera motion between the two images.

In order to conclude that the two images are in fact geometrically related, we must define the type of camera transformation that we seek. In the simplest of cases, a 2D similarity or affine transformation may be used. However, these will quickly fail when there is any non-trivial camera motion (such as sideways camera translation, especially when the scene consists of multiple objects at different depths). Another option is to use a homography, which maps any quadrilateral to any other quadrilateral, and can be used in the cases of pure camera rotation (the camera did not translate) or a planar scene. However, to avoid these restrictions, a transformation that is based on epipolar geometry must be employed.

An image is a 2D projection of 3D data, and therefore there is lost information. Specifically, the depth of the scene at each point in the image is not recorded. Because of this loss of information, there is an ambiguity when attempting to construct a direct mapping between the pixel locations of two images (assuming arbitrary camera pose and arbitrary scene content). Epipolar geometry is precisely the mechanism which allows for this ambiguity, but still represents the known constraints on the relative camera transformation. To model this constraint, instead of a direct pixel-to-pixel mapping, epipolar geometry uses a pixel-to-ray mapping, where an observed pixel in one image maps to a ray in the other, where the corresponding scene element could lie anywhere along that ray, depending on its depth.

The two most popular ways to describe this epipolar relationship are the fundamental and essential matrices. Both are 3×3 matrices, but the essential matrix additionally embeds information about the camera intrinsics (focal lengths, principal point, and skew). To compute an essential matrix, one needs only to have five valid feature matches (as well as the camera intrinsics) (Nistér, 2003), whereas the fundamental matrix can be computed from seven (Hartley and Zisserman, 2004) or eight (Longuet-Higgins, 1981; Hartley, 1997) valid matches.

Given the candidate feature matches and a way to compute and describe a camera transformation between the images, we still need to address the inevitable erroneous matches that will occur. While several methods exist (Hough transform (Duda and Hart, 1972), robust regression (Rousseeuw and Leroy, 2005), *etc.*), RANSAC (RANdom SAmple Consensus) (Fischler and Bolles, 1981; Raguram et al., 2008, 2013) is by far the most common strategy. RANSAC operates by iteratively selecting minimal subsets of the data, where the minimal subset size is determined by the number of samples required to estimate a transformation. Then, using this subset, a transformation is estimated, and the set of other samples that conform to this transformation (up to a predefined threshold) is computed. Based on the number of conforming samples (termed inliers), the RANSAC iterations may cease or continue running until the most likely transformation (according to some probability) is discovered (or a predefined number of iterations is exhausted). Finally, given this output set of inlier feature matches and the estimated camera transformation, the usefulness of the solution can be enforced by putting a minimum threshold on the number of discovered inliers. Additionally, degenerate configuration can be explicitly tested for and handled using methods such as QDEGSAC (Frahm and Pollefeys, 2006).

While structure-from-motion is almost always based on images related by epipolar geometries, other applications may use different definitions. For instance, in the applications of image retrieval, the retrieved results are often filtered using geometric verification. However, in this case, the geometric verification is defined by the existence of a valid affine transformation (Chum et al., 2007; Philbin et al., 2007; Chum and Matas, 2010; Chum et al., 2011), homography (Kalantidis et al., 2011; Arandjelović and Zisserman, 2012), or independent scaling and translation along the x and y axes of the image (Stewénius et al., 2012).

2.1.5 Connected Component Discovery

Given the ability to determine if a pair of images are related geometrically, we now seek to find larger sets of related images within the input dataset. Naïvely, each image in the input dataset could be matched with every other. However, this would yield $(m-1)m/2 = O(m^2)$ computation (with m being the number of images), which is prohibitive for moderate to large-scale datasets. Therefore, methods typically rely on other strategies in order to efficiently propose candidate image pairs on which to perform geometric verification.

One common strategy is image retrieval, in which an image from the dataset is issued as a query, and the most similar images within the dataset are returned for use in geometric verification. Image retrieval can rely on a global representation of the image, such as the GIST descriptor (Oliva and Torralba, 2001) or tiny-images (Torralba et al., 2008), or it can leverage a representation of the local features within the image, such as a bag-of-words representation (Nistér and Stewénius, 2006) or spatial pyramid (Lazebnik et al., 2006). By enforcing a predefined limit to the number of similar images that are returned by each query, the overall number of verification attempts can be reduced to $O(m)$ complexity.

If additional prior information is known about the images, such as their GPS location, this could be used to replace or supplement the content-based image retrieval. Furthermore, additional constraints such as vanishing points, inertial sensor data, or semantic content could be used to provide priors on the relative scene geometry or camera placement (Hays and Efros, 2008; Crandall et al., 2011, 2012).

After performing geometric verification on the set of retrieval results for all images, their connectivity can be analyzed. Here, a set of images that are transitively linked by valid transformations are termed to belong to the same connected component of images. Each of these connected components ideally represents multiple views of a unique object or scene, and can be used for structure-from-motion in the next stage.

As connected component discovery is one of the main goals of this work (discussed in Chapter 3), we will expand upon the bag-of-words representation, as it is the image similarity mechanism on which the proposed method of Chapter 3 relies. Just as a section of text could be represented by the set of words that it contains, an image can be represented by the visual elements (visual words) that it contains. Therefore, we seek a mapping between the salient elements of the image, and discrete identifiers for those elements.

To accomplish this, methods first leverage local feature extraction, in which a set of visual descriptors is extracted from the image (such as SIFT (Lowe, 2004), or the methods mentioned in Section 2.1.2). Then, these descriptors are quantized into a predefined number of quantization bins. Here, one common method is the vocabulary tree (Nistér and Stewénius, 2006).

A vocabulary tree is typically built via hierarchical k-means, in that a training set of feature descriptions is recursively partitioned using the k-means algorithm. For example, given a particular branch factor, such as $k = 10$, k initial cluster centers are found in the initial descriptor set. Then, each descriptor is assigned to its nearest cluster center, and k-means is run once again on each of these groupings. This process continues until the descriptors have been sufficiently partitioned, which is determined by a target number of clusters in the lowest levels of the tree (*e.g.* 1,000,000). The cluster centers of these lowest levels define the visual words, as they are a data-driven representation of the distribution of the descriptor space. While k-means could have initially been run using the final target number of clusters, using a hierarchical scheme greatly reduces the computational burden of this approach.

To assign a candidate descriptor to a visual word, we seek to determine the best matching descriptor (cluster center) in the lowest level of the tree. While this could be exhaustive, methods typically leverage the hierarchical nature of the tree for improved efficiency (just as the efficiency of a hierarchical approach was used to build the tree). Therefore, to determine the descriptor's assignment, it recursively traverses the vocabulary tree, determining the most similar descriptor at each level of the tree. Once the descriptor traverses to the lowest level of the tree (a leaf node), it assumes the visual word index assigned to that node.

Once all of the image's descriptors have been assigned to visual words, a histogram of their occurrences is generated. It is this histogram that defines the bag-of-words representation. Typically, this histogram will be sparse, as the number of features extracted from a single image is much smaller than the total number of visual words in the vocabulary tree. For instance, a typical image in our experiments had at most 4,000 features, whereas the vocabulary tree had 1,000,000 visual words.

To determine the similarity between two bag-of-words histograms, two popular metrics are the dot-product distance (multiplication of corresponding histogram bins), and the intersect distance (minimum value between corresponding histogram bins). With these measures, in order to avoid biasing towards histograms with relatively large numbers of visual words (or biasing against histograms with relatively few words), histograms are typically normalized to unit length, using the ℓ^2-norm for the dot-product distance measure, and the ℓ^1-norm for intersect distance.

2.1.6 Incremental Structure-from-Motion

Incremental structure-from-motion is the process by which images in a connected component are selected, their inlier feature matches analyzed, and the 3D camera poses and sparse 3D geometry are recovered. There are alternatives such as global or hierarchical methods (Shum et al., 1999; Martinec and Pajdla, 2007; Farenzena et al., 2009; Gherardi et al., 2010; Sinha et al., 2010; Crandall et al., 2011, 2012; Chatterjee and Govindu, 2013; Jiang et al., 2013; Wilson and Snavely, 2014), but the end goal of recovering the scene's geometry is the same. In this work, we leveraged the results of incremental methods such as VisualSFM (Wu, 2013) and Bundler (Snavely et al., 2006).

Incremental structure-from-motion typically begins by selecting a pair of images and triangulating an initial set of 3D points from their inlier feature matches. The pair of images is usually chosen to have a high number of inlier feature correspondences, as well as a sufficient baseline (translation between the camera centers) so that the 3D points can be estimated more reliably (Beder and Steffen, 2006).

Once the initial set of points is computed, additional images are registered to the model using 2D-3D registration techniques (Gao et al., 2003; Bujnak et al., 2008; Irschara et al., 2009; Lepetit et al., 2009). Here, images that contain inlier matches to already triangulated 3D points can have their pose estimated using a RANSAC-based alignment. Once a pose is estimated, additional 3D points can be triangulated using the two-view inlier matches that the newly added image shares with an already registered image (Hartley and Sturm, 1996; Aholt et al., 2012; Kang et al., 2014). Continuing this process, remaining images in the component can be registered, until either all images are exhausted, or no further images successfully pass the RANSAC registration criteria. In the case when not all images were registered, a new two-view reconstruction can be formed, and incremental structure-from-motion can begin again.

While the above strategy will register images, compute their poses, and triangulate points, it is likely the case that drift or other inaccuracies will occur. This could be caused, for example, by slight misalignments of the feature locations or distortion in the image caused by the optical design of the camera's lens. Therefore, to reduce the effects of these issues, bundle adjustment must be employed. Bundle adjustment is a non-linear least squares refinement (typically solved using the Levenberg-Marquardt algorithm) of the cameras' poses (center and orientation) as well as the triangulated 3D point positions. Here, the optimization seeks to minimize the reprojection errors, which is the measured displacement between a feature's detected 2D location, and the projection in the image of its corresponding triangulated 3D point. While refining the scene's geometry, bundle adjustment also typically will estimate for lens distortion, to account for and minimize its effects.

Bundle adjustment is one of the most expensive (in terms of computation) parts of a structure-from-motion system. Therefore, there are various strategies for when and how bundle adjustment should be run during the incremental reconstruction process (Agarwal et al., 2010b; Wu, 2013). Typically, however, bundle adjustment will be run at periodic stages as new images and points are reconstructed, with a final bundle adjustment at the end to compute the final refined model.

2.2 Recent Advances in Structure-from-Motion Systems

Given this overview of a structure-from-motion system, many works have been proposed to address issues relating to the overall system design, efficiency, and robustness of the system. The following sections outline several works in each of these areas.

2.2.1 System Design

When dealing with the problem of structure-from-motion (SfM) for unordered image collections, one of the first works to address this issue was that of Schaffalitzky and Zisserman (2002). To avoid the brute-force matching of all combinations of images, the authors extract affine-invariant features from each image, index them in a binary space partition tree, and then retrieve candidate matching features within a given threshold. Later, in the works by Snavely et al. (2006, 2007), the authors present an SfM system useful for organizing and browsing a photo collection. Here, the authors leverage brute-force matching between all pairs of images as input to incremental SfM. Then, a graphical user interface allows a user to visualize the reconstructed 3D sparse model, and browse from photo to photo using image-based rendering and the recovered 3D information to provide smooth transitions. Similarly, the work by Snavely et al. (2008a) attempts to automatically discover natural paths through a scene by analyzing the distribution and arrangement of the reconstructed camera poses. Then, these automatically discovered paths can be used as input for the controls to image-based rendering, allowing the user to browse photos along paths that are natural for the scene.

Also with the goal of organizing unordered photo collections is the work by Irschara et al. (2007). Here, the authors target the use-case of a community-generated photo collection that is continuously expanded. As an increasing number of photos are added, the method uses a vocabulary tree (Nistér and Stewénius, 2006) to retrieve similar existing images on which to attempt geometric verification. Depending on the results of the verification, the new image is either incrementally added to an existing SfM reconstruction, used to join two or more reconstructions, or used in a set of photos for a new reconstruction. In this way, several incremental reconstructions are maintained that capture the various scenes present in the growing photo collection. Highly similar to this is the work by Strecha et al. (2010). Here, the authors grow several reconstructions in parallel, but then leverage geotags, digital elevation models (DEM), 2D building models, and image correspondences to align and position the independent reconstructions within a global coordinate system.

While the above works targeted scenes of tens or hundreds of photos, many structure-from-motion systems have attempted to deal with the vast number of photos present in Internet photo collections. The first such works, by Agarwal et al. (2009, 2010a, 2011), propose an SfM system that demonstrates results on datasets with hundreds of thousands of images. To scale to this level, the method leverages a vocabulary tree and query expansion for similar image retrieval (Chum et al., 2007), distributed processing for feature and track generation, a subset (skeletal set (Snavely et al., 2008b)) of photos for SfM, and an image clustering strategy for dense multi-view stereo (Furukawa et al., 2010). In the end, this work demonstrates impressive results, generating reconstructions with thousands of images and spanning many different views of the imaged scenes. An alternative to this work, and one that demonstrated results on millions of images, is that of Frahm et al. (2010). Here, the authors explicitly leverage the redundancy present in a large Internet photo collection, in that a common landmark will have many photos of it that are all highly similar. Specifically, the work makes use of the whole-image GIST descriptor (Oliva and Torralba, 2001) and binary quantization (Raginsky and Lazebnik, 2009) to efficiently determine clusters of similar images using GPU computation. Then, each cluster is represented by an iconic image (Li et al., 2008), and relationships between the iconic images are discovered either through binarized-GIST, geolocation, or visual word (vocabulary tree) similarity. This final set of relationships is then fed as input for incremental SfM, the result of which is used for dense geometry estimation using multi-view stereo and fusion techniques (Kim et al., 2007; Gallup et al., 2010a,c). In comparison to Agarwal et al. (2009, 2010a, 2011), the results of Frahm et al. (2010) are more fragmented, in that the final reconstructed models do not have as much variation in their views. This is primarily a result of the image similarity metric, in that GIST clustering is not as robust as image retrieval with a vocabulary tree, and tends to favor isolated clusters of images. To help address this issue, and to increase the overall amount of registration in the dataset, the work by Raguram et al. (2011) proposes to re-cluster the initially unregistered images, in an attempt to discover new iconic images within the input dataset. Then, after this second pass of clustering, each remaining unregistered image attempts to register to a set of its nearest neighbors as determined by GIST descriptor similarity.

While the previous systems leveraged incremental SfM (in that images are added incrementally to the current reconstruction), a different class of SfM techniques exist that rely on global strategies. Here, the poses of the dataset's cameras are all estimated and refined simultaneously. One such recent work is that by Crandall et al. (2011, 2012). In this work, the system leverages various forms of prior information about the cameras' poses, including available geotags, vanishing points, and pairwise relative pose estimates. Then, this information is used as input to a discrete, belief-propagation phase, followed by continuous optimization (bundle adjustment). Another global SfM work is that by Wilson and Snavely (2014). In it, the authors leverage pairwise relative pose estimates to first estimate the global camera rotations, and then the global camera translations. While utilizing a prior work to estimate the global rotations (Chatterjee and Govindu, 2013), they propose a method to estimate the global translations through the use of inference on 1D projections.

2.2.2 Efficiency

When processing images and performing structure-from-motion, two main bottlenecks are the estimation of pairwise relationships between images, and the recovery of 3D camera and geometry information. Many works have sought to address these issues, and improve performance by either proposing faster, more efficient techniques, or through an intelligent reduction in the amount of information that must be processed.

One such work is that by Snavely et al. (2008b), where the authors propose the concept of skeletal graphs. Here, the goal is to reduce the number of images required for SfM. The authors achieve this reduction by selecting a minimal set of cameras that both span the full set of cameras (captures a large portion of its content) and retains a high level of accuracy. Similar in motivation are the works by Li et al. (2008), Raguram et al. (2011), and Frahm et al. (2010). In these works, the authors leverage the idea of an iconic image to summarize a cluster of images with similar appearance (according to the GIST descriptor (Oliva and Torralba, 2001)). These iconic views are then used for efficient structure-from-motion or recognition of a landmark scene.

Turning to the issue of pairwise relationship estimation, several works have targeted improvements to RANSAC (where RANSAC is an estimation framework that is typically necessary to find valid pairwise relationships in the presence of noisy feature correspondences between images). For instance, Raguram et al. (2008) propose a modification to RANSAC that targets real-time applications. The key insight in this work is to provide a fixed time-budget, but adapt the number candidate hypotheses based on the results of a breadth-first search through the current hypothesis set. Another work, by Sattler et al. (2009), proposes a filtering step to remove candidate correspondences before RANSAC is even run. The intuition in this work is that valid correspondences will often cluster together in the images (lying on the shared content of the images). By removing those correspondences that don't belong to such a cluster, the fraction of noisy (outlier) correspondences is reduced, which can significantly decrease the amount of time required to find a consistent solution.

In Raguram et al. (2012), the authors leverage the clustering and iconic image selection pipeline of Frahm et al. (2010), but have the goal of improving efficiency by learning from the results of past geometric verification attempts. Specifically, the authors use a classifier in order to predict the usefulness of an image for registration based on the visual words (Nistér and Stewénius, 2006) in that image. To train the classifier, the clusters' iconic images are used as positive training samples, and unregistered images are identified as negative samples. In this manner, the classifier can be trained on-the-fly as images are processed, and then employed to avoid even attempting to register to images that are rejected by the classifer. In addition to this, the authors also learn which visual words are more likely to be useful for registration. By keeping track of which image features are inliers to pairwise relationships, and the features' corresponding visual words, a ranking is constructed of the visual words to most frequently be an inlier. Then, this ranking is used to prioritize sampling within a RANSAC framework.

Also leveraging visual words is the work by Havlena and Schindler (2014). In it, the authors use a fine-grained visual vocabulary (16 million visual words), and make the key assumption that at such a fine level of quantization, image features sharing the same visual word belong to the same feature track (set of observations of a 3D point). Then, by enforcing a minimum number of common visual words between each pair of images, and by clustering those images together that contain the same tracks, they obtain a set of connected components without ever explicitly performing geometric verification.

Looking to put an overall bound on the runtime of an incremental SfM pipeline is the work by Wu (2013). In it, the author seeks to improve various parts of the pipeline so that an overall $O(m)$ complexity is obtained with respect to the number of cameras (as opposed to the $O(m^3)$ or $O(m^4)$ complexities of other prior works). In order to achieve this, Wu introduces a new method to avoid estimating pairwise relationships between images with a low chance of success. Here, SIFT features (Lowe, 2004) are extracted, and those features with the largest scales are analyzed for their similarity. If two images have insufficient similarity in their upper-level features, then that pair of images is skipped during pairwise relationship discovery. In addition, Wu also proposes a new linear-time strategy for bundle adjustment, by limiting how frequently a full bundle adjustment operation can occur. In order to minimize a loss of accuracy, a re-triangulation step is also proposed, which seeks to attempt to triangulate 3D points for feature matches which had previously failed that operation.

2.2.3 Robustness

In the process of reconstructing a scene via structure-from-motion, it may occur that the scene contains duplicate or repetitive content. This can cause errors in the final SfM result. The reason for this is that the placement of cameras may be ambiguous, or more commonly, the greedy-based nature of typical incremental SfM systems incorrectly registered separate instances of the duplicate or repetitive structure. Handling these types of scenarios is one way in which an SfM system can be robust to its input imagery, and is the type of robustness that is primarily addressed in this thesis.

The first such approach to handle these issues is that of Zach et al. (2008). Here, the work utilizes a concept of *missing correspondences* to correct for the influence of indistinguishable scene elements, where the main principle is to identify consistent camera triplets that contain a similar set of feature observations. If an image in a triplet is missing a substantial number of feature correspondences compared to the other two images, then that image is suspect of being a false match. By identifying the correct set of triplets, and combining them together, a correct reconstruction is obtained. Expanding on this work, Roberts et al. (2011) also utilize missing correspondences, but focus on scenes with large duplicate structures. The authors use an expectation maximization (EM) algorithm to combine verified camera triplets and form a correct reconstruction that minimizes the number of missing correspondences. To cope with particularly difficult scenes, a portion of their results rely on image timestamps to resolve ambiguities. Using motivation from these works, Jiang et al. (2012) use missing correspondences to correctly reconstruct a scene. Here, the underlying assumption of the approach is that the images depict a single complete model, and by optimizing over various possible reconstructions, one can minimize a cost function related to the total number of missing correspondences.

An alternative to these approaches is that by Zach et al. (2010), which analyzes geometric loop constraints. This work makes the observation that, given a cycle of connected cameras and the relative transformations between them, traversing and accumulating the loop's transforms should result in the identity transform. Any loop that deviates too far from this is identified as containing at least one inconsistent image match. By analyzing a large number of loops, they discover the inconsistent camera matches.

A more recent work is that by Wilson and Snavely (2013). Here, the authors leverage the bipartite local clustering coefficient ($blcc$) to determine those 3D points that lead to an erroneous reconstruction. The $blcc$ metric achieves this by analyzing a bipartite graph encoding the visibility of 3D points in each image. Then, points whose neighbors are themselves not strongly connected to each other are identified as having a low $blcc$ value and are pruned from the reconstruction. The intuition is that 3D points within a similar part of the scene should have similar visibility throughout the images in the reconstruction. In order to identify the final set of indistinguishable points, the authors assume that the final number of split components (sub-models in the reconstruction) is known beforehand.

All of the aforementioned works attempt to explicitly correct for potential errors caused by duplicate or repetitive structure in a scene. Another class of works, though slightly less related, attempts to use these structures to improve the final reconstruction assuming that no prior errors exist. For example, Wu et al. (2011) and Köser et al. (2011) respectively identify repetition and symmetry within an image, and then use that regularity as multiple observations of a structure to generate a reconstruction from only one image. Jiang et al. (2011) detect both repetitive and symmetric structure in an SfM reconstruction from small-scale datasets (tens of images), and use these to generate a more complete, accurate, and dense final model. Finally, Cohen et al. (2012) locate planes of symmetry within a scene, and then use these as constraints in bundle adjustment.

2.3 Remaining Challenges

Given this body of prior work, there are several remaining challenges to be addressed in order to construct an efficient and robust large-scale structure-from-motion system.

One such remaining challenge is the data association problem, in which the relationships between a large set of images must be discovered. Current approaches either require significant computation time (in that all possible pairings of images must be evaluated) (Snavely et al., 2006; Wu, 2013), or high memory requirements (in that a large database which indexes all of the images in the dataset must be stored in memory) (Agarwal et al., 2009, 2010a, 2011; Frahm et al., 2010; Havlena and Schindler, 2014). This work addresses both the computation time and memory requirements of the data association problem, and describes such a method in Chapter 3.

Another remaining challenge is the robust disambiguation of duplicate structures within a reconstructed scene. Here, previous approaches have limiting assumptions such as priors on the final model that is output (Zach et al., 2008; Jiang et al., 2012), reliance on repeatable and consistent feature observations (Wilson and Snavely, 2013), image sequence information (Roberts et al., 2011), or significant inconsistency of the computed camera poses (Zach et al., 2010). These shortcomings make these methods unfavorable choices when handling the diverse set of images and scenes encountered in a crowdsourced image collection. Therefore, we motivate and propose two methods in Chapters 4 and 5 which address this challenge, but lack the limitations of prior works.

CHAPTER 3: STREAMING PARADIGM FOR CONNECTED COMPONENT DISCOVERY

3.1 Introduction

For decades, modeling the world from images has been a major goal of computer vision, enabling a wide range of applications including virtual reality, image-based localization, and autonomous navigation. One of the most diverse data sources for modeling is Internet photo collections, and the computer vision community has made tremendous progress in large-scale structure-from-motion (LS-SfM) from Internet datasets over the last decade. However, utilizing this wealth of information for LS-SfM remains a challenging problem due to the ever-increasing amount of image data. For example, it is estimated that 10% of all photos have been taken in the last year alone (Memories, 2014). In a short period of time, research in large-scale modeling has progressed from modeling using several thousand images (Snavely et al., 2006, 2007) to modeling from city-scale datasets of several million (Frahm et al., 2010). Major research challenges that these approaches have focused on are:

- **Data Robustness**: Enable the modeling from unorganized and heterogeneous crowdsourced Internet photo collections.
- **Compute & Storage Scalability**: Achieve efficiency to meet the true scale of the ever-increasing size of Internet photo collections.
- **Registration Comprehensiveness**: Discover and verify as many camera-to-camera associations as possible.
- **Model Completeness**: Reconstruct 3D scene models that are as extensive and panoramic as possible.

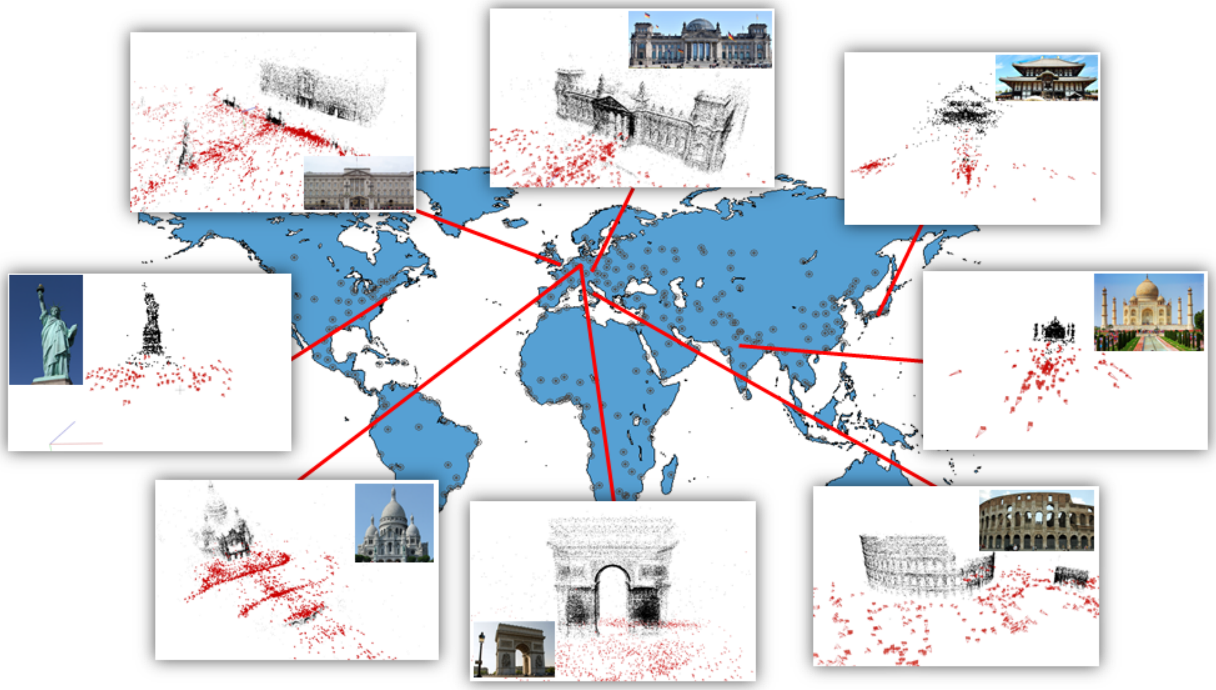

Figure 3.1: Examples of our world-scale reconstructed models.

In practice, these goals have been prioritized differently by existing LS-SfM frameworks (Snavely et al., 2006, 2007; Agarwal et al., 2009, 2011; Frahm et al., 2010; Schönberger et al., 2015b). The approach of Frahm et al. (2010) emphasizes scalability to enable modeling from millions of images. While it achieves impressive city-scale models, this emphasis leads to limitations in the model completeness. In contrast, the approach of Agarwal et al. (2009, 2011) prioritizes model completeness, but can only model from hundreds of thousands of images, instead of millions. We propose a novel structure-from-motion framework that advances the state of the art in scalability from city-scale modeling to world-scale modeling (several tens of millions of images) using just a single computer. Moreover, our approach does not compromise model completeness, but achieves results that are on par or beyond the state of the art in efficiency and scalability of LS-SfM systems. We demonstrate this scalability by performing 3D reconstructions from the 100 million image world-scale Yahoo Flickr Creative Commons dataset (Shamma, 2014; Thomee et al., 2015). Our method reconstructs models from a world-scale dataset on a single computer in six days leveraging approximately 96 million images (see examples in Figure 3.1).

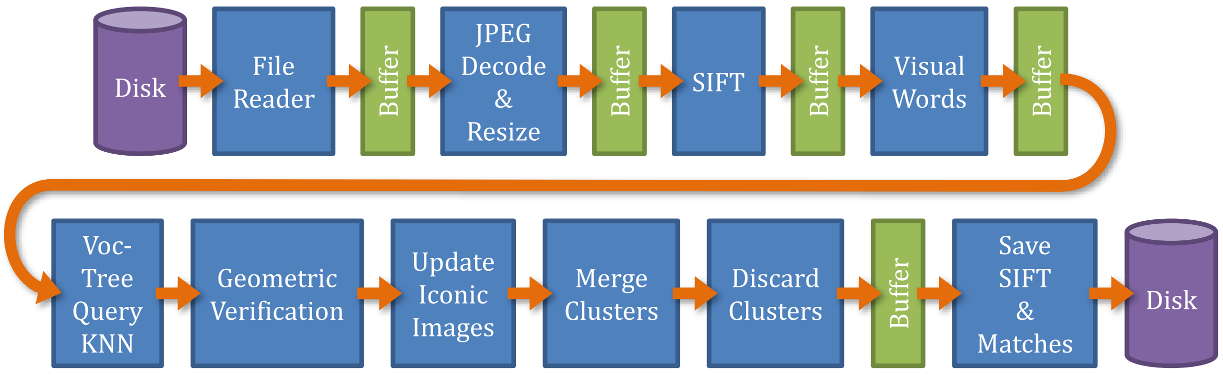

Figure 3.2: Overview of the pipeline of our method.

Our framework achieves this high scalability by adopting a streaming-based paradigm for connected component discovery. In order to balance between registration comprehensiveness and data compactness, we employ an adaptive, online, iconic image clustering approach based on an augmented bag-of-words representation. The new image cluster representation overcomes several limitations of previous representations, which tended to partition images of the same scene into multiple independent models. In achieving more large-scale scene integrity, our novel cluster representation also avoids needlessly increasing the size of the indexing structure, which previously prohibited the use of datasets of tens of millions of images. Given the constantly increasing size of available photo collections, we posit streaming-based processing as a natural compute paradigm for world-scale structure-from-motion (WS-SfM).

3.2 World-Scale Structure-from-Motion

When considering the goal of processing world-scale data, we must define what it means for a dataset to be world-scale. One primary characteristic of such a dataset is size. For instance, we seek to design a system that can handle a magnitude of images that was not previously feasible. Practically, the largest datasets which had been used for structure-from-motion were approximately three million images in size (Frahm et al., 2010). Here, images were downloaded from a single photo-hosting service (Flickr), and corresponded to a particular city (Rome, Italy or Berlin, Germany). When downloading images from Flickr for other cities around the world, such as Paris, France or London, England, we discovered around 10 million images for each of these cities (Table 3.1). Therefore, a world-scale dataset in our definition should consist of at least this number of images, for instance, containing several tens of millions of images.

Another aspect of a world-scale dataset is diversity. A world-scale dataset should consist of images from a multitude of landmarks from around the globe. In this manner, the images would not all be taken from the same geographic region, and would reconstruct into many independent connected components (as in the best case, there could be connected components for each of the connected landmasses on earth). This is in contrast to a city-scale dataset, where images all depict locations within a common city, and could potentially be registered into a single connected component.

Combining these two aspects is the Yahoo 100 million image dataset (Shamma, 2014; Thomee et al., 2015). It contains approximately 100 million images (a size that was previously infeasible), of locations geographically distributed throughout the entire world (and thus represent many independent reconstructable locations). Therefore, we will use this dataset to demonstrate the effectiveness of our method on such large-scale and diverse image collections.

The major challenge for WS-SfM is the massive amount of imagery, where our method needs to efficiently identify overlapping images of each captured scene (Section 3.2.1). City-scale data processing algorithms (Agarwal et al., 2009, 2011; Frahm et al., 2010) already spend significant effort on efficient representations to maintain performance. To scale to world-scale datasets, we propose an efficient method for processing images in a sequential fashion (streaming). Our proposed streaming imposes the constraint on the processing that, in one pass through the data, an image is only loaded once from disk (or other input source) and the image is discarded after a limited period of time (much smaller than the overall computation time). The efficiency of streaming methods for big data has long been known, for example, in mesh processing (Isenburg and Lindstrom, 2005), identifying the most frequent elements in a stream (Manku and Motwani, 2002; Karp et al., 2003; Cormode and Muthukrishnan, 2005b), or determining various statistics for a constant stream of changing data (Gaber et al., 2005). The major challenge posed by stream processing

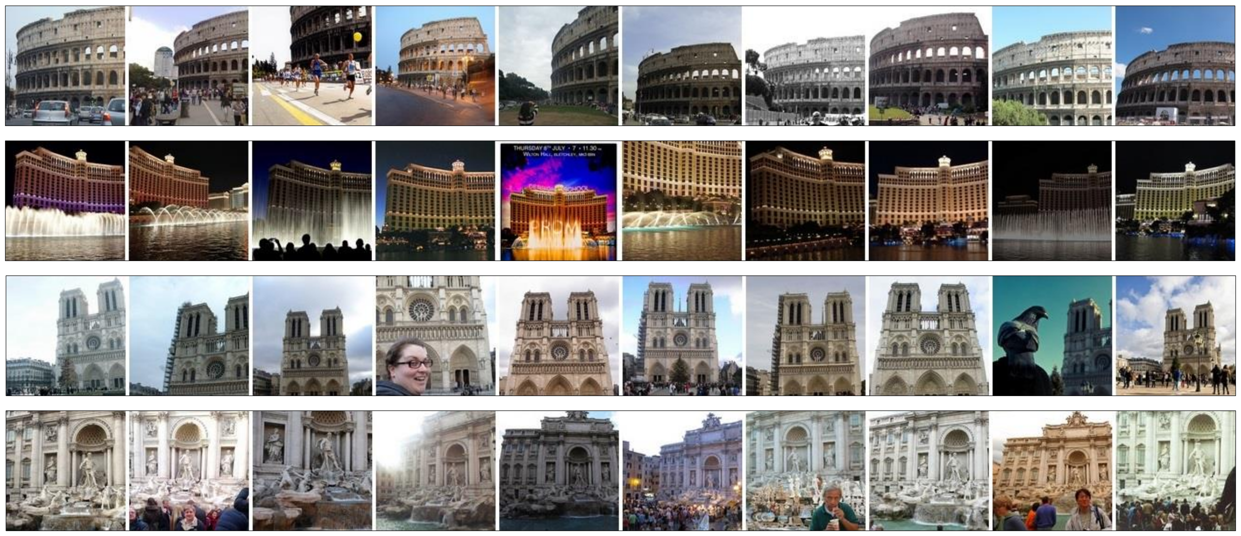

Figure 3.3: Sample image clusters from our pipeline. The leftmost image in each cluster is the iconic image.

for image overlap detection is to ensure that overlap is detected even when the images are not concurrently loaded. To meet these constraints, we propose to maintain and update in realtime a concise representation of our current knowledge of the images' connectivity. Upon discovering the sets of connected images (referred to as connected components), we then perform incremental SfM to recover the 3D geometry of the dataset's scenes (see Section 3.2.2). A high-level flow chart of our method is shown in Figure 3.2.

3.2.1 Clustering & Connected Component Discovery

In our streaming algorithm, we aim to identify images that view the same scene. Similar to Frahm et al. (2010), we seek to discover clusters of similar images, where each cluster is represented by an iconic image (see Figure 3.3 for examples). In this context, in order for an image to belong to a cluster, it must successfully register to the iconic image of that cluster – *i.e.* , there has to be a valid epipolar geometry between the iconic and the cluster image (geometric verification). We add to this the notion of connected components, where each is a set of connected clusters (clusters of the same scene for which images exist that establish an epipolar geometry between the iconic images).

Algorithm 1 Streaming Connected Component Discovery

```
1: Create empty set of clusters $\mathbb{C}$
2: for each batch of images $\mathbb{B}$ do
3:     if first batch of images then
4:         for each image $I_{\mathbb{B}}$ in batch $\mathbb{B}$ do
5:             Create new single-image cluster in $\mathbb{C}$ using $I_{\mathbb{B}}$
6:     else
7:         for each image $I_{\mathbb{B}}$ in batch $\mathbb{B}$ do
8:             $\mathbb{K} \leftarrow$ IconicImageRetrieval($I_{\mathbb{B}}$, $\mathbb{C}$)
9:             $\mathbb{R} \leftarrow$ GeometricVerification($I_{\mathbb{B}}$, $\mathbb{K}$)
10:            $\mathbb{M}, \mathbb{S} \leftarrow$ ProcessGeometricVerificationResults($I_{\mathbb{B}}$, $\mathbb{K}$, $\mathbb{R}$, $\mathbb{C}$)
11:            $\mathbb{S}$, $\mathbb{C} \leftarrow$ ClusterMerging($\mathbb{M}$, $\mathbb{S}$, $\mathbb{C}$)
12:            $\mathbb{C} \leftarrow$ IconicSelection($\mathbb{S}$, $\mathbb{C}$)
13:            $\mathbb{C} \leftarrow$ ClusterDiscarding($\mathbb{C}$)
```

To perform the cluster and connected component analysis in our streaming approach, we process the images in batches (as outlined in Algorithm 1). The images of the first batch are used as our initial iconic images (Step 3 in Algorithm 1); in other words, the first batch represents our scene viewpoints. Note that these initial clusters will be merged or discarded, as appropriate, in the later processing. Hence, even if they are not suitable iconic images, they do not impair our results. For every following batch we perform the main components of our system (Step 7 in Algorithm 1), which is detailed in the next sections.

3.2.1.1 Image Overlap Detection

The objectives of our method during streaming are the detection of pairwise image overlap and the discovery of connected components. We propose to combine these two objectives into a unified computation, which allows us to achieve significantly higher data throughput and reduced computational complexity. We use the iconic images (more specifically, their augmented features, see Section 3.2.1.2), to represent the currently known state of the scene within our system. Loosely speaking, we represent the visual information of a particular viewpoint by an iconic image's augmented features indexed in a vocabulary tree.

Algorithm 2 Geometric Verification

```
1:  function GeometricVerification($I_{\mathbb{B}}$, $\mathbb{K}$)
2:      Create empty set of successful registrations $\mathbb{R}$
3:      for each iconic image $I_{\mathbb{K}}$ in $\mathbb{K}$ do
4:          if $I_{\mathbb{B}}$ and $I_{\mathbb{K}}$ belong to the same connected component then
5:              continue
6:          $R \leftarrow$ AttemptRegistration($I_{\mathbb{B}}$, $I_{\mathbb{K}}$)
7:          if registration $R$ is successful then
8:              UpdateIconicImageRepresentation($I_{\mathbb{B}}$, $I_{\mathbb{K}}$, $R$)
9:              Add registration $R$ to set of successful registrations $\mathbb{R}$
10:         if number of registration attempts = $k_v$ then
11:             break
12:     return $\mathbb{R}$
```

During the streaming of the dataset, every loaded image uses the vocabulary tree to query for its k-nearest neighbors (IconicImageRetrieval of Algorithm 1, where we chose $k = 25$). In order to verify if these nearest neighbors overlap with the new image, we perform efficient geometric verification (AttemptRegistration of Algorithm 2) using ARRSAC (Raguram et al., 2008), which is a version of RANSAC designed for real-time applications. Coupled with this, we use a 5-point essential matrix estimator (Nistér, 2003), with estimates for the intrinsic camera parameters initialized using JPEG EXIF data whenever possible (assuming a $40°$ horizontal field-of-view otherwise). Additionally, we limit the number of ARRSAC iterations to 400, for the same reasons as (Frahm et al., 2010).

While geometric verification can be performed extremely efficiently (Frahm et al., 2010; Raguram et al., 2012), it is still a major contributor to the computational expense of an SfM system. We empirically observed that not all retrieved nearest neighbors are equally valuable for image overlap detection (a similar observation was made by Lou et al. (2012)). Leveraging this observation, we set a budget $k_v < k$ for geometric verification (Step 10 of Algorithm 2) and only evaluate the k_v most relevant nearest neighbors (we set $k_v = 2$ when generating our results). Our strategy is to first spend the k_v match budget per image on the highest-ranked nearest neighbors in the k retrieval results. However, once a successful match is achieved and there is a remaining budget, further matches are only performed on nearest neighbors that do not belong to the same connected

Algorithm 3 Process Geometric Verification Results

```
1: function ProcessGeometricVerificationResults($I_{\mathbb{B}}$, $\mathbb{K}$, $\mathbb{R}$, $\mathbb{C}$)
2:     Create empty list of candidate clusters to merge $\mathbb{M}$
3:     Create empty list of clusters for iconic image selection $\mathbb{S}$
4:     if number of successful registration results in $\mathbb{R}$ = 1 then
5:         Add $I_{\mathbb{B}}$ to registered cluster
6:         if new size of registered cluster = 3 then
7:             Add cluster to list of clusters for iconic image selection $\mathbb{S}$
8:     if number of successful registration results in $\mathbb{R} \geq 2$ then
9:         Add $I_{\mathbb{B}}$ to best registering cluster
10:        Link clusters from $\mathbb{R}$ into connected component
11:        Add clusters from $\mathbb{R}$ to candidate merge list $\mathbb{M}$
12:    if number of successful registration results in $\mathbb{R}$ = 0 then
13:        Create new single-image cluster in $\mathbb{C}$ using $I_{\mathbb{B}}$
14:    if at least 2 iconic images from first 3 entries of $\mathbb{K}$ belong to same connected component
       and $I_{\mathbb{B}}$ registered to that connected component then
15:        Add the clusters from the same connected component to the candidate merge list $\mathbb{M}$ in
       pairs
16:    return $\mathbb{M}$, $\mathbb{S}$
```

component (Step 4 of Algorithm 2, similar to (Lou et al., 2012) and (Heath et al., 2010)). Intuitively, this fosters registration to new iconic images not already associated with the currently-matched component.

During the above processing, we seek to discover any connections between the current image and the set of iconic images. Once an image registers to an iconic image, we associate it with that iconic image and add it to its cluster (Step 4 of Algorithm 2). However, in the case where an image registers to two or more iconic images, we associate it with the iconic image with which it had the highest number of inliers (Step 8 of Algorithm 2). Next, we detail our iconic image representation before discussing the selection strategy for choosing iconic images.

3.2.1.2 Iconic Image Representation and Selection

While we leverage the idea of iconic images representing clusters of images from Li et al. (2008) and Frahm et al. (2010), their use of the GIST descriptor results in the clusters covering a small distribution of images around a particular viewpoint and at similar lighting condition. Moreover,

Algorithm 4 Update Iconic Image Representation

```
1: function UPDATEICONICIMAGEREPRESENTATION($I_{\mathbb{B}}$, $I_{\mathbb{K}}$, $R$)
2:     for each inlier correspondence $i$ in $R$ do
3:         $v \leftarrow$ visual word of $i$ from $I_{\mathbb{B}}$
4:         $\mathbb{V} \leftarrow$ set of visual words of $i$ from $I_{\mathbb{K}}$
5:         if $v$ not in $\mathbb{V}$ then
6:             Add $v$ to $\mathbb{V}$ and update $I_{\mathbb{K}}$
```

Algorithm 5 Iconic Selection

```
1: function ICONICSELECTION($\mathbb{S}$, $\mathbb{C}$)
2:     for each cluster $c$ in list of clusters for iconic image selection $\mathbb{S}$ do
3:         $R \leftarrow$ Attempt registration between 2nd and 3rd images within cluster $c$
4:         if registration $R$ successful then
5:             Select the image from the first 3 images of cluster $c$ that has the greatest number
           of inlier feature correspondences to the other 2 images and use it as the iconic image for the
           cluster in $\mathbb{C}$
6:     return $\mathbb{C}$
```

GIST-based clustering has very limited ability to cope with occlusions, which are frequent in Internet photo collections. To control the complexity of the representation, we propose a new cluster representation that covers a broader set of views by taking inspiration from image retrieval techniques. For instance, there have been a number of approaches that leverage the idea of query expansion or relevance feedback to improve the quality and breadth of the retrieved results (Lou et al., 2012; Chum et al., 2011; Agarwal et al., 2009, 2011; Frahm et al., 2010). Generally speaking, these methods retrieve a subset of results, and then based on what was returned, a new query is issued to find an enhanced set. An alternative strategy is database-side feature augmentation (Turcot and Lowe, 2009; Arandjelović and Zisserman, 2012), which leverages a static dataset to extend an image's bag-of-words representation with the representations of its geometrically verified neighbors. We opt for database-side augmentation to achieve high efficiency by not incurring the expense of reissuing queries.

In our approach, the database-side feature augmentation (Turcot and Lowe, 2009; Arandjelović and Zisserman, 2012) is applied to our current set of iconic images. Each iconic image is represented by a set of visual words (used for image retrieval in Section 3.2.1.1), which is then augmented based on the images that register to it. Specifically, every time a new image is linked to an iconic image, we add the visual words of the new image's inlier features to the set of visual words belonging to the iconic image (Algorithm 4). Each feature in an iconic image then tracks the visual words with which it has been associated (either by original assignment or via an inlier to a newly-match image, refer to $\mathbb{V}$ from Step 6 of Algorithm 4).

For efficiency and sparseness of representation, we limit the augmentation to only include those visual words not already associated with the iconic image's feature to which they were an inlier. This prevents an unnecessary bias toward the current set of inlier features, allowing the other features in the image to more readily be used for retrieval. In addition to improving the quality of retrieval results, the augmentation can also be viewed as overcoming quantization artifacts of the vocabulary tree. For instance, if a feature is nearly equidistant to two or more visual words, that feature can be associated with those visual words once it becomes an inlier match to an image that had a different visual word assignment for a similar feature.

Having discussed our iconic image representation, we now detail the process of iconic image selection. Conceptually, our iconic images represent the images assigned to their clusters. Hence, if we encounter a new image that does not register to any current iconic image, we consider it to be representing an as-yet unknown scene or scene part. This new image temporarily represents a new cluster until further images are added to the cluster. Taking inspiration from Frahm et al. (2010), we select the permanent iconic image after the cluster has grown to contain q images ($q = 3$ for all our experiments, and Step 6 from Algorithm 3 and Step 7 from Algorithm 6). The permanent iconic image is selected as the cluster image with the highest number of inliers to the other images in the cluster (Step 5 from Algorithm 5).

3.2.1.3 Cluster Merging

During the above process of creating new iconic images, it is possible that two iconic images are created for essentially the same scene content. For instance, this can most easily be seen for the first batch of images whose images automatically become iconic images without being evaluated for mutual overlap. Other cases of similar iconic images could result from retrieval failures or due to the

Algorithm 6 Cluster Merging

```
1: function ClusterMerging($\mathbb{M}$, $\mathbb{S}$, $\mathbb{C}$)
2:     Sort $\mathbb{M}$ so that the candidates with the smallest-sized clusters appear first
3:     for each candidate pair of clusters to merge $P$ in sorted list $\mathbb{M}$ do
4:         $R \leftarrow$ Attempt registration between iconic images from both clusters of $P$
5:         if registration $R$ successful then
6:             Merge the two clusters in $\mathbb{C}$ and represent them with the iconic image from the larger of the two clusters
7:             if size of merged cluster is now $\geq 3$ then
8:                 Add merged cluster to list of clusters for iconic image selection $\mathbb{S}$
9:     return $\mathbb{S}$, $\mathbb{C}$
```

limited compute budget k_v in the verification of the retrieved candidates. Retrieval failures result in the ideal iconic image not being retrieved due to quantization artifacts, a high amount of occlusion, or other confusing visual words being present in the image. The limited compute budget can lead to non-evaluated relevant iconic images. To overcome these limitations, we propose a cluster merging step in which geometric verification is attempted on similar iconic image pairs. The first indication that a pair of iconic images may be similar is when a new image successfully registers to two iconic images (Step 8 from Algorithm 3). To handle the case where the iconic images reside in the same connected component (as we prevent duplicate matches to the same connected component), we also look at the order of retrieval results. If a new image matches to one of the first r iconic image retrieval results, and there are retrieval results that belong to the same connected component, we flag these iconic images as candidate clusters for merging (in our experiments we set $r = 3$, see Step 14 from Algorithm 3).

Once we have found the candidate clusters to merge, we sort them by size so that we merge the smallest cluster first (Step 2 from Algorithm 6). The reasoning for this is that we want to maintain a compact and concise iconic image set, and merging two smaller clusters increases the average iconic-to-cluster image ratio more than merging a small cluster with a large one. If the iconic images for a pair of candidate clusters register, the cluster images and iconic image from the smaller of the two clusters are appended to the larger cluster and the larger cluster's iconic image's representation is augmented. This merging ensures that, over time, our scene representation

Algorithm 7 Cluster Discarding

```
1: function ClusterDiscarding($\mathbb{C}$)
2:     for each cluster $c$ in list of all clusters $\mathbb{C}$ do
3:         $d \leftarrow$ (size of cluster $c$) / (number of images processed since cluster $c$'s formation)
4:         if $d < 1/w$ then
5:             Remove cluster $c$ from $\mathbb{C}$
6:     return $\mathbb{C}$
```

stays as compact as possible. Now that we have introduced our new stream processing algorithm for obtaining overlapping images and connected components, we will discuss the challenges in world-scale data management that remain even with our compact and efficient representation.

3.2.1.4 World-Scale Data Management

Unordered world-scale photo collections pose significant challenges for data storage and, in general, cannot be maintained in memory. It is critical to develop an efficient strategy for data association and for the pruning of unrelated images. We propose a strategy that measures the increase of information of a cluster in order to decide on its importance for the world-scale scene representation. This strategy enables our streaming approach and improves the efficiency for handling world-scale data of arbitrary size.

To ensure memory efficiency, an image's data (SIFT features, visual words, camera intrinsics) are stored in memory only as long as it is needed. For instance, an iconic image could be matched to at any point, so its SIFT features should be readily available. Furthermore, a cluster of size less than q will need to retain its images' data until it undergoes its iconic image selection phase. All other images can have their data immediately discarded, as the images will not be used for any further match attempts.

For large or diverse datasets, this may still overreach the memory resources, as the number of iconic images could continually increase. To circumvent this problem, we limit the number of images we store in memory by enforcing a minimum information growth rate for each cluster. The motivation for this measure comes from the observation that as the number of clusters grows, the scene coverage saturates. Therefore, we desire to prune those clusters that no longer add value to the scene's representation in memory. We enforce a minimum growth rate (which we call the discard rate) by computing the ratio between a cluster's current size and the total number of images that have been processed since the cluster's creation. If this ratio falls below a threshold $1/w$, we discard the cluster's image information from memory (Step 4 from Algorithm 7). Another way to think about this parameter is that it defines the width of a sliding window in which a cluster's size is required (on average) to grow by at least one. Note that during this process of cluster discarded we still track that a discarded cluster belongs to its connected component, we just do not allow it to grow any further. A side benefit of this strategy is that it naturally limits the lifetime of unrelated/single images, as a single image cluster will persist only until w additional images have been processed.

Additionally, our strategy for discarding clusters helps to eliminate bad iconic images. For instance, the case may exist where two iconic images show similar content, but fail to register to each other (and thus do not merge). If one of the iconic images has a low-quality set of features or visual words, and if no better option was available during the iconic image selection phase, then its cluster size will be significantly smaller than the iconic image with a high-quality, repeatable representation. Therefore, as processing continues, the smaller cluster, and lower-quality iconic image, will be discarded as the higher-quality iconic image registers to an increasing number of images.

Choosing the growth parameter w immediately influences our probability to find overlapping images in the dataset. Let us assume that every image within a connected component can successfully register to every other image of the component. While this assumption does not fully hold in practice, especially for components that have a wide spatial coverage, this assumption is much more reasonable for a single cluster of images. Additionally, let us assume that the images for the connected component are randomly dispersed throughout the entire dataset of size M. If the c_m images are dispersed in the worst case, the average number of images between them in the input ordering is the greatest (*i.e.* the c_m images occur at intervals of M/c_m). Then, finding matches between the images is only possible if w is large enough to preserve images in memory for that duration. Specifically, w would have to be set such that $w > M/c_m$. Therefore, for a dataset that contains 10 million images, and with $w = 100{,}000$, conceptually we could hope to reliably recover only those connected components (or clusters) of size > 100 images. However, there are additional issues such as the reliability of image retrieval and geometric verification which further impact these estimates. Therefore, a deeper look at this issue is explored in Section 3.3.6. In our experiments, we set $w = 100{,}000$ for the city-scale datasets, and $w = 200{,}000$ for the world-scale dataset (Thomee et al., 2015).

3.2.2 Structure-from-Motion

To generate structure-from-motion (SfM) models, we leverage the connected components already discovered during the streaming phase, but densify the connections in order to allow for more accurate and complete reconstruction. This provides us a significant advantage over previous methods such as Frahm et al. (2010) as we do not need to burden our structure-from-motion processing with cluster or iconic image matching, which can be a significant effort for tens or hundreds of thousands of iconic images as encountered in our processing. Note the amount of iconic images that we obtain is at the scale of the number of images processed in previous methods such as Agarwal et al. (2009, 2011). For increased performance and stability, we perform a separate hierarchical structure-from-motion process for each connected component by first building a skeletal reconstruction based on the iconic images of clusters with more than three images and a few linking images for those iconic images. Then, we register the remaining images with respect to the skeletal reconstruction.

For the reconstruction from the iconic images it is important to note that for the sparseness of the representation in the streaming, we enforced the fact that iconic images should be a sparse representation of the scene and hence they do not match densely with each other. Therefore, to foster a successful reconstruction, we need to first add additional images and densify their set of image connections.

We chose the additional images to be those images with connections to multiple iconic images, so that each iconic image is connected to as many other iconic images as possible. To quantify the connectivity during the addition of the linking images, we track this connectivity by creating a sparse adjacency matrix A. Each entry $a_{i,j}$ of matrix A will store the number of connections between iconic image i and j. At first we test for connection with the 50 nearest neighbors of each iconic image within the current iconic image set using vocabulary tree image retrieval and geometric verification. Based on the results of these match attempts, we update A to have an entry of 1 wherever two iconic images successfully matched. As our iconic images are very disjoint, A is by design still very sparse after this step. To increase the density of A, we now turn to the linking images within our dataset. Here, a potentially beneficial connecting image is any image that registered to two or more other images during the streaming process (*i.e.* an image that matched to and connected two iconic images). Our goal is to add a subset of these connecting images to our current iconic image set, such that we are left with a set of images that is well-connected and ready for reconstruction.

In order to discover the subset of connecting images to use for reconstruction we employ a greedy strategy which adds connecting images based on the number of iconic images to which they register. We compute this number by first matching each connecting image to its 10 nearest neighbors in the iconic image set (once again using the vocabulary tree). Then, we rank the connecting images by the number of new connections that they add, and greedily select the ones with the most new connections. We continue until there are either no connecting images left, or the connecting images no longer provide new links between iconic images.

After preparing all connected components for reconstruction, we then process them in parallel using our structure-from-motion software. Once structure-from-motion provides a sparse 3D model for each connected component, we register the remaining connected component images to this model using an efficient P3P (perspective-3-point) algorithm. Specifically, each connected component image should be connected to one or more iconic images, and we further increase this connectivity by matching each unused connected component image to its 2 nearest neighbors within the current reconstruction. Then, we perform registration between these connected component images and the model, enforcing that no new points are triangulated, and no global bundle adjustment is run.

Therefore, while we do not increase the completeness or accuracy of the final model's geometry, we do have pose estimates for as many of the connected component images as possible.

3.2.3 Sparse Reconstruction Correction

When performing structure-from-motion on urban scenes, one issue that arises is that of symmetry and misregistered cameras. Specifically, the presence of identical-looking structure in different parts of the scene can confuse SfM algorithms causing cameras and geometry to be misplaced (and our results do demonstrate this behavior). Fortunately, there are several recent works that addressed this issue (Zach et al., 2010; Roberts et al., 2011; Jiang et al., 2012; Wilson and Snavely, 2013; Heinly et al., 2014a,b). We leverage the work by Heinly et al. (2014a) (Chapter 4) for its robustness, but optimized the implementation to achieve higher processing times for our models.

To increase the speed of the method, we made the observation that if a symmetry error is present, it is usually isolated to one or more distinct elements within a scene (a building facade, dome, *etc.*). Therefore, as opposed to processing and analyzing the entire scene at once, we propose to divide the scene into meaningful separate sub-models. We then process each of these sub-models separately, and combine the results back together.

To discover the separate sub-models of the reconstruction, we first project all of the 3D points to the ground plane (where the ground plane is discovered by fitting a plane to the camera positions). Then, we run mean-shift over the projected point positions, and the detected point clusters are determined to be separate geometric structures. We then isolate those cameras that see a particular point cluster, and run the method of (Heinly et al., 2014a) on this point and camera subset.

3.3 Experimental Evaluation

To test our approach, we ran our method on datasets of widely varying sizes (see Table 3.1), the smallest being around 74,000 images and the largest being about 96 million. Two of the datasets

	Number of Images							Time (hours)			
Dataset	Input	Valid	Registered	CC_1	CC_2	Iconics	SfM	Stream	Densify	SfM	Register
Roman Forum	74,388	73,566	45,341	17,804	2,971	3,408	20,176	0.35	0.27	0.42	0.10
Berlin	2,704,486	2,661,327	702,845	259,705	6,869	42,612	194,870	7.89	1.14	2.92	2.66
Paris	10,390,391	10,177,094	2,492,310	1,228,062	7,016	131,627	858,134	29.16	4.04	57.85	6.96
London	12,327,690	12,036,991	3,078,303	779,036	17,382	228,792	566,778	38.29	5.57	22.72	6.57
Yahoo	96,054,288	92,282,657	1,499,110	75,308	64,995	74,660	168,099	105.4	3.5	17.4	-

Table 3.1: Statistics for our tested datasets. The Roman Forum dataset was obtained from Lou et al. (2012), the Berlin dataset from Frahm et al. (2010), and the Yahoo dataset from Thomee et al. (2015). CC_1 and CC_2 refer to the size of the first and second largest connected component. Iconic images are for clusters of size ≥ 3, and the SfM results report on the 32 largest connected components (or components with ≥ 50 images for the Yahoo dataset).

were obtained from the authors of previous publications (Lou et al., 2012; Frahm et al., 2010), which provide a basis of comparison between the methods.

3.3.1 Implementation and Parameter Configuration

In our evaluation, we leverage a mixed MATLAB, C++, and CUDA implementation of our proposed streaming method. For the streaming and reconstruction of the city-scale datasets, we used the same PC as in Frahm et al. (2010) to allow direct comparison of results. For processing the world-scale dataset (Thomee et al., 2015) we used a dual processor computer with 256 GB of RAM and five Nvidia graphics cards which are leveraged in the CUDA-based parts of our system (SIFT computation (Wu, 2007) and SIFT descriptor matching).

In our system, we used a 10^6 visual word vocabulary tree trained on approximately 250M SIFT features from the Berlin dataset from (Frahm et al., 2010). For feature extraction, we relied on a GPU-based SIFT implementation (Wu, 2007), and extracted a maximum of 4096 SIFT features from each image. Then, for geometric verification, we enforced a minimum of 30 inlier matches in order for a pair of images to be considered successfully registered. Additionally, we ignored any image pair that had 70% of its inliers along the outer border of the image, as these matches were most frequently caused by watermarks. A recent work (Weyand et al., 2015) has proposed a learning-based method to detect these types of undesirable scenarios, so this could be incorporated in the future. Finally, when registering cameras to the already built 3D models, we enforced a minimum of 50 P3P (perspective-3-point) inliers.

Streaming Module	# Threads / # GPUs	Rate
Read Files from Disk	4 / -	120 Hz
Decode & Resize JPEGs	4 / -	177 Hz
Compute SIFT	8 / 8	138 Hz
Compute Visual Words	4 / -	434 Hz
Query Voc-Tree KNN	16 / -	4,475 Hz
Geometric Verification	16 / 8	261 Hz
Add Images to Voc-Tree	16 / -	14,485 Hz
Save SIFT to Disk	3 / -	186 Hz

Table 3.2: Performance of the streaming modules for the experiments using the city-scale datasets. The rate is the number of images, queries (voc-tree knn), or image pairs (geometric verification) processed per second.

In general it can be observed that our computation for the city-scale datasets is limited by the I/O bandwidth of our system (see Table 3.2), where we only reach a sustained disk read rate of 120 Hz when reading images at about 1024×768 resolution. For the world-scale dataset (Thomee et al., 2015) we leveraged seven high-performance hard drives, and used images at 640×480 resolution. In this case, disk I/O was no longer the bottleneck, and SIFT computation and geometric verification then became the limiting factors.

3.3.1.1 Single PC Implementation versus Distributed Processing

In our system, we chose to target a single PC implementation. One of the primary reasons for this is simplicity, in that the hurdles of development and debugging are limited to a single processing instance, as opposed to multiple instances running simultaneously and communicating over a network. Additionally, there is the issue of cost. Running our system on our largest dataset (approximately 96 million images (Thomee et al., 2015) processed over six days (Table 3.1)) would cost $1,000 - $2,000 at current rates for distributed computing instances (Amazon, 2015). This would be for only a single pass through the data, so additional passes for debugging and statistics purposes would increase that cost.

Nevertheless, our streaming paradigm could be adapted to run in a distributed environment. For instance, each distributed compute node could process its own portion of the stream. Then, whenever a new cluster is discovered, or a cluster's representation is updated, this new piece of information could be broadcast to the other nodes in the compute environment. The main challenge here will be communication and effectively sharing the state of the streaming system. For example, the set of registered clusters and their representations is constantly changing, so if each node keeps a copy of the entire representation, every change to a cluster would need to be broadcast to the entire network. Furthermore, the size of the cluster representations stored in memory can reach several tens or hundreds of gigabytes (depending on the discard rate, see Figure 3.15), so this would be a significant amount of data to duplicate to every node in the network. An alternative would be to store the entire streaming state on a central storage node, and each compute node could access it at runtime. However, this would also impose a high amount of communication burden on the network, as well as latency in accessing the shared representation. Therefore, a different design would be required in order efficiently share, distribute, or duplicate the streaming state among the nodes.

Another challenge would be the discovery of rarely-occurring related images. For instance, if each compute node is operating independently over its own stream, then similar images that are split across the different streams may be missed by the method if multiple similar images do not appear in the same stream. This problem becomes more pronounced as additional compute nodes are used in the network, as each node processes a progressively smaller amount of the stream. This problem is not unlike the limited discard window which is used for cluster discarding, in that any given cluster has a limited amount of time to register to additional images before it is discarded.

In summary, scaling the system to a distributed environment, while possible, would involve additional algorithmic consideration and significant engineering development. Specifically, the problems of dividing up the work (*e.g.* through dividing the input stream of images), the issue of sharing the current state of the clusters and their representations between different compute nodes, and adapting the algorithm to maintain comparable registrations rates would all need to be considered and addressed.

3.3.2 Results on Previous Datasets

The smallest of our datasets, the Roman Forum, was previously used by MatchMiner (Lou et al., 2012). Our system registered 45,341 images and had a connectivity entropy of 7.58 (lower is better; we refer to (Lou et al., 2012) for a motivation of this measure), compared to the 40,604 registered images and 11.62 entropy of MatchMiner. In contrast to our single PC, MatchMiner used a 53-node compute cluster and took 1.65 hours to discover the connected components in the dataset (Lou et al., 2012), whereas our single-machine system finished in 21 minutes for the streaming. There are several factors underlying the differences in results. For instance, the criteria for valid geometric verification (*i.e.* minimum required number of inliers, which was not reported by MatchMiner (Lou et al., 2012)) may have been different between the approaches. Additionally, MatchMiner used a much higher match budget, allowing an average of 20 match attempts per image, whereas we used used $k_v = 2$ for this and all our other experiments to ensure comparability across our different datasets. Our system does employ GPU computation for SIFT extraction (Wu, 2007) and SIFT descriptor matching (leading to greater efficiency in these modules), however MatchMiner does not include SIFT extraction and visual word computation in their timings at all, further emphasizing the efficiency of our approach. Overall, we achieve a comparable level of connectivity but at significantly lower computational cost.

Our second smallest dataset, Berlin, Germany, contains 2.7 million images and was obtained from the authors of Frahm et al. (2010). It was reported (Frahm et al., 2010) that, in the geometric cluster verification of Frahm et al. (2010), 124,317 images were registered overall for the dataset. In contrast, we register around 5.5 times as many images (*i.e.* 702,845 or 26% of the dataset, see Table 3.1) from the same data. When considering only the images registered to the 32 biggest reconstructed 3D models, we reconstruct 194,870 images, which is around 6.25 times the number of overall images reconstructed by Frahm et al. (2010) (31,190). The largest reconstructed model of Frahm et al. (2010) contained 3,158 images, whereas ours contains 32,745 images and is close to a kilometer long in the longest direction (shown in Figure 3.4). This significantly higher registration rate is a result of our significantly improved cluster representation and the streaming computation

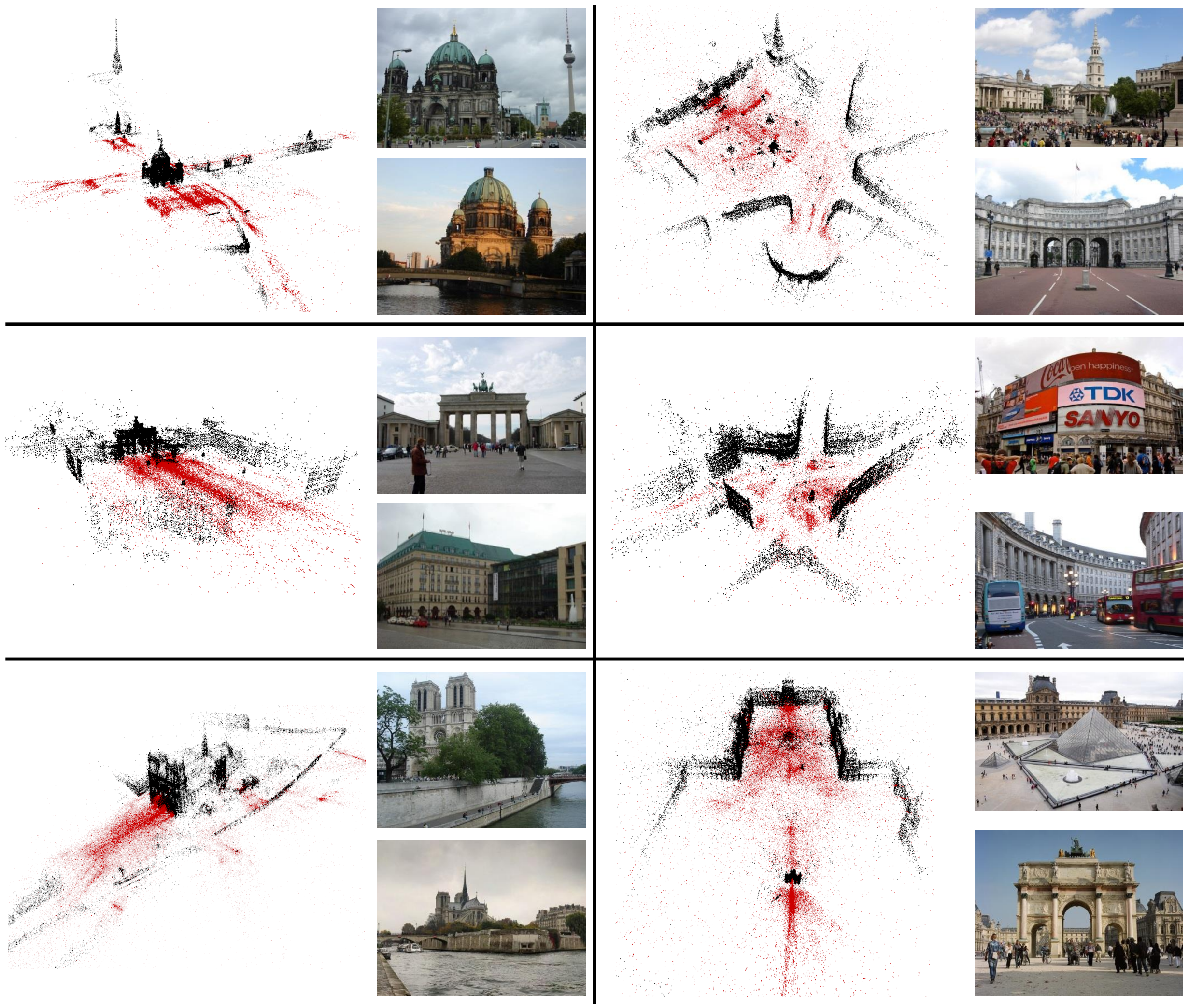

Figure 3.4: Sample SfM models output by our system on the city-scale datasets. From left to right, then top to bottom: Berliner Dom, Trafalgar Square, Brandenburg Gate, Piccadilly Circus, Notre Dame, and Louvre.

that readily obtains connected components. Frahm et al. (2010) report a computation time of 20.32 hours for the structure-from-motion part of their system. On the same machine we achieve a processing time of 14.61 hours for registering more than an order of magnitude more images for the same dataset.

3.3.3 Results on Previously-Unused Large-Scale Datasets

The third and fourth dataset we tested were datasets from Paris, with 10.3 million images, and from London, with 12.3 million. Both datasets were downloaded from Flickr. It can be seen

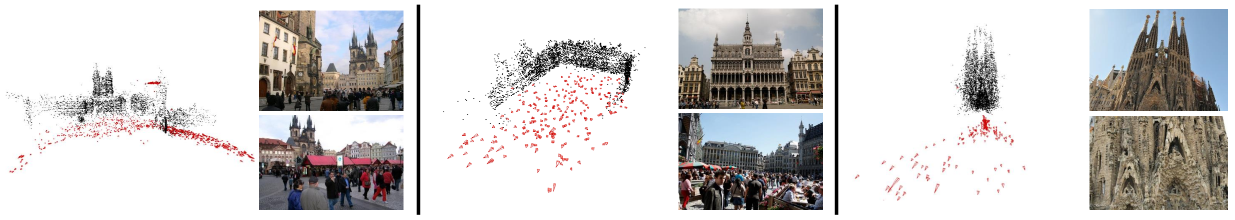
Figure 3.5: Sample SfM models output by our system on the Yahoo 100M world-scale dataset. From left to right: Prague, Brussels, and Sagrada Família.

that in both datasets our method reaches a registration rate of around one quarter of the images (Paris 24% registration rate and London 26%) which is similar to the 26% registration rate for the Berlin dataset. It can be seen that the computation rates for these datasets are also scaling linearly (less than 6% variation from linear). This underlines the scalability of our proposed method that reconstructs from an order of magnitude more image data than previously proposed methods while reaching state-of-the-art model completeness. Example data are shown in Figure 3.4 and the detailed statistics are provided in Table 3.1.

To demonstrate the true world-scale processing, we processed 96 million images spanning the globe from the Yahoo webscope dataset (Shamma, 2014; Thomee et al., 2015). The processing time was approximately 5.26 days. Our pipeline is the first system to be able to reconstruct from a world-scale dataset like this. Example models are shown in Figure 3.5 and the detailed statistics are provided in Table 3.1. This clearly demonstrates the scalability of our newly proposed reconstruction system enabling us to reconstruct the world in six days on a single computer. While we did register almost 1.5 million images, the generated reconstructions were smaller compared to the specific city-scale datasets (as the city-scale datasets have a denser sampling of images). Therefore, we skipped the iconic-image-based reconstruction, and instead used all of the images in the connected components directly.

3.3.4 Construction of Dataset with Approximate Ground-Truth Connectivity

In order to further evaluate the performance of the system, we constructed a dataset with approximate ground-truth connectivity. Here, we selected a subset of the Yahoo 100M dataset,

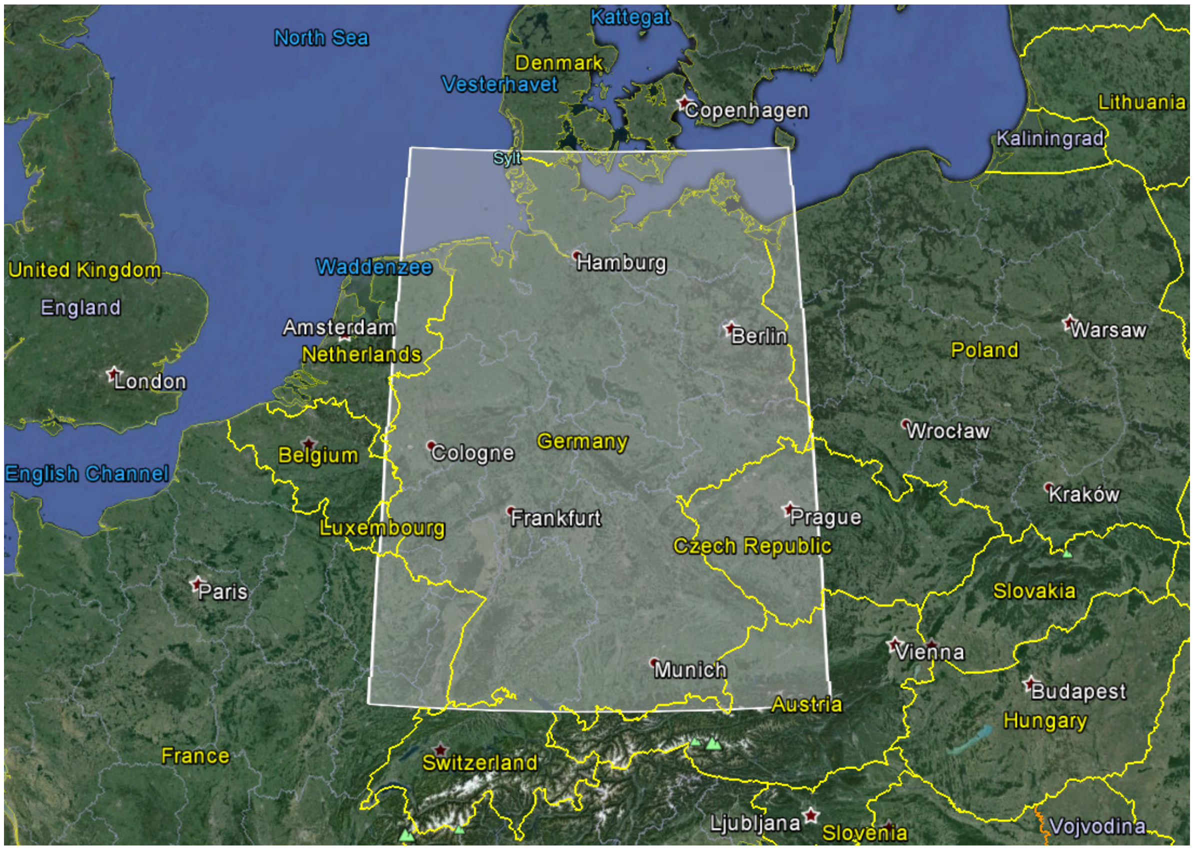

Figure 3.6: Map (Google, 2015) showing the geographical region used to create a subset of images from the Yahoo 100M dataset (Thomee et al., 2015) for computing approximate ground-truth connectivity.

in an attempt to recreate part of its behavior, but on a smaller scale. Specifically, by defining a geographical region surrounding Germany (spanning within $47.5 - 55°$ latitude and $6 - 15°$ longitude, see Figure 3.6), we extracted almost 2.3 million images (see Table 3.3).

In order to extract this geographical subset, we only used images which had been geotagged (almost half of the images in the Yahoo 100M dataset are geotagged (Thomee et al., 2015)). Therefore, it should be noted that the density of coverage in this geotagged subset is less than that of the full 100M dataset. However, it should serve as a reasonable test set of images, and allow us gain useful insights into the behavior of the system.

To determine the connectivity between the images, each image attempted geometric verification with its 1000 nearest neighbors (according to GPS location), or images within 1 kilometer, whichever

Number of Images					Number of Components		
Input	Valid	Register ≥ 2	Register ≥ 25	SfM ≥ 25	Register ≥ 2	Register ≥ 25	SfM ≥ 25
2,287,985	2,216,925	737,975	220,240	105,332	174,549	2,201	1,820

Table 3.3: Summary of the subset of the Yahoo 100M dataset (Thomee et al., 2015) used for computing approximate ground-truth connectivity. For the number of images in the Register or SfM columns, only those components which had greater than or equal to the listed number were counted. Likewise, the number of components columns report on the number of connected components or structure-from-motion models that had at least the listed number of images.

was fewer. This resulted in 807,579,942 unique image pairs which needed to perform geometric verification.

After performing this verification, there were over 700 thousand registered images (32% of the input dataset) distributed across 170 thousand connected components. However, when considering those components with at least 25 images, there were around 220 thousand registered images in 2,200 components. After structure-from-motion, 105 thousand of these images ended up in around 1,800 models (see Table 3.3).

Inspecting these results, more than half of the registered images did not end up in a reconstructed model. To gain an intuition into this, refer to Figure 3.7. Here, we see that a vast majority of the unreconstructed images correspond to sporting events, concerts, meetings, or other social events. Many of these events were captured by a single photographer, and as such, the photographs are either taken from a stationary location, or contain a watermark overlaid by the particular photographer. In these cases, stable scene geometry cannot be estimated by structure-from-motion and the images are discarded. In the few cases where the images appear to show usable content, these images may have failed the additional constraints imposed by the structure-from-motion system and were left unused.

To use this dataset for evaluation of the streaming system, we will only consider those connected components and reconstructed SfM models that have at least 25 images. Then, we will report on the fraction of this subset of images that was recovered by the streaming system under different parameter configurations.

Looking further at the connectivity of those connected components with at least 25 images, we see that a large fraction of the images only registered to a few neighboring images (see Figure 3.8).

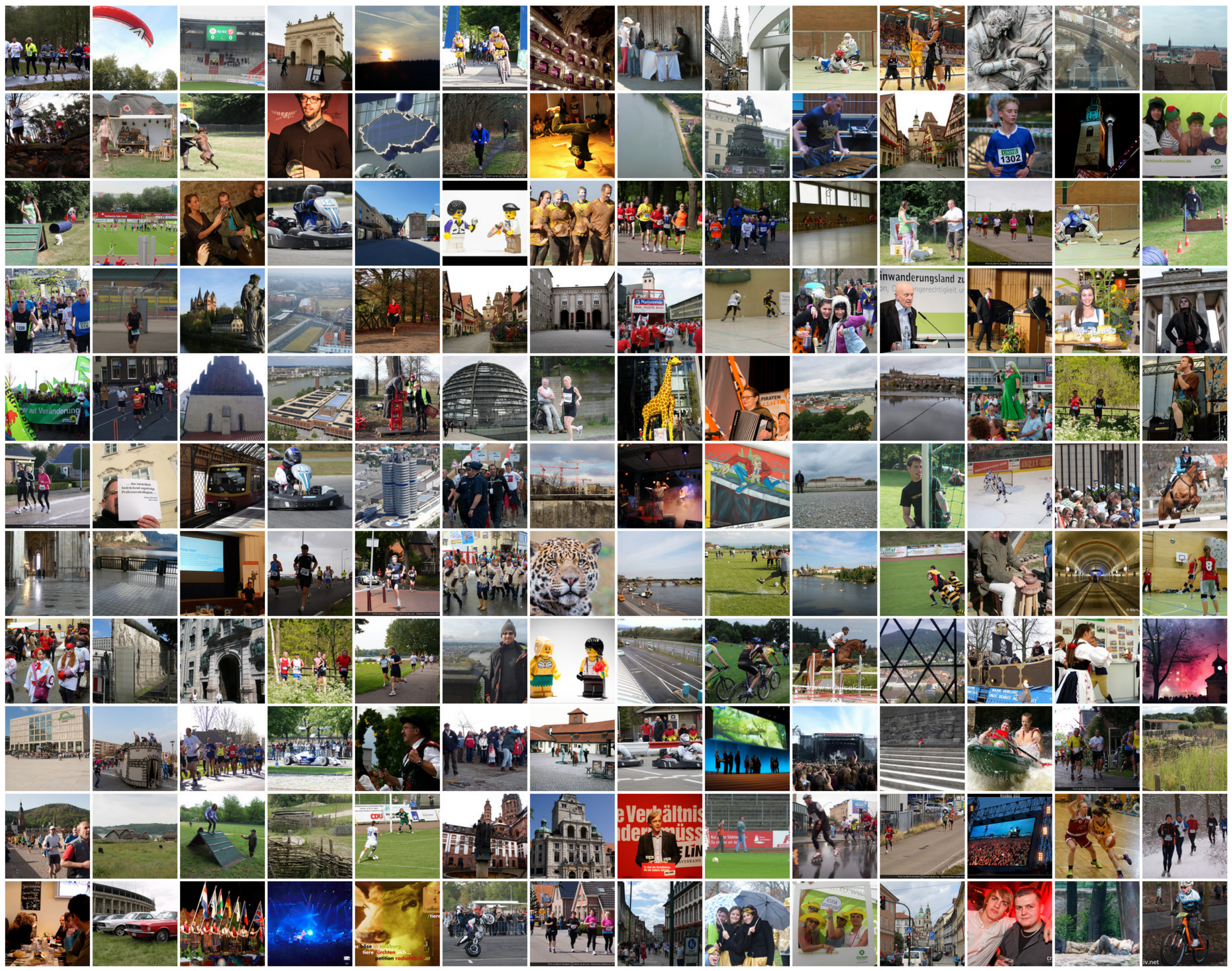
Figure 3.7: Example images that were registered by the streaming system but failed to be included in a reconstructed structure-from-motion model.

For instance, over 32,000 images are only connected (registered) to one or two other images in the considered subset. This means that our streaming system has to be very lucky in order to recover these photos, as in order for the image to be registered, one of its connected neighbors must be selected as an iconic image.

3.3.5 Effects of Parameter Configuration

Given the method as proposed, it is important to understand the behavior of various parts of the system, and the effects that various parameters and data configurations have on the results of the

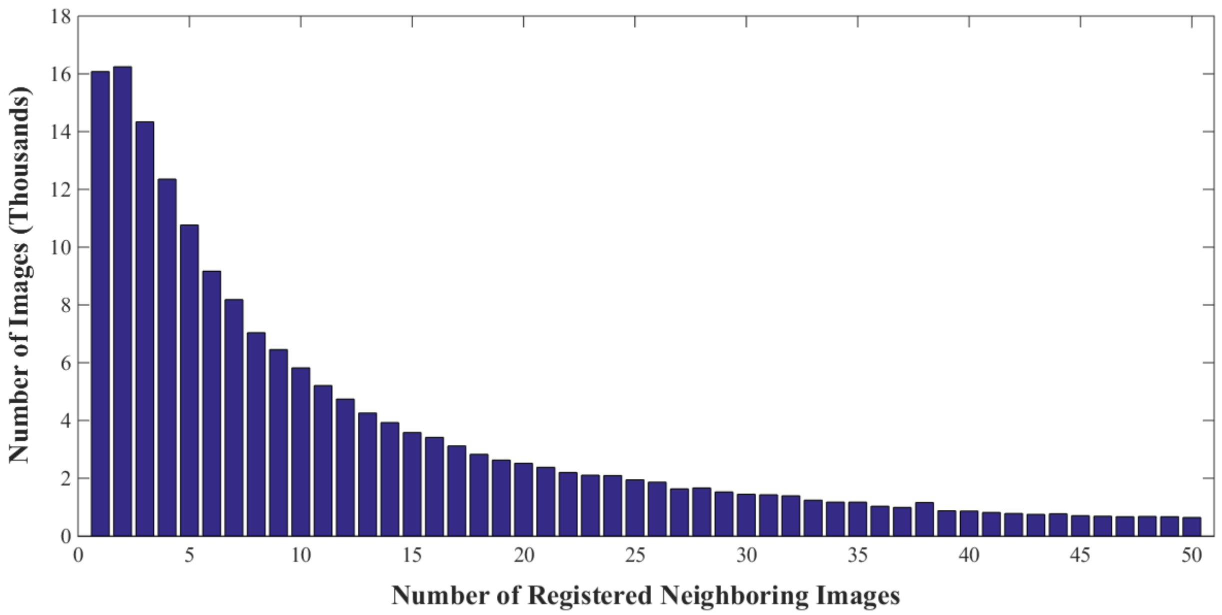

Figure 3.8: Histogram showing the distribution of the number of connected images per image in the approximate ground-truth connectivity subset of images (the histogram was artificially limited to a maximum of 50 neighboring images). Here, connected images are those that successfully pass geometric verification.

pipeline. Therefore, the following sections attempt to address and explore several of these behaviors, and discuss their results.

3.3.5.1 Effect of Image Order

One of the first tests we performed in evaluating the effects of different parameter configurations on the system was the effect that the image order had on the connected component discovery's results. To test this, we generated five random perturbations of the images in the Roman Forum dataset (Lou et al., 2012). Then, running the streaming system on these different perturbations, we recorded the total number of registered images. In all, there was less than 0.4% difference between the maximum and minimum number of registered images for the five different experimental runs.

3.3.5.2 Effect of Image Similarity Metric

Another test we performed was to determine the effect of relying on a bag-of-words representation instead of the GIST descriptor (Oliva and Torralba, 2001) as in (Frahm et al., 2010). To test this effect, we implemented a version of the streaming pipeline that relied on GIST descriptors for its image similarity metric. Here, as opposed to matching new streamed images to the iconic images with the most similar bag-of-words representations, the iconic images were chosen based on their similarity (cosine distance) using GIST.

To test the effect of GIST similarity, we utilized the Berlin dataset from Frahm et al. (2010). On this dataset, it is reported that the method of Frahm et al. (2010) registered around 124K images (4.6%). When using GIST similarity and our streaming paradigm, we register around 241K images (8.9%). Finally, when using the streaming paradigm and bag-of-words similarity, we register around 703K images (26%).

Looking at these results, we see that utilizing a streaming paradigm did improve the GIST-based results. However, one major different between these comparisons is that in Frahm et al. (2010), each cluster image had only one chance to register image, specifically, the iconic image of its cluster. In the streaming paradigm, each image is allowed to attempt registration with multiple nearest neighbors, increasing the likelihood of a successful registration. This gain in registration is not unlike the results in Raguram et al. (2011), where additional passes through the initially unregistered images yielded increased registration rates.

Nevertheless, looking at the difference between the GIST-based similarity and the bag-of-words based method, we see that the latter yields a significant improvement. This is not unexpected, as the GIST descriptor is a whole-image descriptor, and is affected by changes in composition (such as a translation or rotation of the image). The bag-of-words representation is not as affected by these changes, as while it represents the content in the entire image, it is based on a set of local feature descriptors, which are themselves scale and rotation invariant. Therefore, even if the composition of the image were to change, the bag-of-words representation would still have overlap as long as a significant fraction of the local features are still visible.

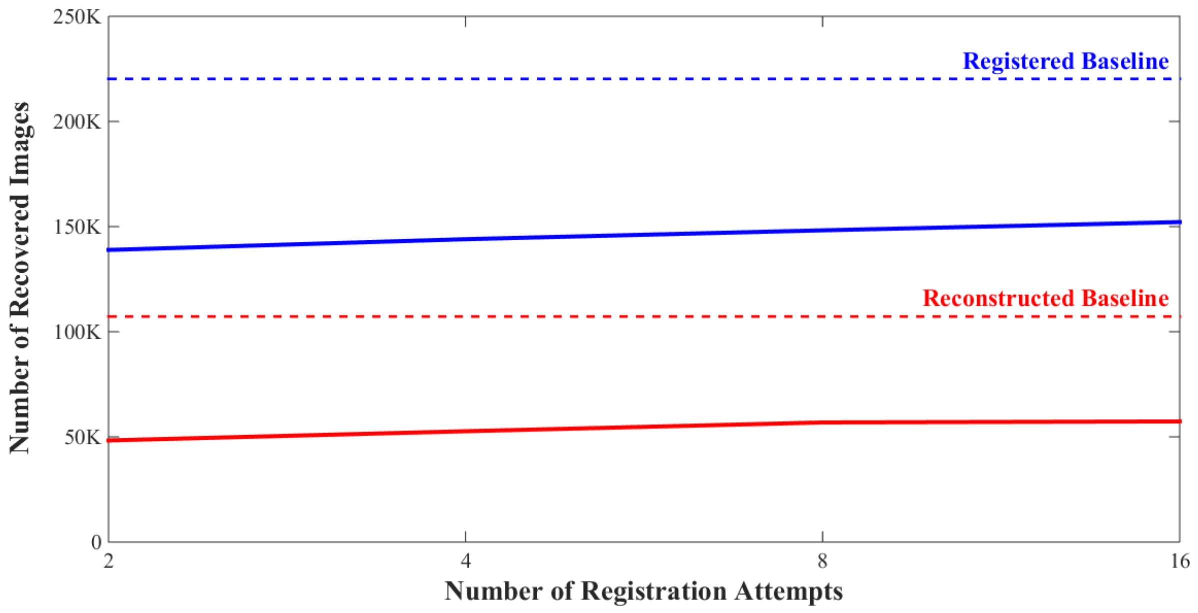

Figure 3.9: Graph showing the effect of the number of registration attempts per image on the amount of registered images recovered by the streaming system.

Number of Registration Attempts	2	4	8	16
Percent of Registered Baseline Recovered	63.1%	65.4%	67.3%	69.1%
Percent of Reconstructed Baseline Recovered	45.1%	49.1%	53.0%	53.5%
Streaming Runtime (hours)	2.30	4.60	9.61	21.71

Table 3.4: Effect of the number of registration attempts per image on the performance of the streaming connected component discovery.

3.3.5.3 Effect of Number of Registration Attempts per Image

One of the main parameters of the system is the number of registration attempts performed per image (k_v from Section 3.2.1.1). To test the effect of this parameter, we ran the streaming system with different values of k_v (2, 4, 8, and 16) on the approximate ground-truth connectivity dataset from Section 3.3.4. Additionally, we used the default imagery with a maximum resolution of 640 pixels, a maximum of 4K SIFT features per image, and a discard rate (w) of 200K.

After performing connected component discovery using different values for k_v, we see that it has a very minimal impact on the amount of recovered registered and reconstructed images (see Figure 3.9). Furthermore, increasing the value of k_v increases the overall streaming runtime at a superlinear rate (see Table 3.4). The reasoning for this is that performing geometric verification (specifically the RANSAC component of it), is the main bottleneck of the system. So, increasing the number of times that geometric verification needs to be run will have a direct impact on the overall runtime. Additionally, the success (being able to find a valid camera transformation) of geometric verification diminishes as additional nearest neighbors are evaluated (as image retrieval has already sorted the results based on their likelihood of being similar). Therefore, this decreased success rate causes RANSAC to more frequently run for the full number iterations, yielding a higher computation time. Finally, by discovering additional connections within the data (however few they may be), additional clusters will be indexed in the vocabulary tree, which will increase the amount of time it takes to perform image retrieval.

3.3.5.4 Effect of Discard Rate

One of the other main parameters of the system is the discard rate (w from Section 3.2.1.4). To test the effect of this parameter, we leveraged the approximate ground-truth connectivity dataset and ran the system with different values of w (100K, 200K, 400K, 800K, and 1600K). In the first test, we used the same 640 pixel resolution imagery, a maximum of 4K SIFT features per image, and two registration attempts (k_v) per image.

After performing connected component discovery using different values of w, we see that it has a more substantial impact on the number of recovered images than varying k_v (see Figure 3.10). Additionally, increasing w yields a sublinear increase in the runtime of the streaming connected component discovery (see Table 3.5), which is significantly better than the superlinear behavior of k_v. The reason for the increase in runtime is that every time the discard rate is increased, additional images are indexed in the vocabulary tree causing the image retrieval operation to have a higher computational cost. However, as image retrieval is not a main bottleneck of the system, this increase in runtime has a less pronounced effect on the overall runtime. Another cause for the increase in runtime is the increased registration rate. Just as with k_v, increasing the number of discovered connections within the data causes an increased number of clusters to be indexed by the vocabulary tree, which increases the overall runtime of the system.

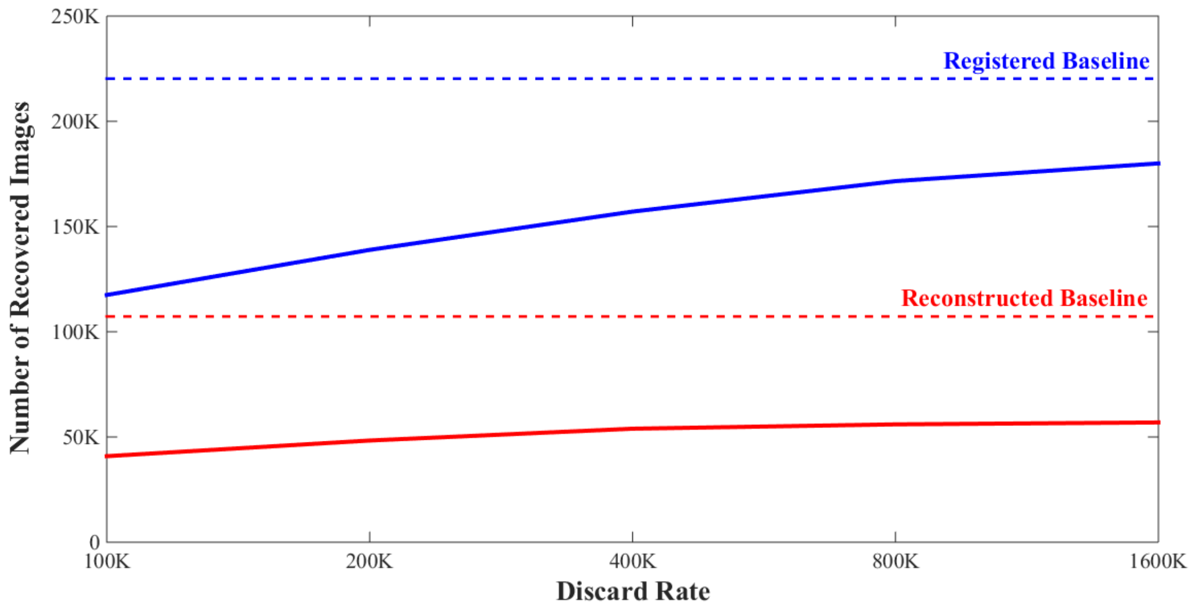

Figure 3.10: Graph showing the effect of the discard rate on the amount of registered images recovered by the streaming system.

Discard Rate	100K	200K	400K	800K	1600K
Percent of Registered Baseline Recovered	53.3%	63.1%	71.3%	77.9%	81.7%
Percent of Reconstructed Baseline Recovered	38.1%	45.1%	50.3%	52.2%	53.1%
Streaming Runtime (hours)	2.22	2.30	2.60	3.11	3.44

Table 3.5: Effect of the discard rate on the performance of the streaming connected component discovery.

As an additional test, we also ran the system with the same configuration described above, but with eight registration attempts per image. Here (refer to Figure 3.11 and Table 3.6), the retrieval rate demonstrates the same overall trend as with two registration attempts per image (in that the retrieval increases rather dramatically). Furthermore, the retrieval rate with eight registration attempts per image is consistently higher than that with only two attempts per image. This is not unexpected, as the tests from Section 3.3.5.3 indicated that an increased number of registration attempts yielded an increased retrieval rate.

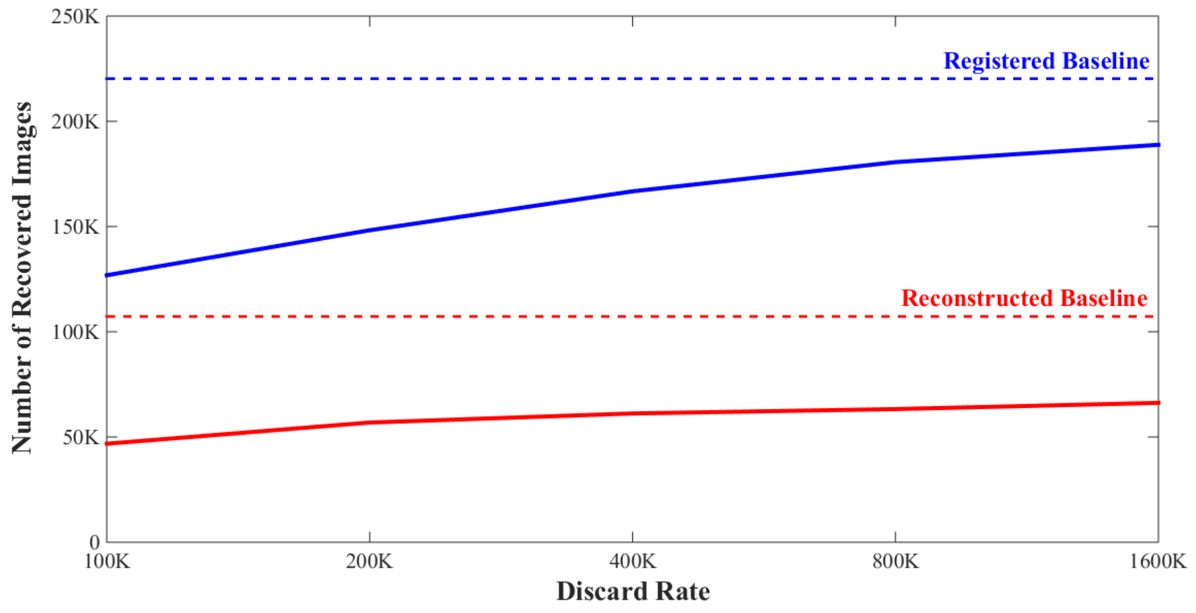

Figure 3.11: Graph showing the effect of the discard rate on the amount of registered images recovered by the streaming system when increasing the number of registration attempts per image to 8.

Discard Rate	100K	200K	400K	800K	1600K
Percent of Registered Baseline Recovered	57.6%	67.3%	75.7%	82.0%	85.7%
Percent of Reconstructed Baseline Recovered	43.6%	53.0%	57.0%	59.0%	61.7%

Table 3.6: Effect of the discard rate on the performance of the streaming connected component discovery when increasing the number of registration attempts per image to 8.

3.3.5.5 Effect of Number of SIFT Features

Another parameter of the system, though one that is not obviously as relevant, is the number of SIFT features (Lowe, 2004; Wu, 2007) that we extract from each image and use for the following tasks in the system (image retrieval, geometric verification, structure-from-motion, *etc.*). To test this parameter, we leveraged the approximate ground-truth connectivity dataset, but downloaded the original resolution images from Flickr. The reason for this is that with imagery with a maximum dimension of 640 pixels, it can be difficult to extract a high number of features from each image. Therefore, when running the streaming system, each loaded original image is resized at runtime to have a maximum dimension of 1600 pixels (which still allows a large number of features to be discovered). This resizing was done to avoid the expensive operation (in terms of runtime) of

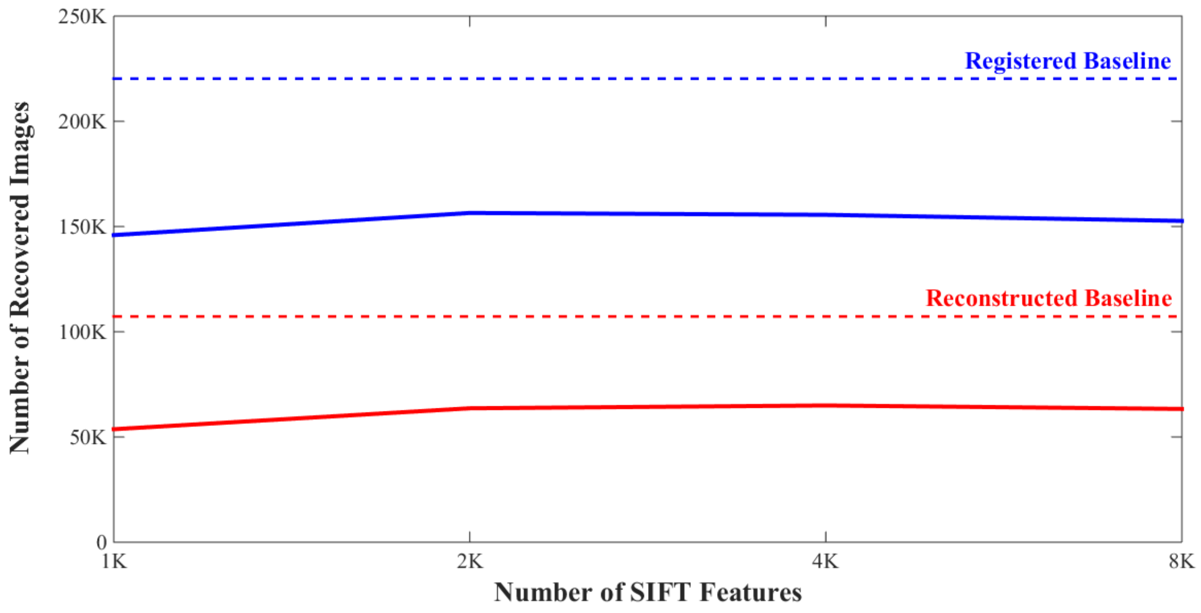

Figure 3.12: Graph showing the effect of the number of computed SIFT features per image on the amount of registered images recovered by the streaming system.

Number of SIFT Features	1K	2K	4K	8K
Percent of Registered Baseline Recovered	66.3%	71.0%	70.6%	69.3%
Percent of Reconstructed Baseline Recovered	50.1%	59.3%	60.6%	59.0%

Table 3.7: Effect of the number of SIFT features per image on the performance of the streaming connected component discovery.

copying a very high resolution image to the GPU and performing SIFT extraction. Additionally, when running the system, we used a 200K discard rate, and two registration attempts per image.

Upon performing connected component discovery and structure-from-motion using various numbers of SIFT features per image (1K, 2K, 4K, and 8K), we discovered that increasing the number of features is useful only until a certain point, at which time increased numbers yield worse results (see Figure 3.12 and Table 3.7). Specifically, the best performance in terms of recovered registered images was with 2K extracted SIFT features per image, with 4K being essentially identical. For the reconstruction, 4K SIFT features per image yielded the highest recovery rate. It is understandable that using 1K SIFT features per image would yield worse results, as there are fewer features for geometric verification (which enforces a hard constraint on the minimum number of inliers). However, it not yet understood why increasing the number of features per image to 8K

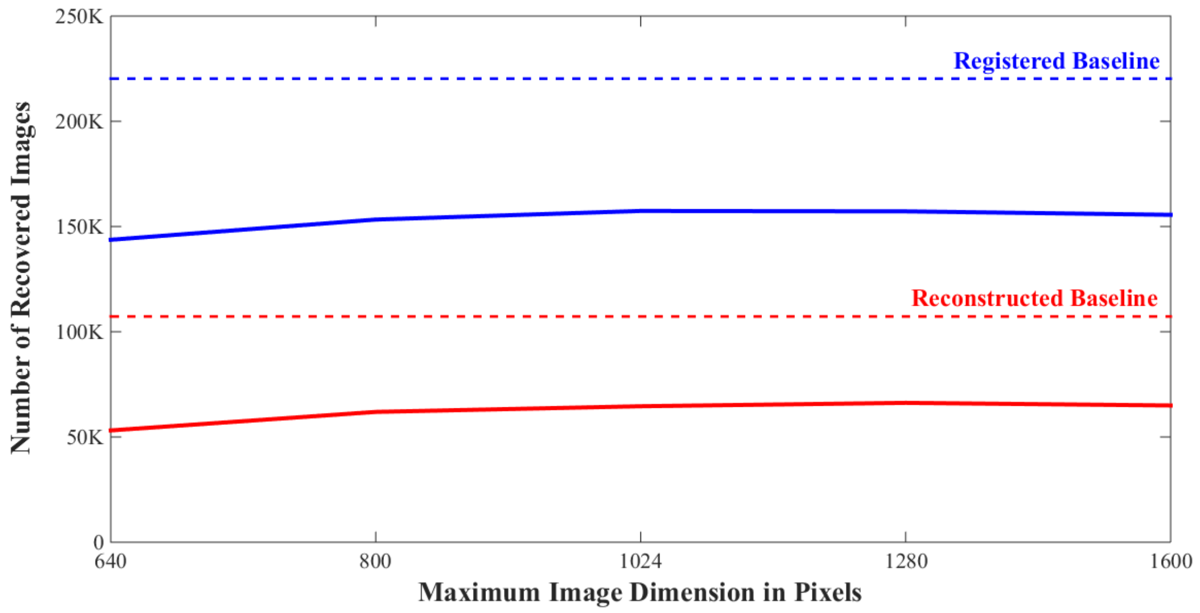

Figure 3.13: Graph showing the effect of image resolution (as represented by the maximum dimension of an image in pixels) on the amount of registered images recovered by the streaming system.

Maximum Image Dimension in Pixels	640	800	1024	1280	1600
Percent of Registered Baseline Recovered	65.3%	69.6%	71.5%	71.4%	70.6%
Percent of Reconstructed Baseline Recovered	49.5%	57.7%	60.2%	61.7%	60.6%

Table 3.8: Effect of image resolution on the performance of the streaming connected component discovery.

would have a negative impact on the amount of recovered images. For instance, it is possible that increasing the number of elements in the bag-of-words representation has a negative impact on retrieval performance. This would be because for each new visual word in the representation, new candidate images are considered in the retrieval results, which could harm the results if too many confusing or outlier images are introduced.

3.3.5.6 Effect of Image Resolution

Similar to the number of SIFT features per image (see Section 3.3.5.5), the resolution of the processed imagery also has an effect on the performance of the streaming system. To test the effect of image resolution, we once again used the downloaded original resolution imagery for the test subset of images from Section 3.3.4. Then, we ran the streaming system, but varied the maximum image dimension (using values of 640, 800, 1024, 1280, and 1600). This has the effect that when images are loaded from disk, they are resized so that their maximum image dimension does not exceed the specified value. For the other parameters, we used two registration attempts per image, a 200K discard rate, and a maximum of 4K SIFT features per image.

After performing the test, we recorded that, like the number of SIFT features per image, increasing the maximum image dimension improves retrieval, but only to a certain point (as indicated in Figure 3.13 and Table 3.8). After this, increasing the image resolution yielded a decreased retrieval performance. This is similar to the behavior of changing the number of SIFT features (Section 3.3.5.5) and is not entirely surprising, as an increased image resolution allows for an increased number of SIFT features to be extracted per image (though it was limited to 4K). However, even if the maximum number of SIFT features were increased to 8K, this would not yield an increase in performance as illustrated by the tests in Section 3.3.5.5.

3.3.5.7 Effect of Iconic Image Set

Another interesting behavior of the system is the effect on the selected set of iconic images. For instance, as the number of registration attempts is increased, the total number of recovered (registered) images also increases (as we already observed in Section 3.3.5.3). However, as the total number of registered images increases, there are a certain number of images that are no longer recovered. For example, we see in Table 3.9 that almost 139 thousand images are recovered when there are two registration attempts per image. However, in the following tests with different values for k_v, only around 135 or 136 thousand of those images are recovered again. Likewise, of the newly discovered images for $k_v = 4$ and $k_v = 8$, only a fraction of them are recovered again in a following test. A similar pattern holds for the discard rate, as seen in Table 3.10.

The reason for this variation stems from the fact that the iconic image set used to register the streamed images changes between the different test configurations. For instance, while a particular set of iconic images may be able to register to a greater number of images in the dataset, their

Registration Attempts	Number of Recovered Images				
	Total	$k_v = 2$	$k_v = 4$	$k_v = 8$	$k_v = 16$
$k_v = 2$	138,903	**138,903**			
$k_v = 4$	144,017	134,872	**9,145**		
$k_v = 8$	148,219	135,205	7,068	**5,946**	
$k_v = 16$	152,146	135,626	7,388	4,369	**4,763**
Total Number of Unique Recovered Images = 158,757					

Table 3.9: Statistics for the sets of recovered images when varying the number of registration attempts per image. Each row represents a different test using a particular number of registration attempts, and the columns of the row ($k_v = 2 \ldots 16$) report on how many of the recovered images were either newly discovered in this test (the bold values) or had already been discovered in a prior test. Here, only those images belonging to connected components with $\geq$ 25 images are considered.

Discard Rate	Number of Recovered Images					
	Total	$w =$100K	$w =$200K	$w =$400K	$w =$800K	$w =$1600K
$w =$100K	117,475	**117,475**				
$w =$200K	138,903	112,289	**26,614**			
$w =$400K	157,096	112,975	23,679	**20,442**		
$w =$800K	171,564	113,922	24,160	18,387	**15,095**	
$w =$1600K	179,985	114,179	24,311	18,469	13,541	**9,485**
Total Number of Unique Recovered Images = 189,111						

Table 3.10: Statistics for the sets of recovered images when varying the discard rate. Each row represents a different test using a particular discard rate, and the columns of the row ($w =$ 100K…1600K) report on how many of the recovered images were either newly discovered in this test (the bold values) or had already been discovered in a prior test. Here, only those images belonging to connected components with $\geq$ 25 images are considered.

particular SIFT features and bag-of-words representations may preclude them from registering to a small set of images that would otherwise be recoverable using a different set of iconic images. Nevertheless, the coverage provided by the iconic image sets does improve with increasing parameter value (for both k_v and w), evidenced by the increasing number of recovered images (Tables 3.9 and 3.10).

3.3.6 Probabilistic Model for Streaming Paradigm

Streaming paradigms are not new in the context of data processing (Toivonen, 1994; Savasere et al., 1995; Manku and Motwani, 2002; Karp et al., 2003; Isenburg and Lindstrom, 2005; Cormode and Muthukrishnan, 2005b; Gaber et al., 2005). Typically, these methods either explicitly provide or are amenable to deriving a probabilistic or theoretical model for their behavior, such as limits on the amount of memory usage or guarantees on the amount of possible error that is to be expected in the results. With a similar goal, this section aims to motivate a probabilistic model for the streaming paradigm of this system.

We model our system following the framework of Manku and Motwani (2002), which targets the domain of approximate frequency counts for data streams. In their work (Manku and Motwani, 2002), the goal is to stream through a large set of integers in a single pass, and at the end of the processing, output the set of integers (and their frequencies) which occur above a predefined frequency. We can use this abstraction to model the behavior of our method, in that we want to stream through a set of images, and output those images (clusters of images represented by an iconic image) which occur frequently in the data (have a large enough cluster size).

In the case of determining the frequency of integers in a data stream, expressing the association between elements is trivial. Specifically, if the two elements have the same integer value, then they are equivalent elements and count toward the same frequency. Furthermore, in order to determine if an element has been seen before, efficient methods like hash-tables or sketches have been used (Charikar et al., 2002; Cormode and Muthukrishnan, 2005a; Goyal and Daumé, 2011).

When considering streams of imagery, defining and discovering equivalence between images is not as straightforward as it is for integers. In our application, we define two images as being equivalent if they are able to successfully register to each other during geometric verfication (with a predefined minimum number of inliers, Sections 3.2.1 and 3.3.1). Furthermore, we determine if an image (the content it contains) has been seen before through the use of a vocabulary tree and image retrieval techniques (Section 3.2.1.1).

With these mechanisms of image retrieval and geometric verification, we define a cluster as a set of images that all successfully register to a common image, the iconic image of the cluster (see Section 3.2.1). Given this definition, it is important to note that given a large set of interconnected images, there is not one unique division of those images into clusters. There may be many such divisions, just as there are many possible outcomes when running k-means or other randomly-initialized clustering methods (Arthur and Vassilvitskii, 2007). Nevertheless, given a set of images and a graph of their ground-truth connectivity, a subset of them that are all within a path length of 1 from a common node could be selected, and evaluated for their recovery by the streaming system. In this setup, as opposed to determining if the exact same cluster of images was formed (as we just described that there are many possible clusterings), we can evaluate what fraction of that cluster of images appeared in any of the output clusters.

3.3.6.1 Single versus Multi-Pass Algorithms

While several streaming algorithms employ two or more passes over the data in order to achieve accurate and robust results (Toivonen, 1994; Savasere et al., 1995; Karp et al., 2003), we chose a single-pass implementation. One of the first reasons for this choice is that computational efficiency is always an important goal. So, if reading through the data a second time would necessitate an increase in runtime (disk I/O was the bottleneck for the processing of the city-scale datasets, see Table 3.2), then attempting to achieve the best possible performance in a single pass is a valid goal. Furthermore, by focusing first on a single pass and optimizing its performance, the benefits of a second pass can be better compared at a later date. Finally, while the algorithm in its current state conforms to an offline processing model, restricting ourselves to a single pass allows for a much easier transition to an online processing model in the future where data would be continually input into the system (such as an infinite stream of images, see Section 3.3.7).

Another consideration is the target application. If the focus of the system to recover the highest number of images possible, then a second pass will greatly assist in achieving this goal. However, if the goal is to quickly process the data and perform structure-from-motion on only the most popular landmarks, then some trade-offs can be made in order to achieve this aim. Our system strikes a balance between the two by providing an estimate of the recovery rate of the method (Section 3.3.6.4), while limiting computation to an efficient first pass. For further discussion on how to adapt the proposed system to leverage a multi-pass algorithm, refer to Section 6.1.1.

3.3.6.2 Formulation

Using a notation similar to Manku and Motwani (2002), we will now introduce the formulation used to describe our system.

Given a stream of images of current length M, we would like to recover all clusters that have a size of at least sM, where s is some value in the range $(0, 1)$. Additionally, we would like to incorporate some guarantee about the accuracy of the recovered set of clusters, such that the clusters discovered a sufficient portion of their member images. To add these constraints, we impose the additional parameter ϵ also in the range $(0, 1)$, such that the number of unrecovered images from a particular a cluster is at most ϵM (where $\epsilon < s$).

Within the domain of streams of integers, the above definitions would be strict guarantees, and there would be no false negatives in the output. However, when dealing with images, we rely on the concepts of image retrieval and geometric verification to determine the similarity (which elements in the stream are potentially related) and equality (which elements in the stream represent the same content) of two images. Therefore, to account for these mechanisms, we introduce two additional terms. The first, p_r, is the probability that given a query image, an image from the same cluster will be returned by image retrieval (assuming that such an image exists in the retrieval database). The second, p_v, is the probably that two images from the same cluster will successfully pass geometric verification.

Typical values for p_r can range widely, depending on the properties of the dataset, the underlying feature's robustness to transformations, or the complexity of the retrieval system (Philbin et al., 2007; Chum et al., 2007; Zhang et al., 2011; Jégou and Chum, 2012; Arandjelović and Zisserman, 2012). However, in one instance, a reference system achieved a 90.6% accuracy when inspecting the correctness of only the first returned result (Nistér and Stewénius, 2006).

For the value of p_v, as long as there are a sufficiently high number of inlier feature correspondences, we can expect p_v to equal the confidence used in our RANSAC algorithm (which would typically be set at 95% - 99%). However, we need to account for the fact that we artificially limit the number of RANSAC iterations in order to achieve higher efficiency (Section 3.2.1.1). Specifically, when limiting RANSAC to 400 iterations and utilizing the 5-point estimation method (Nistér, 2003), we can, with 99% confidence, recover an underlying transform that is supported by at least 40.9% of the correspondences (see Equation (3.1), which is based on the standard RANSAC termination criteria (Fischler and Bolles, 1981; Raguram et al., 2013)).

$$\left(1 - \exp\left(\frac{\log(1 - 0.99)}{400}\right)\right)^{1/5} = 0.4090 \tag{3.1}$$

In many cases, two similar images will have at least this many inliers (as observed by Frahm et al. (2010)), so the impact on the confidence will be negligible.

As proposed in Section 3.2.1.4, we will rely on a discard rate w in order to control the amount of memory used by our system. Specifically, w is set to be equal to $1/\epsilon$. Whenever a cluster is formed (as a single, unregistered image), it stores its position (c_i) within the stream. Then, at regular intervals of w, the size of the cluster c_m is evaluated, and if the following inequality is true

$$c_m < \frac{M - c_i}{w} \tag{3.2}$$

then the cluster is discarded. As in Section 3.2.1.4, this equates to a cluster maintaining an average growth rate that is at least one new image registered per w images processed in the stream since the cluster's inception.

With this formulation, we will now discuss the bounds of the memory usage of the system, as well as the registration completeness that it attains.

3.3.6.3 Memory Usage

With the formulation presented in Section 3.3.6.2, we can show that the streaming paradigm will maintain at most $(1/\epsilon)\log(\epsilon M)$ cluster representations in memory.

To motivate the above claim, let us assume for the moment that the discard criteria of Inequality (3.2) was modified so that instead of computing the precise difference between the cluster's formation and the current stream length, the stream is partitioned into windows of width w (aligned to the beginning of the stream), and the discard criteria is enforced at window boundaries (as in Manku and Motwani (2002)). In this manner, a cluster is expected to grow in size by one (on average) by the time the next window boundary is encountered. Note that this is more restrictive than the criteria in Inequality (3.2), and will provide a shorter opportunity for all except those clusters which appear directly after a window boundary.

Given this temporary definition of the discard criteria, let B be the current number of windows that have been processed so far in the stream, and let d_i be the number of current clusters that were initially created during window $B - i + 1$, where i is in the range $[1, B]$. Any such cluster that counts toward d_i must have a size of at least i images, as the cluster corresponds to a range of windows from $B - i + 1$ through B, and would have been discarded if its size was not at least equal to i. With this observation, we can form the following inequality

$$\sum_{i=1}^{j} id_i \leq jw \qquad \text{for} \quad j = 1, 2, \ldots, B \tag{3.3}$$

which defines how the $jw = M$ images from the stream have been partitioned into the d_i clusters of size i. We would like to show that the number of clusters in memory is bound by

$$\sum_{i=1}^{j} d_i \leq \sum_{i=1}^{j} \frac{w}{i} \qquad \text{for} \quad j = 1, 2, \ldots, B \tag{3.4}$$

and will prove this by induction.

Inspecting Inequality (3.4), the base case of $j = 1$ is trivially true given Inequality (3.3). Assuming, for induction, that Inequality (3.4) is true for $j = 1, 2, \ldots, p - 1$, we will prove that it is also true for $j = p$, and complete the induction step. Summing Inequality (3.3) for $j = p$ with Inequality (3.4) for $j = 1, 2, \ldots, p - 1$ yields

$$\sum_{i=1}^{p} id_i + \sum_{i=1}^{1} d_i + \sum_{i=1}^{2} d_i + \cdots + \sum_{i=1}^{p-1} d_i \leq pw + \sum_{i=1}^{1} \frac{w}{i} + \sum_{i=1}^{2} \frac{w}{i} + \cdots + \sum_{i=1}^{p-1} \frac{w}{i} \tag{3.5}$$

which can be combined and simplified in the following manner:

$$\sum_{i=1}^{p} id_i + \sum_{i=1}^{p-1}(p-i)d_i \;\le\; pw + \sum_{i=1}^{p-1}\frac{(p-i)w}{i} \tag{3.6}$$

$$\sum_{i=1}^{p} pd_i \;\le\; pw + \sum_{i=1}^{p-1}\frac{pw}{i} \;-\; \sum_{i=1}^{p-1} w \tag{3.7}$$

$$p\sum_{i=1}^{p} d_i \;\le\; pw + \sum_{i=1}^{p-1}\frac{pw}{i} \;-\; (p-1)w \tag{3.8}$$

$$p\sum_{i=1}^{p} d_i \;\le\; \sum_{i=1}^{p-1}\frac{pw}{i} \;+\; w \tag{3.9}$$

$$p\sum_{i=1}^{p} d_i \;\le\; \sum_{i=1}^{p}\frac{pw}{i} \tag{3.10}$$

$$\sum_{i=1}^{p} d_i \;\le\; \sum_{i=1}^{p}\frac{w}{i} \tag{3.11}$$

which is now equal to Inequality (3.4) for $p = j$, completing the induction.

Given that the number of clusters in memory is $\sum_{i=1}^{B} d_i$ from Inequality (3.4), we can now conclude that the total number of clusters in memory is bounded by

$$\sum_{i=1}^{B} d_i \;\le\; \sum_{i=1}^{B}\frac{w}{i} \;=\; w\sum_{i=1}^{B}\frac{1}{i} \;<\; w\log(B)+1 \;=\; w\log(\epsilon M)+1 \;=\; \frac{1}{\epsilon}\log(\epsilon M)+1 \tag{3.12}$$

as the summation has the form of a harmonic series.

To motivate this derivation, we made the assumption that the discard windows existed at fixed positions within the stream, and clusters were discarded at window boundaries. However, our window boundaries are defined relative to a cluster's position of formation. Fortunately, this does not impact the above analysis, as any cluster that would have have been discarded in a fixed window boundary scheme, will have been discarded by our method (assuming that it did not grow) by the time the next fixed window boundary would have been encountered. If the cluster does grow before the next window boundary, then both methods consume the same amount of memory at that window

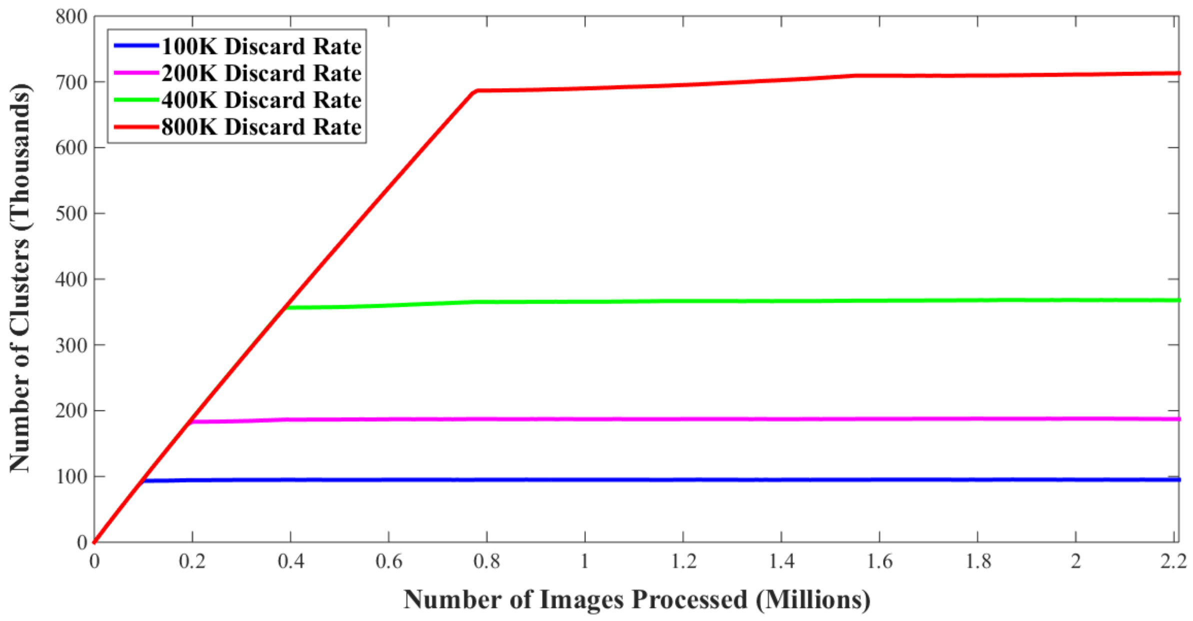

Figure 3.14: Graph showing the number of clusters stored in the memory of the streaming system over time for different discard rates.

boundary, as one method will store the grown cluster, and the other will have a cluster of size one. Conceptualized another way, the average rate of growth of a cluster must be the same under the two schemes in order for the cluster to avoid being discarded. Our method simply offsets the window boundaries for clusters that were formed at a later position within the stream.

Also, notice that the bound on the number of clusters is not impacted by the probability of successful image retrieval or geometric verification (p_r or p_v). This is because the above analysis is agnostic to the actual size of the clusters that are recovered (which would be impacted by p_r and p_v). Instead, the derivation poses a maximum bound on the number of clusters that could exist, under any possible scenario.

To visualize the actual behavior of the memory usage, we ran the streaming system on the dataset of Section 3.3.4, and recorded both the number of active clusters (Figure 3.14) and the amount of physical memory (RAM) (Figure 3.15) used by the system. Here, we see that both the number of clusters and the amount of RAM increases linearly until a number of images equal to the discard rate have been processed. At this point, the number of clusters levels off, while the memory usage rises at a minute rate.

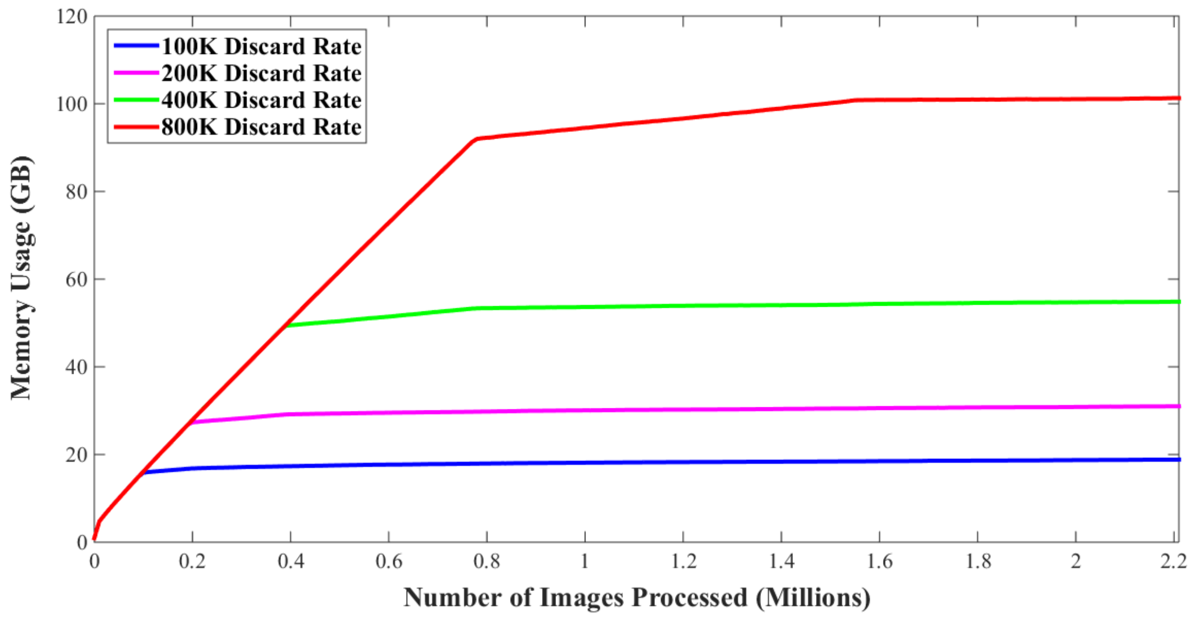

Figure 3.15: Graph showing the memory usage of the streaming system over time for different discard rates.

It is not surprising that the number of clusters maintains an almost constant value (Figure 3.14), as Manku and Motwani (2002) made the observation that if the streamed elements (images) occur independently at random according to some underlying fixed probability distribution, then the number of clusters can be bounded above by a constant value. Furthermore, for real-world datasets, as long as images with very low frequency occur approximately uniformly at random, then the constant number of clusters claim is still true.

There is the additional benefit of our system that any missed registrations due to image retrieval or geometric verification (p_r or p_v) can be be corrected by cluster merging (Section 3.2.1.3), which will remove the additional cluster representation from memory and help maintain a lower memory budget. To test the effect that cluster merging has on the number of clusters stored in memory, we tested our system with cluster merging both enabled and disabled, and show the results in Figure 3.16. Here, we see that while cluster merging does decrease the number of clusters stored in memory, it has a minimal impact, only decreasing the number of clusters by around 0.4%.

For the physical memory usage (Figure 3.15), it is also not surprising that it increases slightly over time. While the only cluster representations stored in memory are for those clusters that have

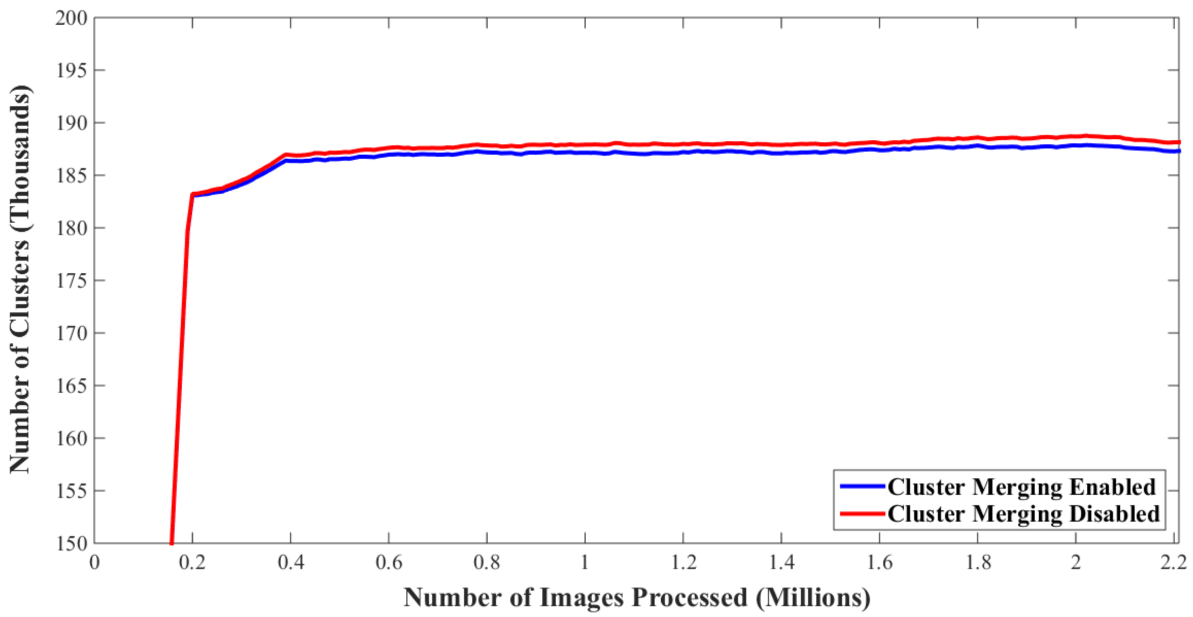

Figure 3.16: Graph showing the effect of cluster merging on the number of clusters stored in memory over time during the execution of the streaming system.

not yet been discarded, the current implementation of the system still keeps some information around for discarded clusters. Specifically, the connectivity information and image indices for any cluster belonging to a connected component of size ≥ 2 is kept in memory and saved to a file at the end of the processing. Additionally, the size of a cluster's representation can grow over time, as additional visual words are used to augment its representation (see Section 3.2.1.2).

3.3.6.4 Registration Completeness

Given the above formulation of the problem (Section 3.3.6.2), we would like to make guarantees about the completeness of the recovered image clusters. As before, we seek to discover clusters with size $\geq sM$, with $s > \epsilon$. At the end of the streaming, the amount of error in each cluster due to the discard strategy is bounded above by $\epsilon M = M/w = B$. This is because a cluster could be missing at most one image per discard window that occurred before the current position within the stream, otherwise, the cluster would not have been discarded and would still exist in memory. Therefore, for a cluster of size sM, we can expect to recover at minimum $sM - \epsilon M = M(s - \epsilon)$ images from it. However, this analysis is only based on the effects of the discard strategy, and does not take

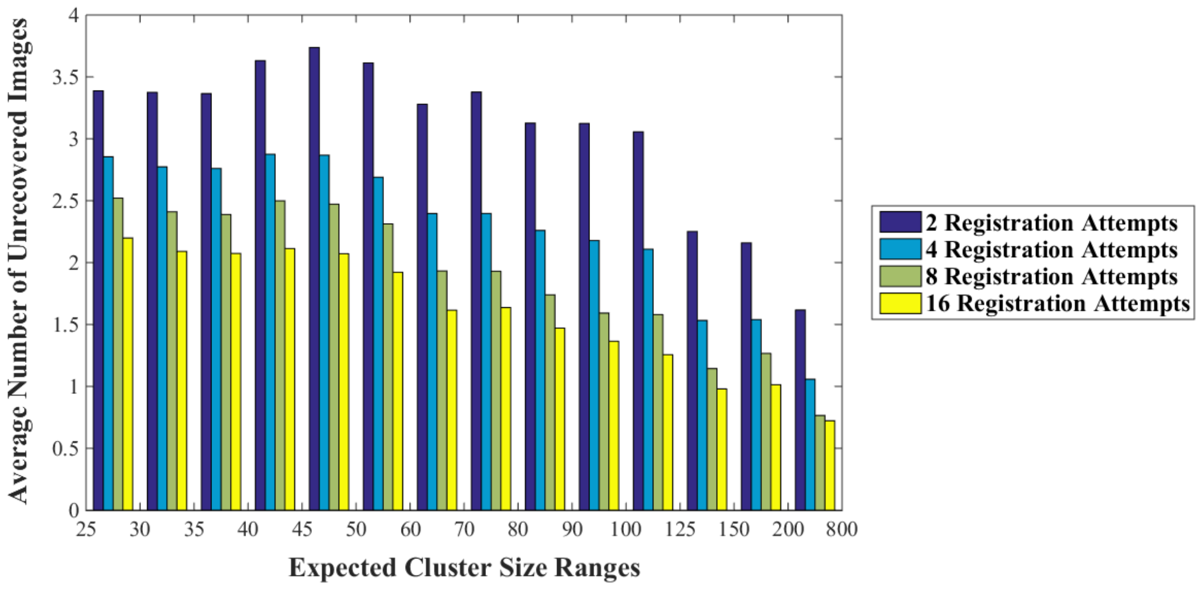

Figure 3.17: Graph showing the average number of unrecovered images per cluster when varying the number of registration attempts per image. Here, clusters of similar sizes are grouped together, and the size ranges of the expected clusters are shown along the x-axis.

into account the effects of image retrieval and geometric verification. As a failure in either image retrieval or geometric verification could result in an image going unregistered, the new minimum size we can expect to recover is the following:

$$p_r \, p_v \, M(s - \epsilon). \tag{3.13}$$

To attempt to validate and visualize this behavior, we extracted a set of clusters from the dataset formed in Section 3.3.4. Here, we used the definition of a cluster from Section 3.3.6, and selected all images that were directly connected to at least 24 others according to the computed approximate ground-truth connectivity. In this manner, we selected clusters of size ≥ 25, of which we found 70,853. Note that these clusters have overlap in their sets of images, but for analysis, we analyzed each of these clusters independently as they each are a valid hypothesis for an image cluster that could be formed. For the following analysis, we grouped these clusters together based on their sizes (with a roughly equal number of clusters per range), so that we compute an average number of unrecovered images for a particular range of cluster sizes.

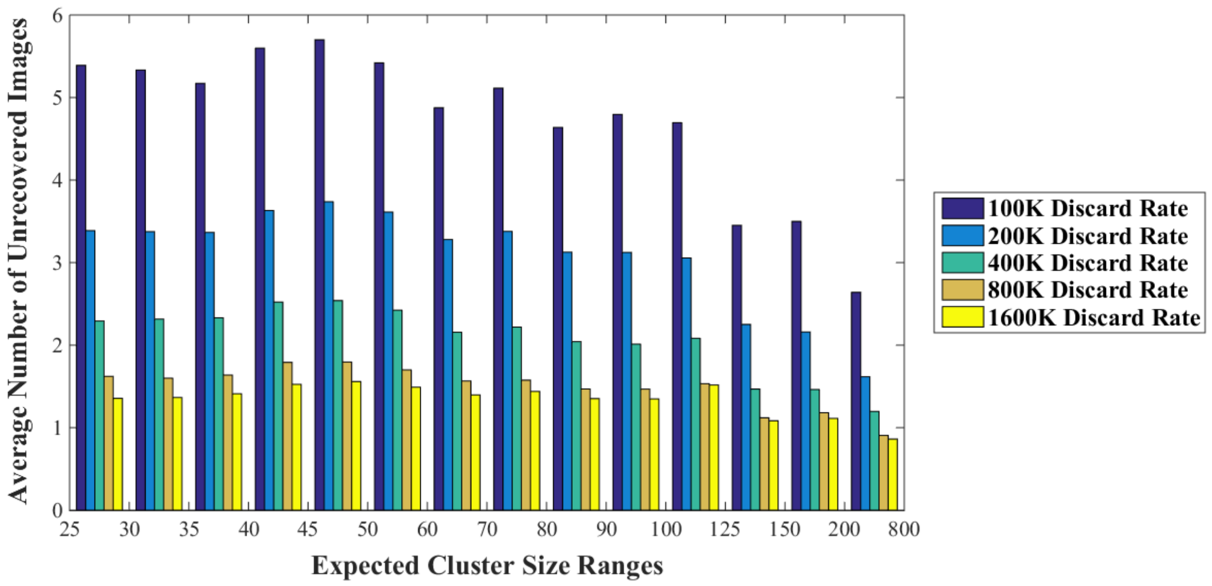

Figure 3.18: Graph showing the average number of unrecovered images per cluster when varying the discard rate of the streaming system. Here, clusters of similar sizes are grouped together, and the size ranges of the expected clusters are shown along the x-axis.

Given this set of expected clusters, we then ran our streaming system using different parameter settings, and analyzed the number of those cluster images that were recovered. As we would not expect the exact same clusters to form in our streaming system, we instead only looked to see if an expected cluster image ended up in any of the clusters output by our system.

The results when varying the number of registration attempts is shown in Figure 3.17. Here, we see that increasing the number of registration attempts decreases the average number of unrecovered images (similar to the effects of Section 3.3.5.3). By increasing the number of registration attempts per image, we are effectively increasing the value of p_r, as now the image retrieval system has a larger set of candidates in which to return the correctly-matching image. Additionally, we observe that larger clusters have a smaller amount of unrecovered images. This is also not surprising, as a larger cluster is less likely to have one of its images discarded early on in the streaming process.

To see the effects of the discard rate, refer to Figure 3.18. As before (see Section 3.3.5.4), increasing the discard rate decreases the number of unregistered images, and has a much more pronounced effect than increasing the number of registration attempts. When doubling the discard

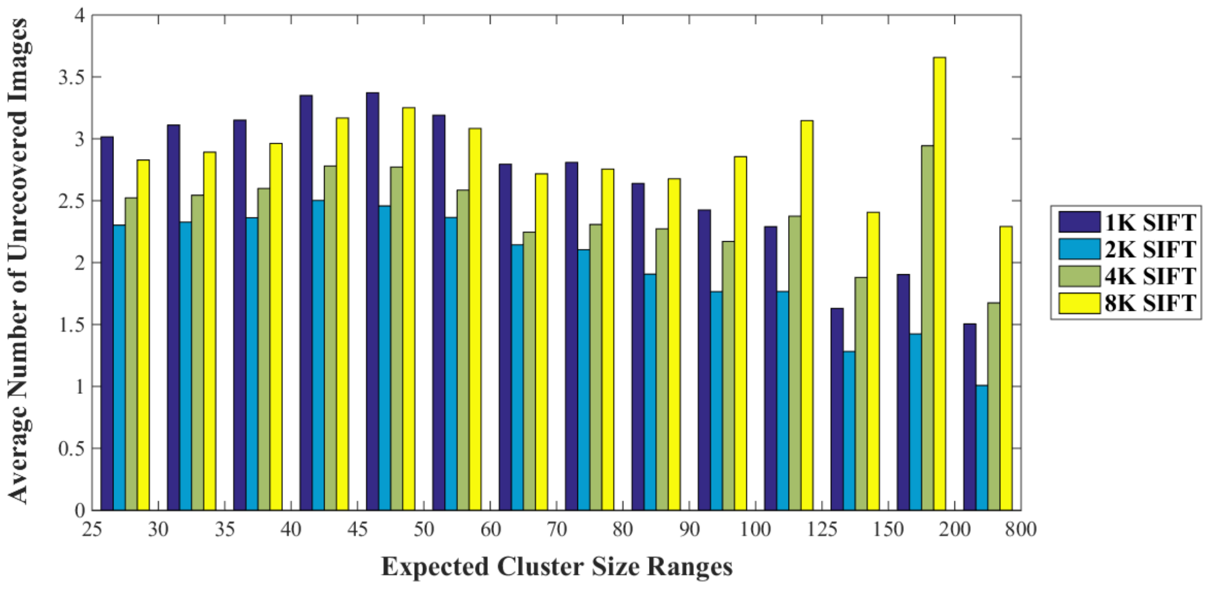

Figure 3.19: Graph showing the average number of unrecovered images per cluster when varying the number of SIFT features per image in the streaming system. Here, clusters of similar sizes are grouped together, and the size ranges of the expected clusters are shown along the x-axis.

rate, we would expect the average number of unrecovered images to decrease by half if we ignored the effects of p_r and p_v. However, by increasing the discard rate, we increase the number of clusters that are actively stored in memory (Figure 3.14), and potentially decrease p_r as now there is a larger set of images from which to perform image retrieval (increasing the chance of a false match being returned). Nevertheless, increasing the discard rate still yields significant decreases to the amount of unrecovered images, outweighing any potential harmful effects on p_r. Furthermore, changes to the discard rate appear to affect all cluster sizes relatively uniformly, in that the number of unrecovered images all decrease by common similar amount. This is the expected result, as the discard rate is not affected by the cluster size when estimating the number of recovered images (Equation (3.13)).

We also varied the number of SIFT features computed per image, and show these results in Figure 3.19. As in Section 3.3.5.5, the tests using 1K or 8K features yielded the worst results. However, here, the 2K features per image test performs noticeably better than 4K (when the results were not as clear in Section 3.3.5.5).

3.3.7 Applicability to Infinite Data Streams

All of the results presented so far are for datasets of finite size, ranging from tens of thousands to tens of millions of images. However, as this system relies on a streaming algorithm, it could be applied to run on an infinite stream of images (such as those continually uploaded to a photo-sharing website).

For instance, consider the case in an infinite stream of images where particular sets of similar images appear and disappear. This could correspond to certain popular events (which are temporal in nature), newly constructed, renovated, or demolished landmarks, or trends in photo-taking. In each of these cases, a new set of connected components and image clusters would form whenever the new set of images appears in the stream. Likewise, after a particular set of images no longer appears in the stream, its corresponding clusters will be discarded after a period of time (depending on the size of the clusters and the current discard rate setting). In this manner, the streaming system and its state would adapt to the changes in behavior of the input image stream, and would not penalize those sets of related images that appear at later positions throughout the stream.

While the streaming method as proposed would adapt to the behavior of an infinite stream of images, the memory usage model as proposed would eventually run out of memory (as there is no explicit cap on the memory usage, see Section 3.2.1.4 and Section 3.3.6.3). Therefore, the algorithm would need to be modified to enforce a maximum memory usage limit, and discard the slowest-growing clusters once this limit is met. Fortunately, there are existing streaming methods that enforce such a criteria (Manku and Motwani, 2002; Karp et al., 2003; Cormode and Muthukrishnan, 2005b), so one of their strategies could be adopted.

Finally, in the current pipeline, structure-from-motion does not begin until the streaming connected component discovery concludes. Therefore, if one desires 3D reconstructions to be built while processing the infinite data stream, SfM will need to be initiated on-the-fly (similar to (Irschara et al., 2007)). Several strategies could be adopted, but perhaps the most straightforward would be to perform a per-cluster reconstruction once a cluster has a sufficient amount of imagery. In order to encourage stable reconstructions, SfM could be initialized only once a sufficient amount of relative camera motion is achieved within the cluster, so that the 3D scene geometry can more accurately be determined. See Section 6.1.1 for more details of how this could be achieved, and the potential benefits of incorporating structure-from-motion within the streaming process.

3.4 Conclusion

We proposed a novel streaming paradigm to enable world-scale 3D modeling from unordered, crowdsourced, Internet photo collections. While the streaming processing allows for high-scalability, it posed challenges for the data association required for 3D reconstruction. We proposed novel data association concepts to overcome these challenges and reach high model completeness. In comparison to the state-of-the-art modeling from unordered photo collections, our proposed method pushes the scale of reconstructabilty by more than an order of magnitude while achieving highly complete models.

CHAPTER 4: AMBIGUOUS STRUCTURE DISAMBIGUATION USING CONFLICTING OBSERVATIONS

4.1 Introduction

In the last decade, structure-from-motion (SfM) has been taken out of the lab and into the real world. The achieved progress is impressive and has enabled large-scale scene reconstruction from thousands of images covering different scenes around the world (Agarwal et al., 2011; Crandall et al., 2011; Frahm et al., 2010; Wu, 2013). In crowdsourced reconstructions of large-scale environments, SfM methods do not have any control over the acquisition of the images, leading to many new challenges. One major challenge that arises is the ambiguity resulting from duplicate structure, *i.e.* different structures with the same appearance. Figure 4.4 shows an example of duplicate scene structure on Big Ben, where every side of the clock tower has the same appearance. SfM methods often erroneously register these duplicate structures as a single structure, yielding incorrect 3D camera registrations (see Figure 4.1). We propose a method that can correct the misregistrations caused by the duplicate scene structure in the final SfM model (see Figure 4.1 for the corrected reconstruction of Big Ben).

To correct the misregistration caused by duplicate structure, it is important to understand the nature of the ambiguity that causes the error. The most common SfM methods operate as an incremental reconstruction, *i.e.* they start from an initial pair or triplet and subsequently extend the reconstruction one-by-one for each remaining image. However, the decision of which image to add next to the reconstruction is not arbitrary. This choice is typically driven by an image similarity metric used to find images that are similar to the ones already registered (Agarwal et al., 2011; Cao and Snavely, 2013; Chum et al., 2011; Frahm et al., 2010; Lou et al., 2012; Raguram et al., 2012). It is within this process that sometimes SfM algorithms select images which do not actually

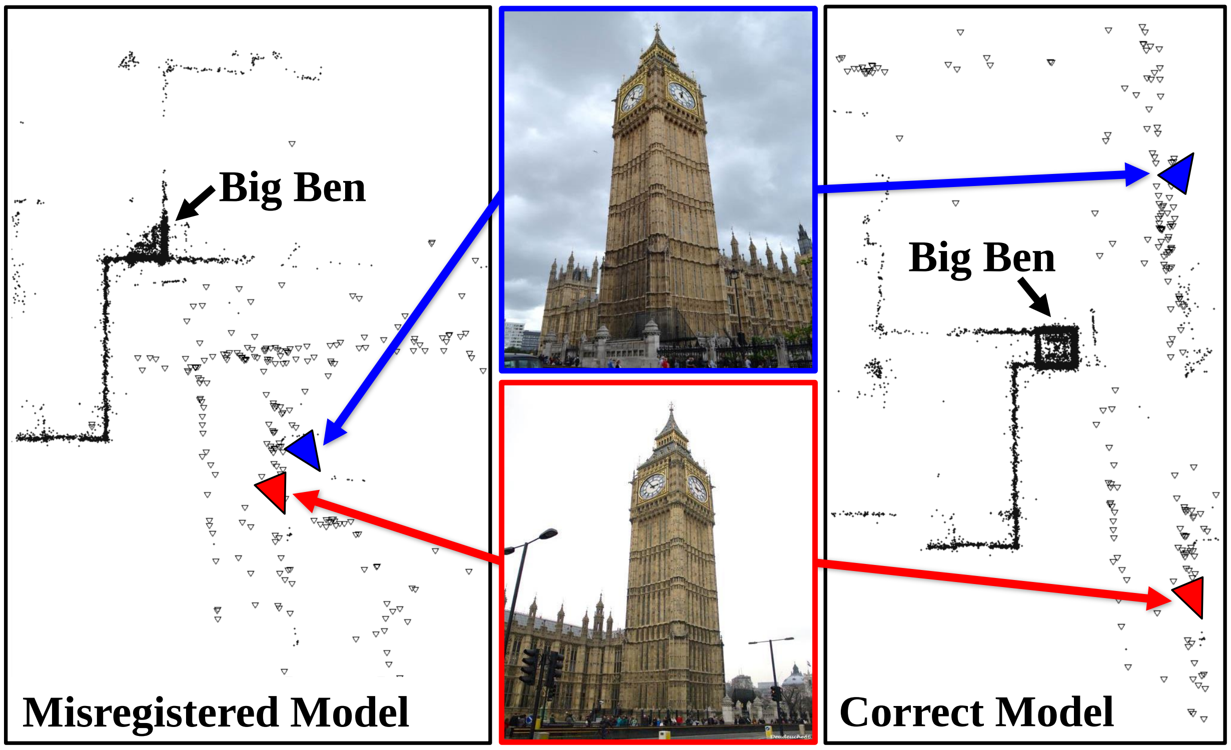

Figure 4.1: Example camera placements in a misregistered and in a correct SfM model.

overlap with the current reconstruction, but do overlap with a different instance of the duplicate structure. These images are then erroneously registered to the wrong instance of the duplicate structure. An indication for this erroneous registration is that only points on the duplicate structure register. Unfortunately, a priori knowledge of the duplicate structure is not available at registration time. Subsequent registrations extend the reconstruction further, but with the two copies of the duplicate structure combined into a single model. This erroneous reconstruction contains incorrectly placed unique structures due to the incorrect registration of the duplicate structure.

Figure 4.2 shows an incremental SfM pipeline that results in erroneous geometry. The images are sequentially registered and added to the reconstruction, and upon reaching the fourth image in the set (which is taken from a different location than the first three), it registers, but only to the facades of the tower. This registration is incorrect, as the camera should have been rotated 90° around the tower. Now, when registering the remaining cameras (which should also be rotated 90°) they will correctly register to the fourth image and start to triangulate 3D structure. However,

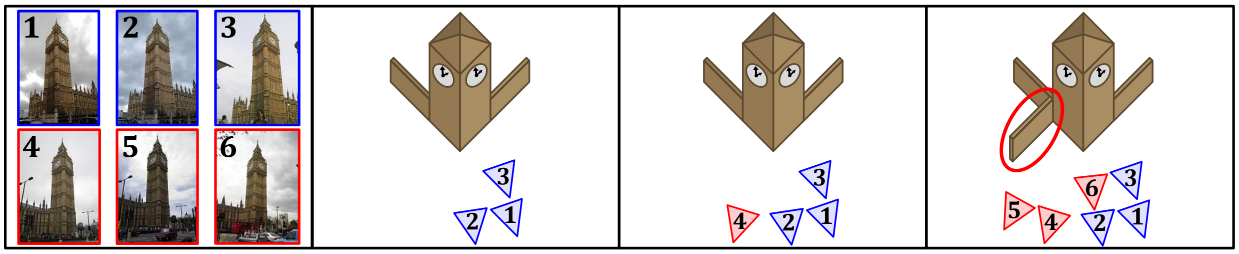

Figure 4.2: Illustration of how duplicate structure causes incorrect reconstructions. Left to right: 1) Input images ordered for reconstruction, 2) Reconstruction after first three images, 3) Fourth camera registers, but only to duplicate structure on Big Ben facades, 4) Remaining images register, and an erroneous structure is created (circled in red).

because of the fourth camera's mislocation, the new structure (and camera poses) will be incorrectly placed within the scene.

Given the difficulties of detecting erroneous registration during reconstruction, we propose a method which can correct the errors upon completion of SfM. Our method identifies incorrectly placed unique scene structures, and from this we infer the points belonging to the duplicate structure. Once our system identifies the duplicate structure, it attempts registration of cameras and points using only the distinct unique structures to obtain a correct model.

4.2 Related Work

Duplicate structure has been of recent interest in the research community and has motivated a variety of applications (Bansal et al., 2012; Cohen et al., 2012; Köser et al., 2011; Schindler et al., 2008; Sinha et al., 2012; Torii et al., 2013; Wu et al., 2011). Generally, there are different types of duplicate scene structures, ranging from duplicate instances caused by 3D rotational symmetries, separate identical surfaces, or repetitive or mirrored structures often found on facades (a survey of symmetry is provided in (Liu et al., 2010)). Duplicate structures are prone to lead to misregistered scene reconstructions, though mirror symmetries do not typically contribute to these errors.

Symmetric and repetitive structures can generally be detected in images through techniques that detect symmetric or repetitive patterns (Cho and Lee, 2009; Cho et al., 2010; Jiang et al., 2011; Lee and Liu, 2012; Liu and Liu, 2013; Zhao and Quan, 2011; Zhao et al., 2012). Methods have leveraged these patterns for urban geolocalization (Bansal et al., 2012; Schindler et al., 2008; Torii et al., 2013) and reconstruction of a scene from only a single image (Köser et al., 2011; Sinha et al., 2012; Wu et al., 2011). Furthermore, there has been recent work on utilizing symmetry as a constraint in bundle adjustment to improve the accuracy of an SfM result (Cohen et al., 2012).

The class of duplicate structures originating from 3D rotational symmetries and different identical copies of the same surface in the scene is typically not detectable by purely image-based measures. It is this class of duplicate structures that we target for correction.

In contrast to previous works which mainly relied on the idea of missing observations (described in Section 2.2.3), our method performs the inference over all cameras in the SfM model, and allows incorrectly matched images to correctly register to different parts of the same single reconstruction (Zach et al., 2008). Furthermore, our method does not require any temporal information for the images and does correctly handle crowdsourced Internet photo collections (Roberts et al., 2011). We also make no assumption about the scenes arrangement, allowing it to split and remain as separate, independent sub-models (Jiang et al., 2012). Finally, our method leverages an automatic merging technique, and circumvents oversplitting by detecting if an SfM model is already correct (Wilson and Snavely, 2013).

In summary, missing correspondences (an intuition used by many previous methods) report on structure that was expected. This implicitly assumes the repeatability of the correspondence mechanism, which can fail because of noise, occlusion, or changes in viewpoint (Mikolajczyk and Schmid, 2005). This assumption severely limits the range and type of incorrect registrations that can be detected. Therefore, the remaining unsolved challenges for model correction include robustly handling duplicate instances without oversplitting, while at the same time being able to correctly recover one or more final models (depending on the configuration of the underlying scene). It is this challenge that our method successfully addresses.

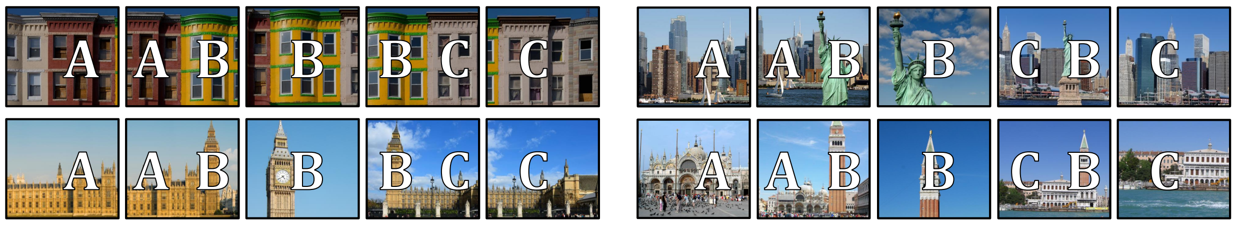

Figure 4.3: Example image sequences that depict scenes with and without duplicate structure. Each sequence in the top row shows images from the same part of a scene, and would not have reconstruction errors. However, each scene in the bottom row has the same 2D ordering of scene content (denoted by the letters A, B, and C) as the scene in the top row, but the bottom row contains a duplicate structure which would cause misregistration in the resulting reconstruction. This helps motivate the need for 3D information when performing disambiguation of duplicate structures in structure-from-motion.

4.2.1 Necessity of 3D Information for Ambiguous Structure Disambiguation

When processing a set of images and attempting to disambiguate the feature-based correspondences between them, it is necessary to leverage 3D information to be able to account for various scene configurations that may arise. For instance, consider the image sequences in Figure 4.3. Each sequence views three primary scene elements that are distributed across five images. In each case, the middle image shows a single scene element (B) that is then observed with the other two elements (A and C) in other images in the sequence. If we were to use a strategy similar to Wilson and Snavely (2013), which analyzes the local clustering coefficient of the point visibility, scene element B would be identified as an ambiguous element, as it is seen with two other scene elements (A and C) which are themselves never seen together (there is not single image that observes both A and C). Furthermore, if we were to attempt to incorporate information about the 2D locations of the feature correspondences, this still results in ambiguity, as each column of sequences shares the same order and layout of its content. However, it is only the bottom row which contains the duplicate structure.

The underlying issue here is that an image is a 2D projection of a 3D scene, and that in this process, there is a loss of information (specifically, the scene's depth). When only performing analysis using 2D information, there are a myriad of scene configurations which can mimic the same 2D connectivity between images. It is only once 3D information is incorporated back into the processing that these difference configurations can be disambiguated. Therefore, while 2D

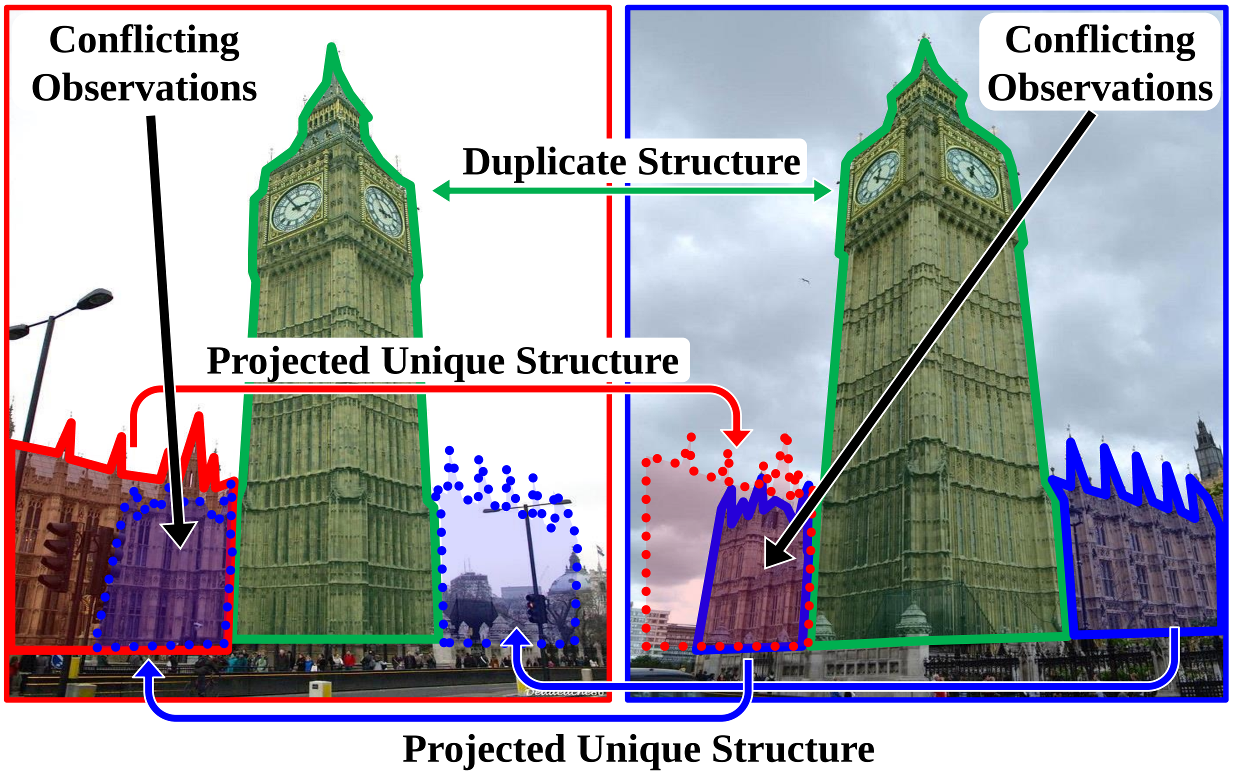

Figure 4.4: Example illustration of conflicting observations between two images.

information can provide useful information about the possibility of ambiguity, we need depth to complete the analysis.

4.3 Algorithm

We propose using *conflicting observations* to identify the incorrect registrations caused by duplicate structure as a more powerful alternative to using missing correspondences. Conflicting observations are 3D points that when projected, using the corresponding 3D reconstruction information, into and across pairwise registered images, conflict in their spatial location, *i.e.* there are observations of alternative structures in the same spatial location in the image plane. For instance, consider two separate views of duplicate structure (like the facades of Big Ben, shown in Figure 4.4). Each image contains observations of the duplicate 3D points, but the observations of the disjoint secondary structures in the scene are unique. The unique structure in the first image, when projected into the second, overlaps with the second image's unique structure. It is this unique structure that

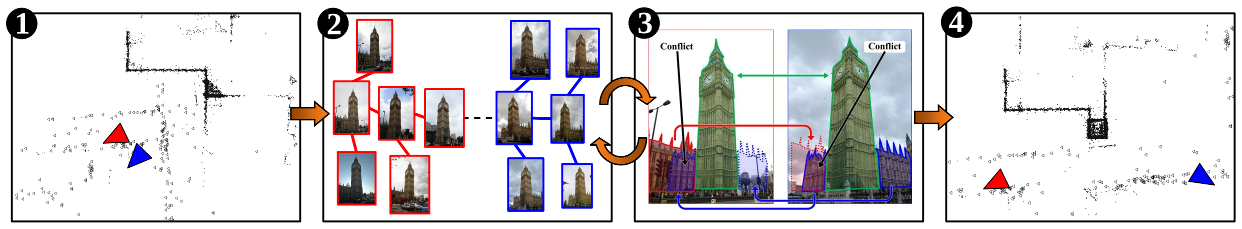

Figure 4.5: Method overview: 1. Input SfM model, 2. Candidate camera graph splits, 3. Conflicting observations, and 4. Model merging.

we analyze for conflict, as it is separate from the duplicate object and provides distinct information about the layout of the scene.

4.3.1 Overview

Given the difficulty of detecting the duplicate structures during the initial registration, our method is a post-processing step to SfM, *i.e.* the input to our system is the output of a sparse 3D reconstruction pipeline. The registered cameras and the 3D points define a *camera graph* where nodes correspond to the cameras, and edges exist between nodes whenever the two cameras view a common set of 3D points. Our method uses a recursive processing procedure whose goal is to determine if there are any errors in the current reconstruction (Step 2 and 3 in the method outline shown in Figure 4.5). During this procedure, Step 2 proposes candidate camera graph splits, dividing the cameras in the current camera graph into two subgraphs, which are then evaluated for conflict in Step 3.

For Step 3, each 3D point of the model is assigned to one of three classes: points seen by cameras in both subgraphs (potentially duplicate structure), points seen by only the cameras in the first subgraph (unique structure for this subgraph), and points seen only by the cameras of the second subgraph (unique structure in the second subgraph). Then, the unique structures are used to test for conflict between the two subgraphs by counting the number of conflicting observations between camera pairs where both cameras observe common points but the cameras originate from different subgraphs. The number of conflicting observations for such a camera pair provides a conflict score for the camera graph split. If there is considerable conflict between the subgraphs, the camera graph is permanently split and the two subgraphs are independently recursively evaluated for further conflict. If a camera graph has no considerable conflict it is accepted as valid.

After identifying the separate subgraphs of the model, our method (Step 4) attempts to merge the subgraphs to recover the correct model. It proposes candidate alignments between the split 3D models (subgraphs), and then evaluates the conflict for the merge (leveraging conflicting observations). If an alignment with sufficiently low conflict is found, then the two subgraphs are combined; otherwise, they are output as independent components of the reconstruction.

To review, we propose a conflict measure to detect overlap between structures that should be spatially unique. Applying this measure both to candidate camera graph splits and merges, we can successfully correct a misregistered SfM model.

4.3.2 Step 1: Input Reconstruction

As our method is a post-processing step for SfM, we require as input typical outputs of such a system (Agarwal et al., 2011; Frahm et al., 2010; Snavely et al., 2006; Wu, 2013). In our results we used (Wu, 2013) and (Snavely et al., 2006). Our method assumes the availability of known 3D camera poses (positions and orientations), the original images (for Step 3 of our method), and the locations of the 3D points and their visibility with respect to each camera. To perform sub-model merging (Step 4), the original feature inliers are required, *i.e.* which features were verified geometric inliers between a pair of images. In the case that some of the above input information is missing, (*e.g.* SfM without correspondences (Dellaert et al., 2000)) it can always be computed from the images and camera poses.

4.3.3 Step 2: Candidate Camera Graph Splits

We wish to generate candidate camera graph splits, where each split divides the cameras into two distinct subgraphs. These two subgraphs will then be passed to Step 3, which will evaluate the conflict between them.

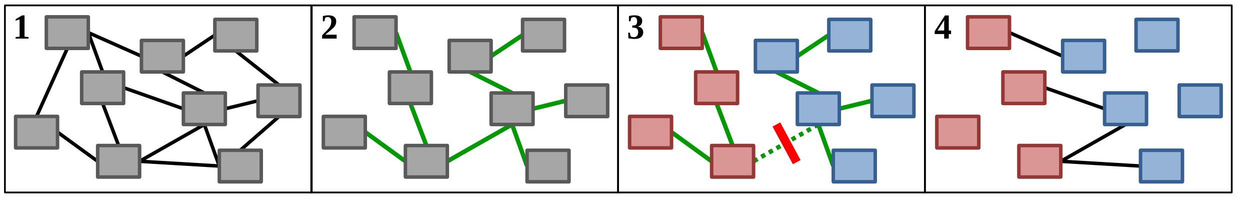

Figure 4.6: Illustration of the camera graph operations used in our method. 1. Full camera graph, 2. Minimum spanning tree, 3. Candidate minimum spanning tree split defining two camera groups, 4. Camera pairs from the full camera graph that were assigned to different groups.

Naïve potential splits can be proposed by enumerating all possible camera groupings. Given that the camera graph contains m cameras and is potentially densely connected, there would be at most $2^{m-1} - 1$ possible ways to assign the cameras to two different groups where each is a connected component (subgraph). This is exponential in m and computationally prohibitive for most large-scale models.

4.3.3.1 Minimum Spanning Tree

To reduce the number of candidate splits, we propose to leverage a minimum spanning tree (MST) representation similar to the one constructed in (Jiang et al., 2012), and illustrated in Figure 4.6, Step 2. Jiang et al. (2012) assigned to each edge a weight that was inversely proportional to the number of 3D point observations shared between the two cameras. We adopt a similar idea, but reformulate the edge cost to account for the fact that if duplicate structure is present, many of the images will have a large number of common points. Accordingly, this raw number of points should not be overemphasized, hence our edge cost leverages the following ratio where e_{ij} is the edge cost between cameras i and j, and O_i, O_j are the sets of 3D points that are visible in each camera:

$$e_{ij} = 1 - \frac{|O_i \cap O_j|}{|O_i \cup O_j|} \tag{4.1}$$

Conceptually, an MST formed using this edge cost tends to link cameras together that share similar content, and in contrast to (Jiang et al., 2012), avoids biasing the results for cameras with relatively few or many point observations.

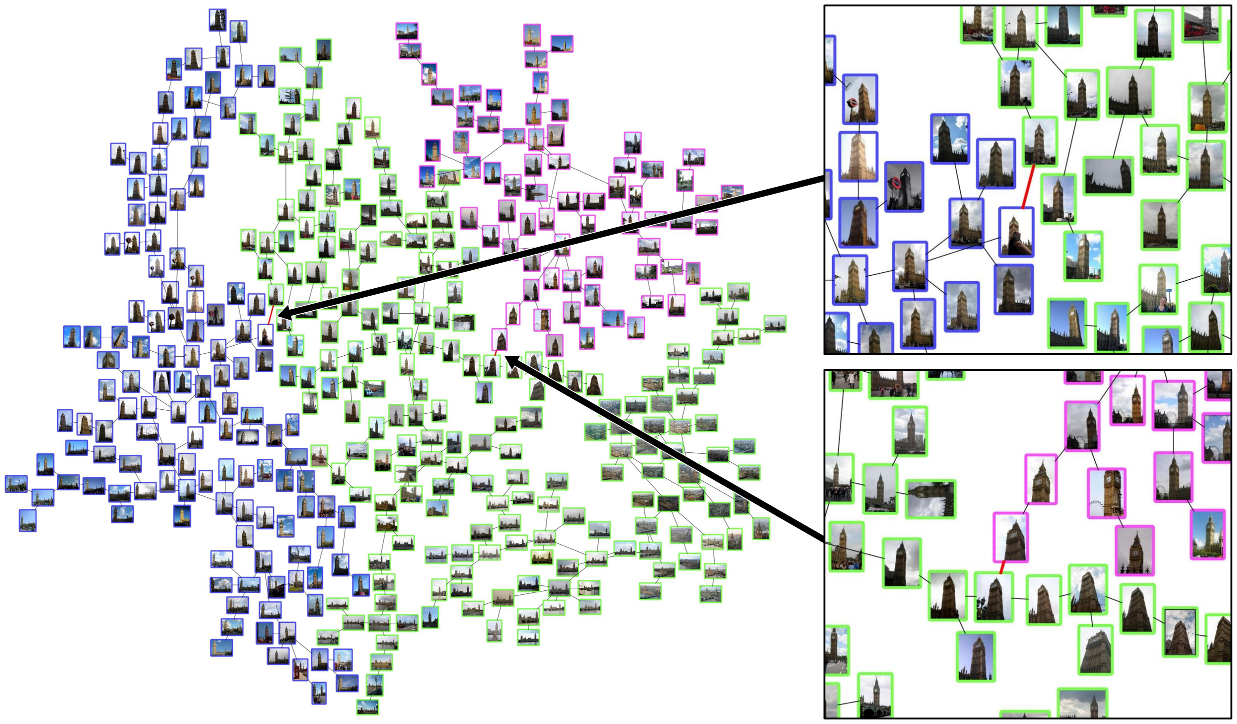

Figure 4.7: Example minimum spanning tree for images from the Big Ben dataset. The image borders are colored based on the ground-truth sub-models to which they belong (front, side, and back facades of Big Ben), and two spanning tree edges are highlighted in red to show the places where the tree should be cut to divide the reconstruction into separate, correct sub-models.

If two cameras see the same set of 3D points then e_{ij} will be zero. Accordingly, two views seeing the same instance of the duplicate structure and the corresponding surrounding unique structure will have a low edge cost. We denote these edges as *desired edges*. Conversely, two cameras, which see two different instances of the duplicate structure but do not see a common unique structure, will have a significantly higher e_{ij} value and are denoted as *confusing edges*.

Intuitively, the MST prefers utilizing desired edges (low edge cost) to connect cameras seeing the same instance of the duplicate structure and will only retain confusing edges (high edge cost), when necessary, to connect cameras seeing different instances of the duplicate structure. Accordingly, the MST will group cameras together that see the same instance of the duplicate structure and the confusing edges will bridge between these groups (see Figure 4.7 for an illustration of this behavior). This grouping behavior is a result of the use of the ratio in Equation (4.1), in that images viewing the same 3D point will tend to connected neighbors in the MST. As most camera will observed the duplicate structure, and thus all have those points in common, the connectivity of the graph is determined by the remaining content in the scene, the unique structures. Therefore, the minimum spanning tree effectively groups the images based on their content (see Figure 4.7).

An alternative way to perform the grouping would have been to leverage the original camera graph (or some graph structured formed using a subset of its edges), but weigh each edge using Equation (4.1). Then, cuts of the graph could be proposed using min-cuts, normalized min-cuts (Shi and Malik, 2000), or various clustering schemes (Johnson, 1967; Zelnik-Manor and Perona, 2004). However, an issue with this approach is that it still becomes costly to enumerate all possible splits. While the edge cost of Equation (4.1) is useful in grouping images based on their content, it is not the case that edges with the highest cost are those edges that are confusing and need to be cut. Therefore, we still need to enumerate all possible splits of the graph, and the minimum spanning tree provides the minimum number of edges needed to achieve this goal.

With this formation of the minimum spanning tree, we can now limit our camera graph splits to those that are defined by the MST, as removing a confusing edge creates two separate subgraphs defining a candidate split. Defining the search space of candidate model splits in terms of an MST representation reduces the number of potential candidate splits from one that is exponential in the number of cameras to $m - 1$, the number of edges in the MST. Refer to Figure 4.6, Steps 1-3 for an illustration of a split in the MST which results in the formation of two subgraphs.

4.3.4 Step 3: Conflicting Observations

After leveraging the MST, the next step evaluates the conflict between the two subgraphs produced by each split of the graph to obtain a reduced set of splits.

4.3.4.1 Common and Unique Structure

First, our approach classifies each 3D point into one of three categories as defined below:

$$\begin{aligned}\mathbb{D} &= \{P_k : \quad (\exists O_{ik} \in O) \wedge \quad (\exists O_{jk} \in O)\} \\ \mathbb{U}_1 &= \{P_k : \quad (\exists O_{ik} \in O) \wedge \neg(\exists O_{jk} \in O)\} \\ \mathbb{U}_2 &= \{P_k : \neg(\exists O_{ik} \in O) \wedge \quad (\exists O_{jk} \in O)\}\end{aligned} \tag{4.2}$$

where $\{P\}$ is the set of 3D points, P_k is the k-th point in $\{P\}$, O represents the visibility of points in the cameras ($O_{ik} \in O$ if P_k is visible in camera i, otherwise it is false), and i, j referring to camera i in the first set of cameras and the j-th camera in the second set of cameras. The common points (the candidate duplicate structure) between the two camera groups are denoted by $\mathbb{D}$, with $\mathbb{U}_1, \mathbb{U}_2$ denoting the unique points to each subgraph.

To improve our robustness to noisy scene geometry we enforce a minimum number of observations for each 3D point, where i is any arbitrary camera:

$$P = \big\{P_k : |\{i : O_{ik} \in O\}| \geq \rho\big\} \tag{4.3}$$

By setting $\rho = 3$ (as we did in all of our experiments), we maintain only those 3D points that are more likely to be stable and properly triangulated.

4.3.4.2 Split Camera Pairs

To evaluate the conflict of a candidate split, we analyze the image pairs from the full camera graph that had originally matched but are now split due to the images being assigned to different subgraphs. For an example of such pairs, refer to Figure 4.6, Step 4. We require a minimum number γ of 3D points that the cameras of a pair must observe in common in order to avoid images that are weakly connected as they do not represent a reliable measure of conflict.

Next, we project the unique points observed in each image to the other image in the pair, and test for conflicting observations. To mitigate the effects of occlusion or large viewpoint change, only cameras observing the same points from a similar surface incidence angle are considered. The difference in surface incidence angle β between the cameras i and j with their centers (T_i, T_j) is:

$$\beta = \texttt{arccos}\left(\texttt{dot}\Big(\frac{T_i - \bar{p}}{||T_i - \bar{p}||}, \frac{T_j - \bar{p}}{||T_j - \bar{p}||}\Big)\right) \tag{4.4}$$

where $\bar{p}$ is the centroid of their common 3D points $\bar{p}$, defined as:

$$\bar{p} = \texttt{mean}\Big(\big\{P_k : (\exists O_{ik} \in O) \wedge (\exists O_{jk} \in O)\big\}\Big) \tag{4.5}$$

where also by construction, $\{P_k\} \subseteq \mathbb{D}$. Given the limitations in matching over large viewpoint changes, our method disregards camera pairs with a difference in surface incidence angle β greater than a predefined threshold θ. The threshold θ is chosen according to the robustness of the SfM system's features with respect to viewpoint changes (for example, around 20° for SIFT features in our experiments).

Splitting at each of the $m - 1$ MST edges leads to a cubic complexity of the evaluated splits given that there is potentially a quadratic number of pairs of cameras (for a fully connected graph) for a given split. To boost efficiency, instead of evaluating all split camera pairs, we propose to inspect only those pairs with the smallest surface incidence angles β, which still allows us to detect the conflict. Specifically, we can inspect the s smallest pairs, giving quadratic overall complexity when s is a function of m, or a fixed number of smallest pairs (*e.g.* $s = 100$) to achieve a linear overall complexity. In our experiments, we have found both strategies to be valid, and thus opt for the linear time approach.

As opposed to using an MST to determine the locations to evaluate conflict, one could imagine that we could instead use a clustering or graph-cutting approach on the full camera graph. We avoided these as they would necessitate computing conflict (t from the next step) between all (or a very large fraction) of the camera pairs which could quickly become computationally prohibitive for larger camera graphs.

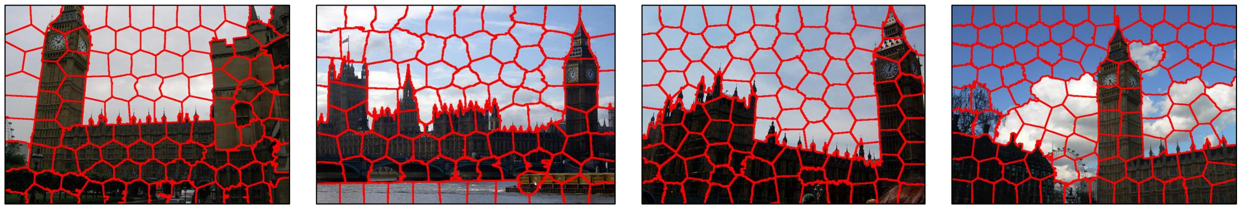

Figure 4.8: Example SLICO (Achanta et al., 2012) superpixel segmentations.

4.3.4.3 Conflict Measure

Our conflict measure leverages the unique points in the scene by projecting them into the other image of the camera pair, and expecting that they should not overlap with the unique points of that image. If there is substantial overlap, then we have reason to believe that there is an error in the current reconstruction (refer to Figure 4.4 for an example). How does one measure overlap in two projected sparse point sets? Ideally, spatially nearby (in the image) unique points on the same structural surface should conflict, whereas nearby points on separate surfaces that lie at different depths should not conflict.

To establish the surface association of the sparse points, we leverage SLICO superpixels (Achanta et al., 2012), whose only parameter is the desired number of superpixels. SLICO will automatically adapt to the texture in the image in order to maintain regular-sized superpixels (examples of which are shown in Figure 4.8). To guard against arbitrary superpixel divisions along the same structural surface, we perform multiple (eight in our experiments) different segmentations of each image by providing mirrored and rotated versions of an image to SLICO. This proved to generate a different segmentation for each (there are only eight different combinations of 90° rotations and mirror operations). With these segmentations, we now define two points to be nearby if their projections lie within the same superpixel in any of the candidate segmentations.

By leveraging the current potential duplicate ($\mathbb{D}$) and unique point sets ($\mathbb{U}_1$, $\mathbb{U}_2$) for every single camera pair, we can evaluate the subsets of these points that are currently visible in the camera pair to identify the conflicting observations.

While we focus on conflicting unique points, the locations of the common points ($\mathbb{D}$) also provide useful information. Both (Roberts et al., 2011) and (Zach et al., 2008) emphasized the usefulness of considering the spatial location of these observations. For instance, the presence of a matched point between two images would down-weigh the contribution of any nearby missing correspondences. Utilizing this concept, we obtain reduced sets (U_1, U_2) by ignoring unique points (from $\mathbb{U}_1$, $\mathbb{U}_2$) that occupy the same superpixel as a common point (from $\mathbb{D}$) in any of the segmentations.

For a given pair of images, we define the conflict t between them to be the minimum number of points from U_1 or U_2 that conflict in both images. If $\texttt{proj}(U_1)$ is the projection of the points U_1 into the second image, and $\texttt{proj}(U_2)$ is the projection into the first, then the conflict t is defined as:

$$N = \texttt{near}\big(U_1, \texttt{proj}(U_2)\big) \cap \texttt{near}\big(U_2, \texttt{proj}(U_1)\big) \tag{4.6}$$

$$t = \texttt{min}\Big(\big|\{u_1 : u_1 \in U_1 \wedge u_1 \in N\}\big|,\ \big|\{u_2 : u_2 \in U_2 \wedge u_2 \in N\}\big|\Big) \tag{4.7}$$

where $\texttt{near}()$ returns the points that are nearby as defined by the superpixel segmentations. To provide further intuition, consider the case where one unique point from U_1 conflicts with many from U_2. This single point from U_1 could be an extraneous structure, and should not count as significant conflict even though it conflicts with many points from U_2. Therefore, we leverage the minimum of the two set sizes, as this enforces a stronger indication of the presence of conflict.

Given the conflict t for a single split camera pair, the conflict over the split camera graph is the average of the conflicts from all split camera pairs between the two subgraphs (Step 4 in Figure 4.6). This average is then independently computed for each split in the MST. If the MST split with the highest conflict is above a predefined threshold τ, we remove the corresponding edge from the MST to generate two separate subgraphs. Each of these subgraphs is then processed by reapplying Steps 2 through 4 with the exception of not recomputing the MST. This is recursively repeated until we are left with a set of subgraphs that are free from conflicting observations, *i.e.* their conflict is below our threshold τ.

4.3.5 Step 4: Model Merging

Once we have a set of camera subgraphs that are free from significant conflict, we now seek to merge them together and recover a correct reconstruction (if one is possible, as the images may come from entirely separate scenes).

The key concept that we leverage here is the idea of *disconnected inliers*. Disconnected inliers are pairs of 3D points whose 2D features had been identified as inliers during the two-view geometric verification of an image pair in the SfM processing. However, due to the duplicate structure in the scene (or potentially other factors, such as feature mismatches) the inlier ended up being triangulated as two separate 3D points. Therefore, to recover candidate merges, we estimate 3D similarities that would align and reconnect the disconnected inlier points.

To estimate the similarity between the split subgraphs (and their associated disconnected inliers), we leverage a RANSAC technique, once again enforcing that a candidate solution should be made up of at least γ points in order to be considered further. Note that when generating candidate similarities, we ignore any common points that are shared between two or more subgraphs (a union of the final $\mathbb{D}$ sets from each of the split subgraphs, which we denote $\mathbb{D}_{\text{Final}}$). These points are the final duplicate structure, and as such, are not reliable for merging as they define the duplicate structure within the scene. However, once a candidate similarity has been proposed, we recompute the similarity inlier set using all disconnected inliers (even including duplicate points).

For each candidate solution, we transform the camera poses using the similarity S and update the unique 3D point structure of the subgraph. The points shared between subgraphs (the duplicate structure) are duplicated and transformed using S to correctly represent the duplicate scene structure. Then, the conflict between the two merged subgraphs is computed. In order to compute this conflict, inliers to S are identified as common structure $\mathbb{D}$. Furthermore, we load any existing 2D inliers for an image pair (from two-view geometric verification) and mark superpixels containing the 2D inlier locations as common structure. We do the latter to recover correspondences that would otherwise not have existed because the SfM algorithm ended up placing the cameras at separate locations within the scene (and choosing not to incorporate the relative pose initially computed between the images).

If the conflict is less than τ, the merge is considered correct. Otherwise, we ignore the points that were inliers to S, and attempt to estimate a different similarity. This continues until either a correct merge is found, or no solution can be computed with γ or more inliers. By repeating this process between all split camera groups, we merge all subgraphs that have valid overlapping geometry and recover a more complete and correct representation of the scene.

Now that we have correctly identified the duplicate structure within the scene ($\mathbb{D}_{\text{Final}}$), this information can be used to allow additional images to be registered to the reconstruction. For instance, when registering a new image, the image should not be allowed to register only to points contained within $\mathbb{D}_{\text{Final}}$, but should instead incorporate unique points not in $\mathbb{D}_{\text{Final}}$. In this manner, new images will not be incorrectly registered to the duplicate structure. Furthermore, this process could be embedded into an incremental SfM pipeline, so that disambiguation would occur at certain intervals to detect and mark as confusing any duplicate structure that is found. This would successfully addresses the source of the problem (the behavior of incremental SfM) as described in Section 4.1.

4.4 Results

In order to evaluate our method, we applied it to a wide variety of datasets (see Table 4.1 for a detailed overview of the datasets that led to misregistered models, not including those in Figure 4.9 that were already correct). First, we evaluated our method on datasets from previous papers (Jiang et al., 2012; Roberts et al., 2011; Wilson and Snavely, 2013), using their qualitative evaluation metric for the correct camera and model arrangement. Figures 4.10 and 4.11 (models 3, 4, 10–12) illustrate the output of our method on these existing benchmark datasets. Upon close inspection, we perform equally well or better than previous methods on their datasets as we split their models correctly (avoiding oversplits) and merge the ones that are mergeable (datasets 10–12). For instance, in dataset 4, we avoid oversplitting the front from the back of Notre Dame, as in

	Dataset Name	# Cams	# Points	Time	SfM Method
1	Big Ben (using iconics)	13,590	167,375	20.5 m	(Wu, 2013)
2	Berliner Dom	1,618	245,079	9.8 h	(Wu, 2013)
3	Sacre Coeur (Wilson and Snavely, 2013)	1,112	378,882	4.4 h	(Snavely et al., 2006)
4	Notre Dame (Wilson and Snavely, 2013) (iconics)	885	176,099	1.8 h	(Snavely et al., 2006)
5	Alexander Nevsky Cathedral	448	92,948	16.6 m	(Wu, 2013)
6	Arc de Triomphe	434	93,452	16.3 m	(Wu, 2013)
7	Radcliffe Camera	282	71,107	31.9 m	(Wu, 2013)
8	Church on Spilled Blood	277	76,582	1.4 h	(Wu, 2013)
9	Brandenburg Gate	175	23,933	3.0 m	(Wu, 2013)
10	Indoor (Jiang et al., 2012)	152	69,632	3.1 m	(Wu, 2013)
11	Cereal (Roberts et al., 2011)	25	12,194	36 s	(Wu, 2013)
12	Street (Roberts et al., 2011)	19	7,607	39 s	(Wu, 2013)

Table 4.1: Statistics showing the number of cameras and points in the dataset, the time required for our method (seconds, minutes, or hours), and the structure-from-motion software used to generate the initial reconstruction.

(Wilson and Snavely, 2013), though our method did output a small nighttime model, as there were day and nighttime versions of local image features that corresponded to the same geometry, and thus generated conflict. We did leverage iconic image selection (Frahm et al., 2010) for this dataset, and for dataset 3, we used the set of images that viewed Sacre Coeur in the covering subgraph from (Wilson and Snavely, 2013). In addition, we also ran our method on the Seville Cathedral and Louvre datasets from (Wilson and Snavely, 2013). For the Seville Cathedral, our method split the model into three main components, whereas (Wilson and Snavely, 2013) had oversplit into four. For the Louvre, we had to set $\tau = 2.0$, and were able to split it into two main sub-models. As a note, (Wilson and Snavely, 2013) split the Louvre into three sub-models, the difference being that their method split two components that were correctly oriented with respect to each other but reconstructed at different scales.

To validate that our method only alters misregistered models, we tested it on reconstructions that were already correct (eight of which are shown in Figure 4.9). In these cases, our method correctly identified them as having negligible conflict and did not attempt further processing.

Beyond benchmark comparisons, we evaluated our approach on seven novel datasets downloaded from Flickr (Figures 4.10 and 4.11, models 1, 2, 5–9). These datasets contain several

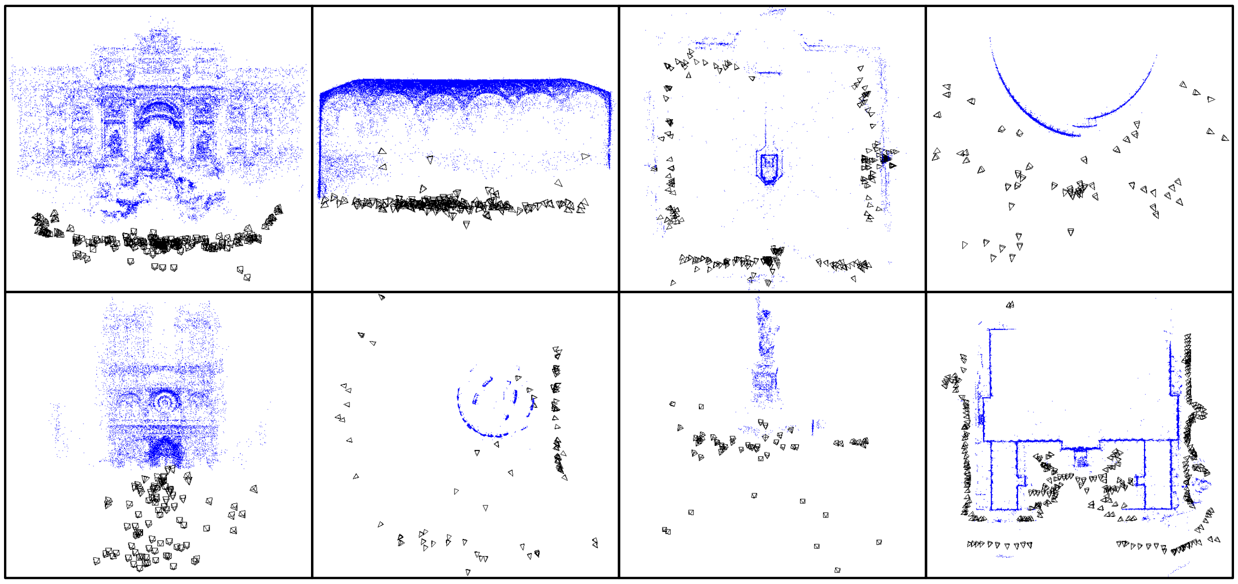

Figure 4.9: Example error-free reconstructions correctly identified by our method. From left to right, top to bottom: Trevi Fountain, Sistine Chapel Ceiling, Harmandir Sahib, Colosseum, Notre Dame Facade, Stonehenge, Statue of Liberty, and CAB (Cohen et al., 2012).

duplicate structures, and are common examples of the types of ambiguities found in urban scenes. They also represent the originally targeted (and previously unsolved) challenge of robustly disambiguating duplicate structure without making a priori assumptions on the number of correct final models to output. For datasets 1, 2, 5, 6, our method correctly split and merged the reconstructions into a single large model. It even handled the difficult challenge of Big Ben (dataset 1) where there were three split subgraphs in the reconstruction. For dataset 6, our method did output a small nighttime model. The remaining three novel datasets (7–9) successfully split and then remained as separate models, as we manually verified that there were insufficient overlapping views to support a merge. The primary reason for this lack of overlapping views is the layout of the scene itself, where photographers are limited in the number of accessible vantage points from which a desirable photo can be taken.

For further comparison, we ran the code from (Wilson and Snavely, 2013) on our novel datasets. For Big Ben, (Wilson and Snavely, 2013) split it into only two sub-models, failing to distinguish the front and back of the tower. The Berliner Dom split into five models, two of which failed to split the front of the building from the side. Alexander Nevsky Cathedral failed to split at all, but the Arc de Triomphe was correctly split into two components. Radcliffe Camera was oversplit into three models. The Church on Spilled Blood was split into two correct models, but the third smallest camera group was discarded as it was not included in the covering subgraph. Additionally, the Brandenburg Gate remained as one model, but all cameras from the back side of the structure had been discarded. Note that in the generation of these results, we extracted 2D tracks (the required input to (Wilson and Snavely, 2013)) from already triangulated 3D points. This should be a benefit to the system, as they are already cleaner than the tracks typically used as input in (Wilson and Snavely, 2013).

While our method exercises approximately linear computational complexity, for large datasets the corresponding overhead can still be reduced by leveraging the idea of iconic view selection from Frahm et al. (2010). Please note this reduction is not required but provides computational savings. For the Big Ben dataset (13,590 images) we extracted 402 iconic images in approximately linear time. Then, we split and merged a reconstruction built from only iconic images and registered the remaining cluster images to the reconstruction by attaching them to their iconic image, along with other nearby images of the same camera subgraph. By only registering to images within the same subgraph, we ignore the effect of multiple instances of duplicate structure in the scene and only register to the instance viewed in the current subgraph. Leveraging the iconic images yields the desired corrected 3D model while boosting efficiency due to the significantly reduced number of images considered in the splitting and merging.

While our method performed well on the datasets that we tested, the key assumption enabling our method is the existence of unique structure. If, for instance, the set of images or resulting reconstruction consists only of duplicate structure, our method cannot identify that the images may have come from different instances of the duplicate structure. However, this is rarely the case for real-world datasets, thus making our approach a viable option for general use.

For all experiments (except where previously noted) the same set of parameters ($\gamma = 8$, $\theta = 20°$, 100 superpixels per image, $s = 100$, and $\tau = 7.0$) was used, underlining the robustness of our method. Execution times are from our MATLAB implementation on a 3.3 GHz Xeon processor with 48 GB of RAM.

4.5 Conclusion

We have presented a novel post-processing method to detect and resolve reconstruction errors caused by duplicate structure (a common occurrence in urban environments). Our method is based on the strong and informative measure of conflicting observations. Our data-driven recursive formulation allows us to not only split an incorrect reconstruction, but to merge it back together (if possible) to recover an error-free result without making assumptions on the final number or configuration of distinct scene elements. In this regard, our experiments confirm that we outperform existing state-of-the-art methods.

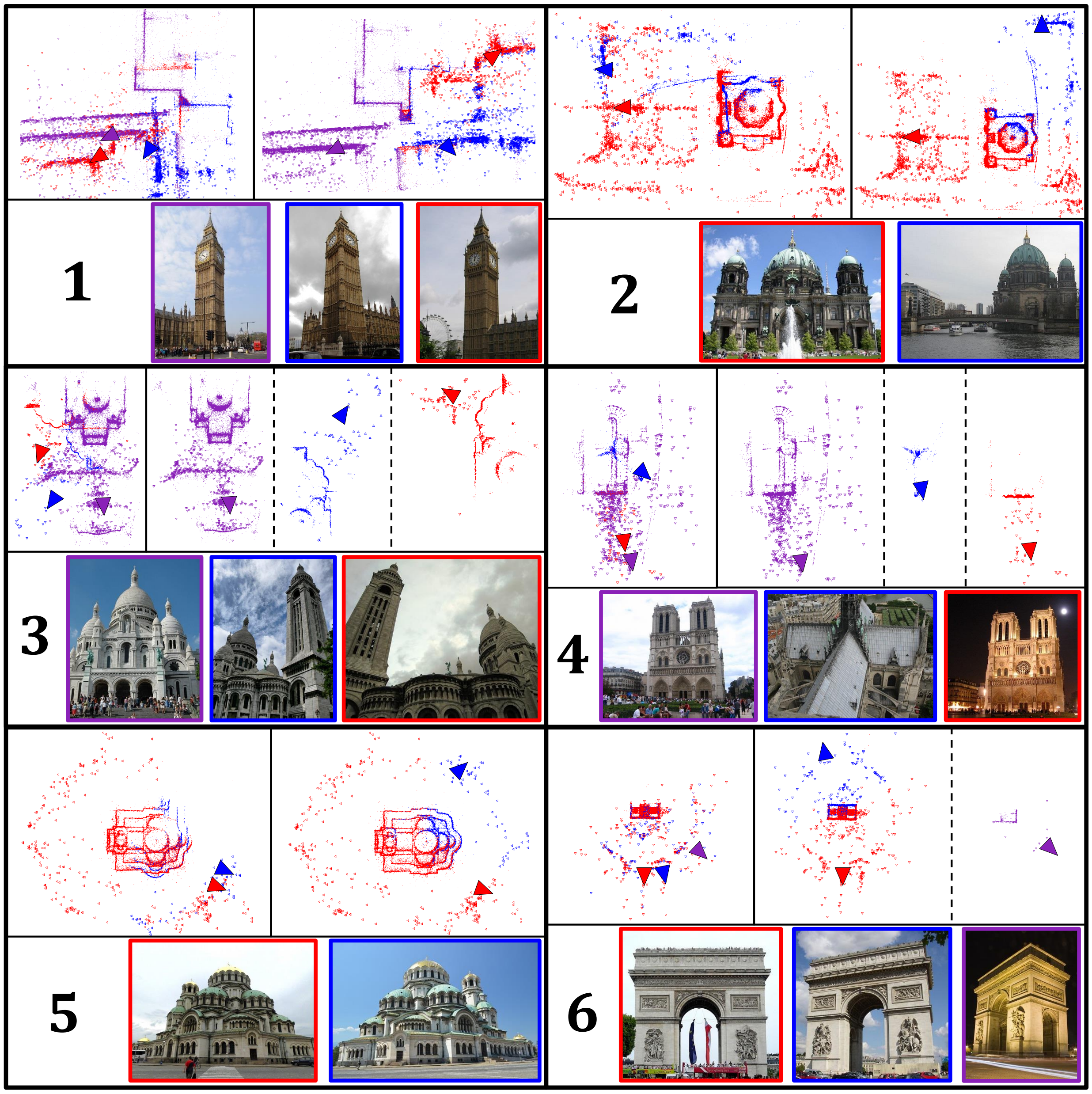

Figure 4.10: Example results from our system. Within each dataset cell: top-left is the original reconstruction, top-right is the final merged or split result (split results are separated by a vertical dashed line), and the bottom shows example images from the different split camera subgraphs. Dataset ordering (1-6) corresponds to Table 4.1.

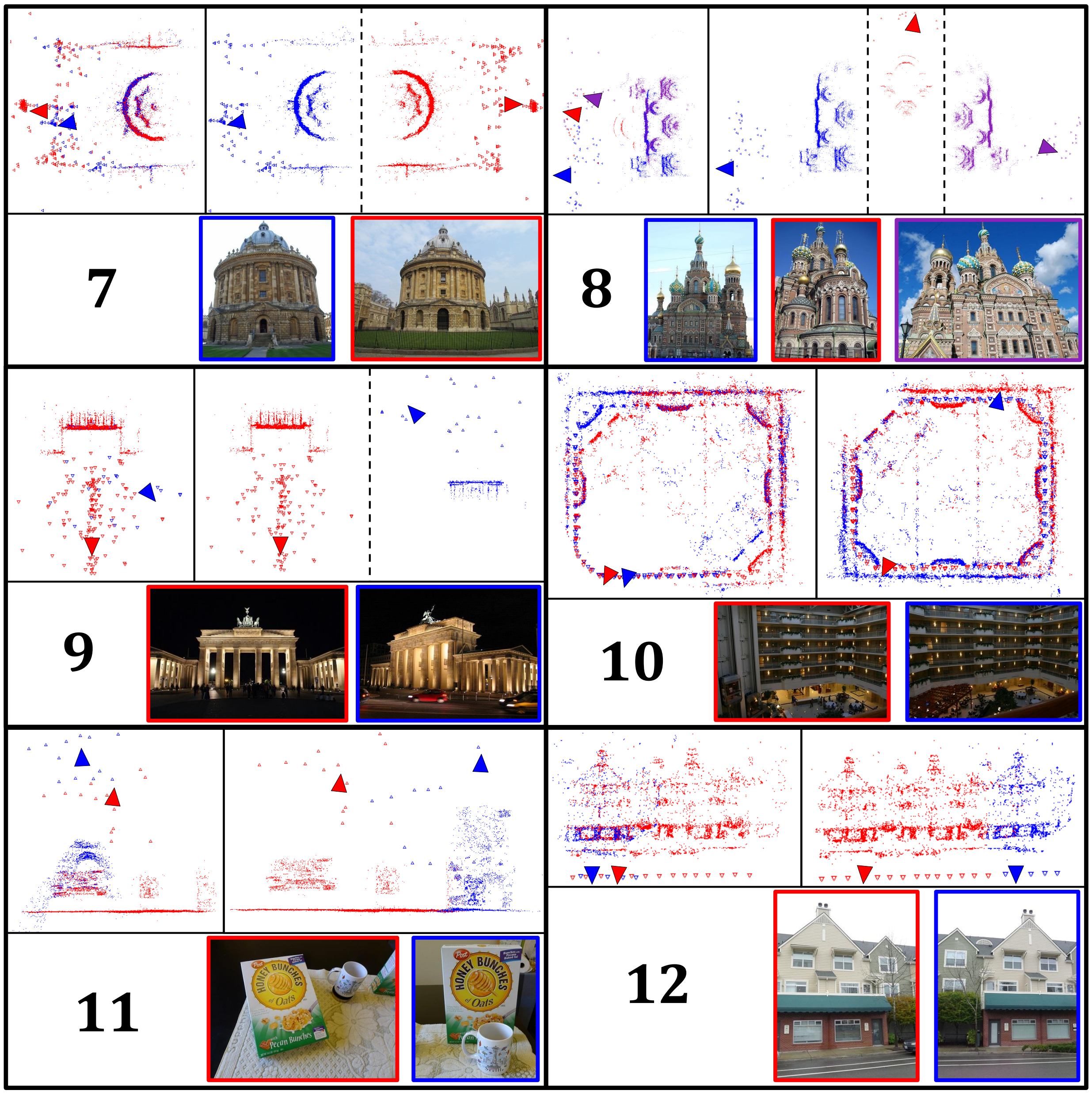

Figure 4.11: Example results from our system. Within each dataset cell: top-left is the original reconstruction, top-right is the final merged or split result (split results are separated by a vertical dashed line), and the bottom shows example images from the different split camera subgraphs. Dataset ordering (7-12) corresponds to Table 4.1.

CHAPTER 5: EFFICIENT AMBIGUOUS STRUCTURE DISAMBIGUATION USING THE LOCAL CLUSTERING COEFFICIENT

5.1 Introduction

As in Chapter 4, we seek to disambiguate and correct for duplicate scene structure as a post-process method to structure-from-motion. Our method leverages the fact that indistinguishable points incorrectly link non-unique or symmetric scene parts. The identification and segregation of these points enables partitioning an existing 3D model into disjoint structures. Model partitioning is achieved through the analysis and manipulation of the linkage relationships between the set of indistinguishable points and the rest of the model. Once a valid partition is achieved, linkage relationships among distinguishable points belonging to different partitions are analyzed to identify possible reconciliation among now disjoint sub-models.

The two most similar works to this method are those by Wilson and Snavely (2013) and the one proposed in Chapter 4. In the work by Wilson and Snavely (2013), their method leverages the *bipartite local clustering coefficient* (*blcc*) to determine those 3D points that lead to an erroneous reconstruction. Our method uses a similar intuition, but adds further levels of robustness to the analysis. Additionally, their method assumes that the final number of split components is known beforehand, though, in contrast, our method does not make any such assumption. In comparison to Chapter 4, this method has a similar inspiration, but makes practical improvements to achieve greater processing efficiency.

5.2 Reconstruction Correction Method

The main abstraction used to characterize linkage relations within our model is the co-occurrence of 3D points across images. Linkage relationships are controlled through the analysis

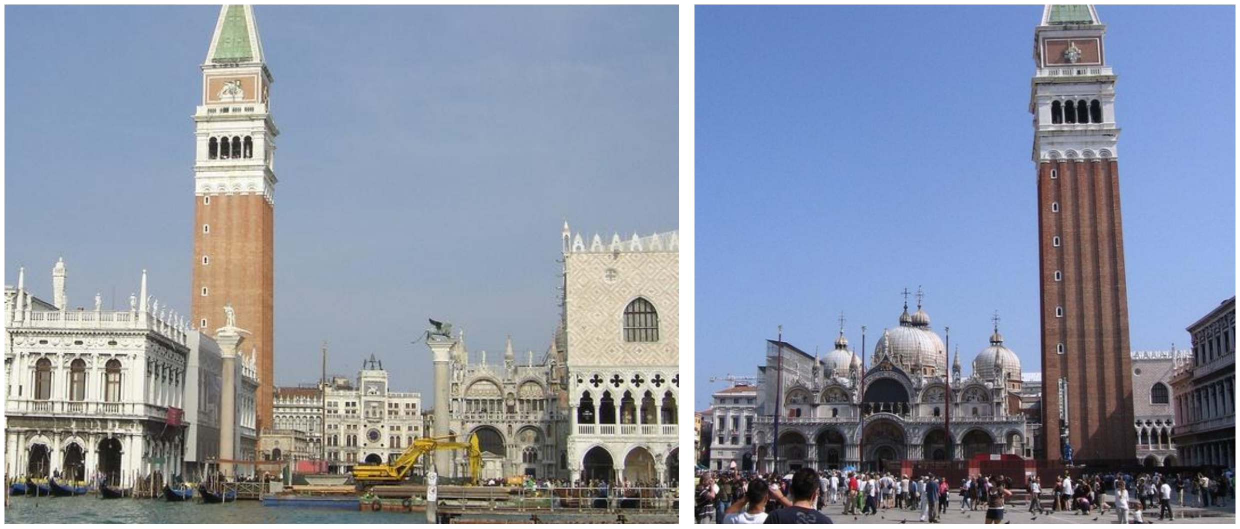

Figure 5.1: Example of two images that would have a high number of inlier matches even though they are from orthogonal views.

of two dual model representations: the *Camera Connectivity Graph* (CCG) and the *3D Point Co-occurrence Graph* (PCOG). We use these structures, along with the estimated SfM geometry, to implement data driven *split* and *merge* mechanisms aimed at identifying and mitigating erroneous 3D structure estimates. Figure 5.2 depicts an overview of our approach. For model splitting, the local connectivity in the PCOG is used as a steering measure for the sequential elimination of 3D points and the consequential dual modifications to the CCG. Splitting is achieved when the CCG is partitioned into separate connected components. For sub-model merging, we utilize geometric reasoning on the set of distinguishable points to perform sub-model to sub-model rigid registration.

To illustrate these concepts, consider Figure 5.1 depicting two images of Piazza San Marco. On the left image we observe the Maricana National Library in the lower left corner. We will refer to the features in this region as set $\mathcal{A}$, while the features on the tower's side will be referred to as set $\mathcal{B}$. Conversely, for the right image depicting San Marco Basilica in the lower left corner, we will denote this feature set as $\mathcal{C}$, while the features on the (orthogonal) tower's side will be referred to as set $\mathcal{B}'$. For our considered scenario, $\mathcal{B}$ and $\mathcal{B}'$ will have fused during SfM through feature correspondence into a single indistinguishable 3D structure $\mathbb{B}(\mathcal{B} \bigcup \mathcal{B}')$. Feature sets $\mathcal{A}$ and $\mathcal{C}$ will be mutually exclusive (*i.e.* no co-occurrence), as they will not appear jointly in our ground-based

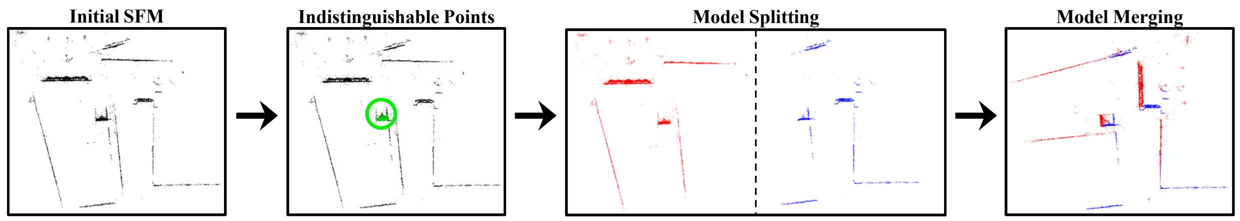

Figure 5.2: Graphical overview of the steps in our pipeline. The steps are 1) input original incorrect reconstruction, 2) identify indistinguishable points, 3) split original model into sub-models, and 4) merge sub-models together to form a correct reconstruction.

image capture, and generate (through additional similar images) independent structures $\mathbb{A}(\mathcal{A})$ and $\mathbb{C}(\mathcal{C})$. Each of these 3D point sets in isolation will approximate a clique within the PCOG (*i.e.* high local connectivity). Given that $\mathbb{B}(\mathcal{B} \bigcup \mathcal{B}')$ will be co-occurrent with both $\mathbb{A}(\mathcal{A})$ and $\mathbb{C}(\mathcal{C})$, which are mutually exclusive, the local neighborhoods of each of the sets in the PCOG are given by:

$$\begin{aligned} N(\mathbb{A}) &= (\mathbb{A} \bigcup \mathbb{B}) \\ N(\mathbb{C}) &= (\mathbb{C} \bigcup \mathbb{B}) \\ N(\mathbb{B}) &= (\mathbb{A} \bigcup \mathbb{B} \bigcup \mathbb{C}) \end{aligned} \tag{5.1}$$

where we obviate the feature dependency from the notation. Accordingly, the neighborhood $N(\mathbb{B})$ will have relatively low local connectivity compared to $N(\mathbb{A})$ and $N(\mathbb{C})$, indicating its likely denomination as indistinguishable scene structure. Sequential pruning (*i.e.* discarding) of the points in $N(\mathbb{B})$ from the PCOG will cause modifications to the edge structure of the CCG and eventually lead to the desired graph partitioning. Namely, as inlier feature matches (determined through pairwise geometric verification) are invalidated, the support for the camera motion estimates is systematically eroded. Once the CCG has been partitioned into disjoint sub-graphs, say $\mathcal{G}_A$ and $\mathcal{G}_C$, the focus turns to any inlier matches (resulting from pairwise geometric verification) that correspond to 3D points that are observed in both $\mathcal{G}_A$ and $\mathcal{G}_C$. The existence of such points offers the potential of providing a 3D registration between $\mathcal{G}_A$ and $\mathcal{G}_C$ through robust estimation procedures.

5.2.1 Initial SfM Reconstruction

We take as input the standard computed output of a generic SfM pipeline: the camera poses, focal lengths, 3D point locations, a list of the 3D points observed in each image, and the original two-view geometric verification inlier information. To generate the reconstructions we used VisualSFM (Wu, 2013). However, with indistinguishable structure, it, as well as other SfM approaches, falls victim to the ambiguity and can generate incorrect final models.

5.2.2 Identify Indistinguishable Points

The next step is to identify those indistinguishable 3D points that are most suspect for causing the corruption.

5.2.2.1 Co-occurrence Matrix

We seek to find those 3D points that incorrectly connect separate parts of a model. To enable this identification, we construct an $n \times n$ point co-occurrence matrix C (where n is the number of 3D points). This co-occurrence matrix stores boolean values indicating whether or not two 3D points were observed in the same image. A co-occurrence element C_{ij} is:

$$C_{ij} = \begin{cases} \text{true}, & \exists k \text{ such that } O_{ik}, O_{jk} \in O \\ \text{false}, & \text{otherwise} \end{cases} \tag{5.2}$$

for points i, j, image k, individual observations O_{ik}, O_{jk}, and the set of all observations O (which stores a list of the 3D points that have been observed in each camera). For larger scenes, we store the co-occurrence matrix using a sparse matrix representation.

5.2.2.2 Smooth Co-occurrences

Ideally, each 3D point should correspond to its own unique visual feature, and features that lie near each other on the same surface should be detected in the same sets of images. However, due to

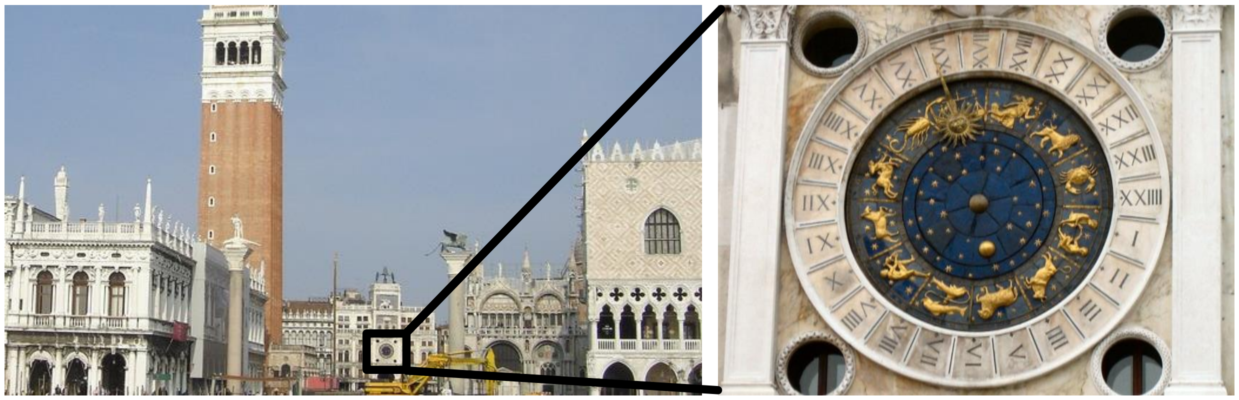

Figure 5.3: Example of two images that observe a common set of features, but at widely differing scales.

mismatches or other artifacts, these ideal conditions are rarely satisfied, leaving co-occurrences that do not represent the ideal connectivity between the 3D points. This issue was partially addressed in (Wilson and Snavely, 2013) by leveraging a covering subgraph (a minimal set of cameras that observe a large fraction of the 3D points). However, this was primarily proposed to deal with uneven scene coverage, and does not fully address the lower-level issue of feature repeatability and observation. To combat this issue, we introduce the idea of smoothing the co-occurrence matrix. Prior works (Zach et al., 2008; Roberts et al., 2011) mention the usefulness of considering nearby 2D correspondences, with the intuition that features near each other on a surface should exhibit similar observation behavior. So, a missing correspondence in the middle of found correspondences has less significance than one that is spatially distant.

The co-occurrence matrix stores information about 3D point observations, so we first determine which 3D points have projections close to each other by leveraging 2D observations of those points in each image. For each pair of observations that are near each other in an image we compute the union of their co-occurrence entries. This allows nearby observations to share their co-occurrence information, thus reducing the impact of mismatches.

To motivate our metric to determine nearby observations, we refer to Figure 5.3, where two images observe a similar set of 3D points at very different scales. In this case, a fixed viewing angle smoothing scheme does not serve our purpose, as the same radius applied to both images would result in vastly different sizes in the physical scene. We would like the smoothing radii to

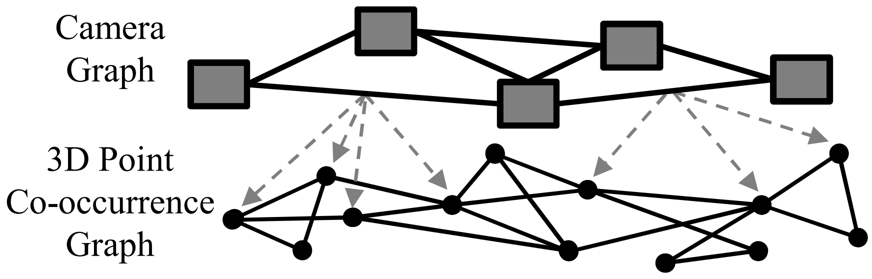

Figure 5.4: Diagram of the relationship between the camera connectivity (CCG) and point co-ocurrence (PCOG) graphs. Edges in the CCG represent shared 3D point observations between two images, whereas edges in the PCOG indicate that two points were observed together. The dashed arrows show which 3D points correspond to the inliers between two images.

have a common physical meaning, *e.g.* a larger smoothing radius must be used in the right image (close-up view). Computing a unique scale for each 3D point observation affords the effect of an adaptive smoothing radius. A feature observed from farther away will be associated with a larger overall scale, so that when observed from a closer distance, that scale will correspond to a larger smoothing radius. To this end, we leverage the available 3D information (up to scale) from the initial reconstruction (as opposed to 2D observations only). For each 3D point i ($1 \leq i \leq n$), camera j ($1 \leq j \leq m$), camera position T_j, horizontal field-of-view θ_j, and 3D point location p_i, we compute an initial scale s_{ij}:

$$s_{ij} = ||p_i - T_j|| \tan\Big(\frac{\theta_j}{2}\Big) \tag{5.3}$$

and final smoothing radius r_{ij}:

$$r_{ij} = \rho \, \frac{\max(s_{i1}, ..., s_{in})}{s_{ij}} \tag{5.4}$$

where ρ is a constant factor. Any two point observations that occur within radius r_{ij} are considered to be similar and are updated to have the union of their co-occurrences.

5.2.2.3 Co-occurrence Analysis

Given the computed point co-occurrence matrix, we want to identify the indistinguishable 3D points responsible for reconstruction inconsistencies. The intuition is that indistinguishable features will potentially incorrectly link (via co-occurrences) two disjoint parts of the model, where those disjoint parts are never viewed at the same time. Alternatively, a normal, distinguishable feature will be connected to points that are frequently seen with each other, as they are all from the same part of the scene.

By interpreting the co-occurrence matrix as an adjacency matrix defining a PCOG (where 3D points are nodes and co-occurrences are edges), we utilize graph theory to analyze precisely this property. The *local clustering coefficient* (lcc) (Watts and Strogatz, 1998) measures how close a vertex's neighbors are to being a complete (fully connected) graph and is defined as:

$$lcc = \frac{2\,(\#\ of\ edges\ between\ neighbors)}{(\#\ of\ neighbors)(\#\ of\ neighbors - 1)} \tag{5.5}$$

where a value of 1 signifies a fully connected set of neighbors, and a value close to 0 indicates reduced connectivity. Points with low lcc values are more likely to be the indistinguishable structure causing the reconstruction artifacts, while higher lcc values denote more typical behavior. This is highly similar to the $blcc$ metric in (Wilson and Snavely, 2013), though $blcc$ is designed to operate over bipartite graphs. In (Wilson and Snavely, 2013), $blcc$ was computed on the original (unsmoothed) co-occurrence matrix of their covering subgraph. In contrast, our approach operates on the full camera and point sets, and leverages a scale-aware adaptive smoothing for added robustness.

Computation of the lcc values is inherently a $O(n^3)$ operation, where n is the number of 3D points. In practice, lcc is typically not computed on a fully connected PCOG, though it is still a computational bottleneck. To mitigate this issue, we leverage a random sampling based approach (similar to (Wilson and Snavely, 2013)) to compute approximate lcc values. While not explicitly mentioned in (Wilson and Snavely, 2013), their reference implementation employs a sampling scheme to achieve high efficiency. Specifically, one that has $O(n)$ complexity with respect to the number of 3D points. Here, sampling the PCOG for a specific 3D point can be modeled as sampling a binomial distribution, therefore we compute the number of samples required to achieve a 99.7%

confidence of being within 0.01 of the actual lcc value. We verified our method using the exact and approximated lcc values, and both resulted in the same final 3D models.

5.2.2.4 PCOG and CCG Pruning

Given the lcc values for each 3D point, we now seek to remove the contribution of the indistinguishable features. We accomplish this by iteratively removing the 3D points with the lowest lcc values, and then inspecting the connectedness of the CCG, where each image is a node and edges between nodes exist as long as there are a sufficient number (τ) of shared 3D points between them. A diagram of this relationship is illustrated in Figure 5.4. By removing an increasing number of 3D points, edges in the CCG are removed (because their shared 3D points have been removed) such that, eventually, independent connected components are generated.

We strive to separate the CCG into a minimal number of error-free connected components. The global cost metric proposed in (Jiang et al., 2012) addresses the determination of the correct number of splits assuming the final reconstruction to be a single model. We aim to allow independent scenes that were incorrectly combined to be split and remain separate. Hence, we assume there is one primary ambiguous element in a corrupted model (empirically, it is a viable assumption).

To identify this primary ambiguous element, we continue to remove the most indistinguishable 3D points until the CCG splits into two main groups (sub-models of size greater than 1). Then, the points in common (the intersection) between these two groups are taken as points belonging to the ambiguous structure. The necessity of the intersection computation stems from the possibility that some of the points with low lcc values may not have actually contributed to the incorrect reconstruction. Therefore, by computing the intersection of the two groups, we obtain the set of points that actually linked the two sub-models.

We again enforce spatial smoothness, such that when counting the number of shared points between two images, we exclude any point within a radius (3ρ) of an observation of a 3D point that has been removed due to its low lcc.

5.2.2.5 Correct Reconstruction Detection

By removing 3D points until the reconstruction splits in two, even an initially correct model will be split. To avoid decimation of a correct model, we evaluate the validity of the proposed split by taking inspiration from Chapter 4 (Heinly et al., 2014a), but with a focus on efficiency.

We first generate a list of all image pairs that had 3D points in common, but were assigned to the two different sub-models. Then, we determine the set of 3D points unique to each of the two models, which are those points not observed in the opposing model (let us denote these as P_1 and P_2). By analyzing the 2D projections of P_1 and P_2 in the images from disjoint sub-models, we develop a notion of *overlapping* correspondences (similar to *conflicting observations* from Chapter 4 (Heinly et al., 2014a)). The intuition is that in a correct reconstruction, the 2D projections of the opposing model's points would not overlap (have a close spatial proximity) with an image's existing observations of the points from its own model. If the observations did overlap, that would indicate the presence of two different reconstructed structures at a similar location within the scene. Therefore, if we detect a large amount of overlap between images from disjoint models, the reconstruction is incorrect and we continue our pipeline. Otherwise, with a lack of overlap, we identify the original model as correct and terminate execution.

5.2.2.6 Measuring Overlap

To mitigate the effect of scene occlusions and increased feature mismatches for images with wide viewpoint differences, we only consider image pairs with similar viewing directions (at most 10° of difference). Also, we suppress overlapping correspondences occurring near the shared 3D points between the images, as these are more likely a result of noise or detection artifacts (also noted in (Roberts et al., 2011) and (Zach et al., 2008)). Instead of counting the raw number of overlapping correspondences (as in (Heinly et al., 2014a)), our metric computes the area of overlap, which normalizes the result against a scene's 3D point density. Here, each point projects to a circle within the image, and we compute the overlap between the respective circles. Each circle's radius, as well as the radius in which observations are ignored around shared 3D points, was chosen to be

0.1 in normalized image coordinates (when the image is inscribed in a circle of radius = 1). To compute a final value, we average the ratios of conflicting coverage for each of the image pairs, and threshold the result (treating any model with less than 1% of overlapping correspondences as correct).

Instead of using normalized image coordinates, we could instead have used multiple superpixel segmentations to determine the local neighborhood of a projected 3D point as in (Heinly et al., 2014a). However, superpixel computation is costly (around 10-15 seconds per image for eight different segmentations (Heinly et al., 2014a)). Therefore, to achieve greater efficiency, we opted for the normalized image coordinate approach described above.

5.2.3 Model Splitting

Given a partition of the CCG into two components, we seek to determine the correct number of groups in which to split the final model. The intuition is that the first split will identify the indistinguishable region of the scene, but the final model may have to be split into a larger number of sub-models depending on the characteristics of the scene (number of ambiguous objects, symmetric facades, etc).

We first expand the set of indistinguishable points by including any other 3D point found to be an inlier (according to the 2D feature matches from the SfM pipeline) to any of the initial indistinguishable 3D points. Typically, only a subset of the indistinguishable structure may have been initially identified, so by leveraging matching and spatial proximity constraints, we dilate the indistinguishable set to better improve our estimate. For the spatial constraint, we analyze 2D point observations and include into our indistinguishable set any point that occurs within a fixed pixel distance of an already identified indistinguishable point (we do two passes using the previously used radius of 3ρ).

With this expanded set of indistinguishable points, we repeat a similar process to PCOG pruning, where we remove the points from the reconstruction, and then inspect the CCG. We eliminate camera connections not sharing a minimum number of points γ, and the final set of connected components are the final camera groups for the model.

5.2.4 Model Merging

For some scenes, the set of sub-models from the previous step may in fact be the correct final solution. However, in many cases, the correct final solution is a merging of the split components, such that they correctly resemble the scene. To this end, we identify original 2D inlier feature matches corresponding to observations of different final 3D points after splitting (*disconnected inliers* from (Heinly et al., 2014a)). At some point during the matching phase, two images may have been correctly matched, but ended up in different groups due to the dominant indistinguishable structure in the scene. By identifying these disconnected inliers, we have a basis to correctly merge the models back together.

For the identification of disconnected inliers, we ignore inlier matches occurring near the final set of indistinguishable features (leveraging the spatial smoothness constraint). The final set of indistinguishable points $\mathbb{P}$ is:

$$\mathbb{P} = \bigcup_{i=1}^{g} \bigcup_{j=i+1}^{g} P_i \cap P_j \tag{5.6}$$

where g is the number of groups and P_i are the points observed by group i. With $\mathbb{P}$, we again enforce matching and spatial smoothness constraints, dilating the point set first to their inliers, and then by the spatial radius ρ.

By ignoring disconnected inliers that were members of $\mathbb{P}$, the remaining set of 3D correspondences is utilized in a similarity-estimating RANSAC technique (we seek a consistent rigid transformation between any of the final camera groups). As the scale of the SfM reconstruction is undetermined, we rely on a 3D distance inlier threshold that is determined as 1% of the 90^{th} percentile of the 3D points' distances from the model's mean 3D location. If RANSAC finds any transform with enough inliers (γ as from Section 5.2.3), we align and merge together the split models, and leave as split any unregistered models. After identifying the indistinguishable features,

Dataset Name	# Cams	# Points	Time	Time (Heinly et al., 2014a)
Piazza San Marco	3372	410592	3.0 m	5.6 m
Brandenburg Gate	50	8046	18 s	12 s
Arc de Triomphe	192	32708	1.7 m	2.7 m
Giotto's Campanile	211	52620	4.4 m	22.5 m

Table 5.1: Summary of the datasets used in our evaluation. The Time columns are the runtime's of our method and (Heinly et al., 2014a) in minutes or seconds.

we densify and augment the final model by replicating those points between all split models, as previously proposed (Jiang et al., 2011; Pauly et al., 2008).

5.3 Results

We evaluated our method on a variety of unordered photo-collections, and for all datasets our method's input was obtained from VisualSFM (Wu, 2013). We defined $\rho = 0.01$, $\tau = 10$, and $\gamma = 18$ for all experiments. Furthermore, to increase efficiency and robustness to noisy correspondences, we leveraged 3D points that had a minimum of four image observations. Table 5.1 shows statistics for our main datasets, with Figures 5.5, 5.6, and 5.7 showing illustrations of our results.

To verify the correctness of our approach, we ran our method on datasets that were already free from error (Figure 5.5). Here, the correct reconstruction detection method from Section 5.2.2 correctly identified that the first split lacked overlap, and thus was already a correct reconstruction. To further verify correctness, we ran our approach on existing benchmark datasets. The Books (Jiang et al., 2012; Roberts et al., 2011), Oats (Jiang et al., 2012; Roberts et al., 2011), and Indoor (Jiang et al., 2012) datasets (Figure 5.6) were all used and correctly solved in previous papers. Our method also correctly identifies the proper split and merge operations for each, though our approach has fewer limiting assumptions when compared to these previous works (see Section 2.2.3).

We also ran our method on four Internet photo collection datasets (Table 5.1 and Figure 5.7). For Piazza San Marco (Figure 5.7.1), we downloaded 3,372 images from Flickr, and performed GIST-based clustering to attain an iconic scene graph of 311 nodes (using a method similar to

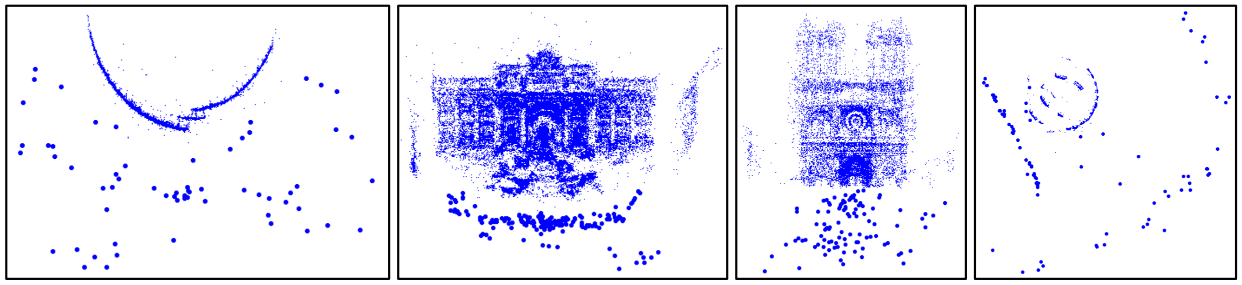

Figure 5.5: Example 3D reconstructions with no existing errors correctly identified by our pipeline (data from (Heinly et al., 2014a)). From left: Colosseum, Trevi Fountain, Notre Dame, Stonehenge.

Frahm et al. (2010)). Each cluster is geometrically verified and the representative iconic cluster centers processed by VisualSFM to generate our input data. After our method's execution, the corrected iconic 3D model is densified by registering each clustered image to its iconic image and the surrounding images from the same split camera group to form a complete and corrected 3D model. For this dataset, our method correctly identified the indistinguishable structure on the tower, and split and merged the reconstruction into a correct final model.

For the Brandenburg Gate (Figure 5.7.2), both sides of the gate had originally been confused. Our method split them into their respective sides and left them as separate models. This solution is correct as there is not enough connecting structure in the original model to allow for the two sides to be correctly merged, due to camera viewpoint distribution.

For the Arc de Triomphe (Figure 5.7.3), our method identified three main camera groups. The largest two groups (the front and back of the arch) were correctly merged, but the third, smaller group failed to merge into the final model. The primary reason for this result was that not only were the images taken from a vantage point not entirely covered by any of the other two groups (cameras predominantly looking at the underside of the arch), but the points that the third group did have in common ended up being too close to other indistinguishable points. While this is undesirable, our method still results in a majority of the images being used to create a full reconstruction of the building.

The Giotto's Campanile model (Figure 5.7.4) was split into four camera groups, two of which were merged back together. The remaining two un-merged groups had no overlap with the first two

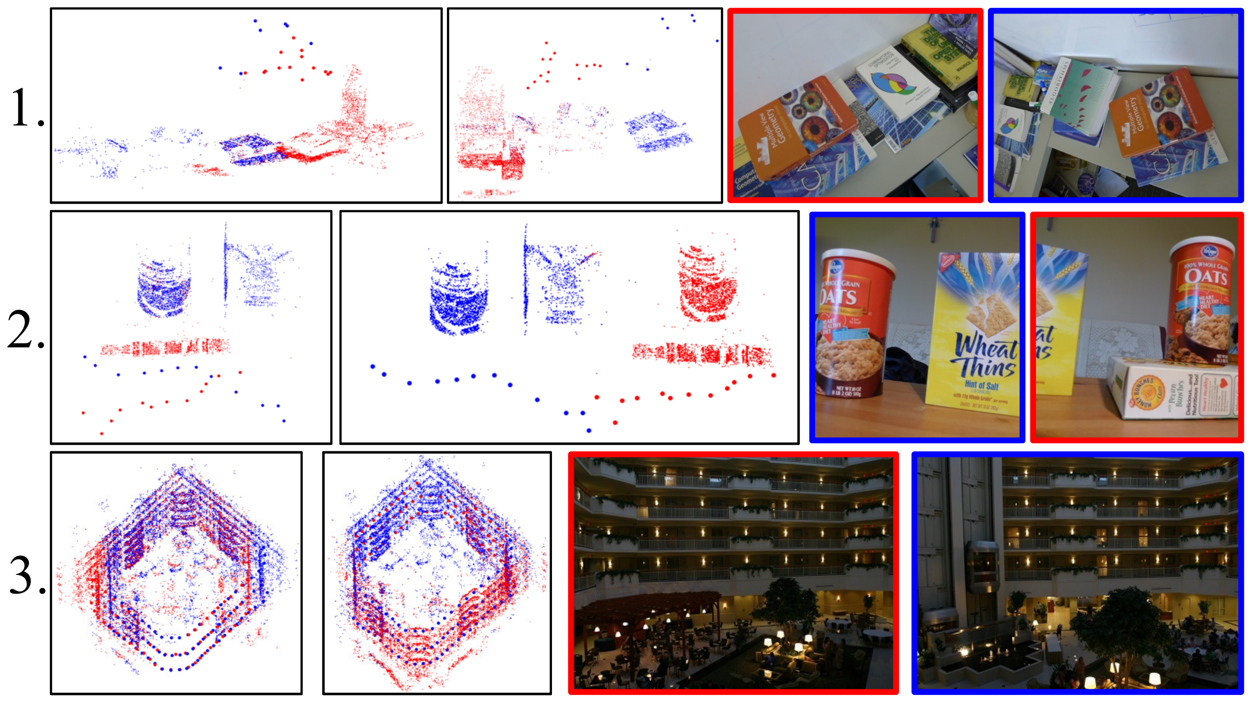

Figure 5.6: Results for the 1) Books, 2) Oats, and 3) Indoor datasets (from (Jiang et al., 2012; Roberts et al., 2011)). See Figure 5.7 for a description of what is shown.

groups, as they were images taken from the building's opposite side. While these un-merged groups observed a common structure, their vastly different perspectives prohibited disconnected inliers.

To provide further comparison to previous work, we ran the method of (Heinly et al., 2014a) on our above four datasets. Our method typically has a faster runtime (see Table 5.1), and note that superpixel computation time is not included in Table 5.1, further emphasizing our greater efficiency over (Heinly et al., 2014a). Additionally, (Heinly et al., 2014a) failed to merge several of the datasets' components (for instance, Piazza San Marco, the main components from Arc de Triomphe, and the first two components of Giotto's Campanile). While (Heinly et al., 2014a) did achieve correct splits in each of these cases, there was insufficient scene coverage for the method of (Heinly et al., 2014a) to achieve successful merges.

We also ran the method of (Wilson and Snavely, 2013) on our datasets. For Piazza San Marco, (Wilson and Snavely, 2013) outputted a correct subset of the main plaza (red in Figure 5.7), but completely discarded all cameras from the adjoining plaza (blue). For Brandenburg Gate, the method discarded all cameras in the reconstruction, resulting in an empty final model. For Arc de

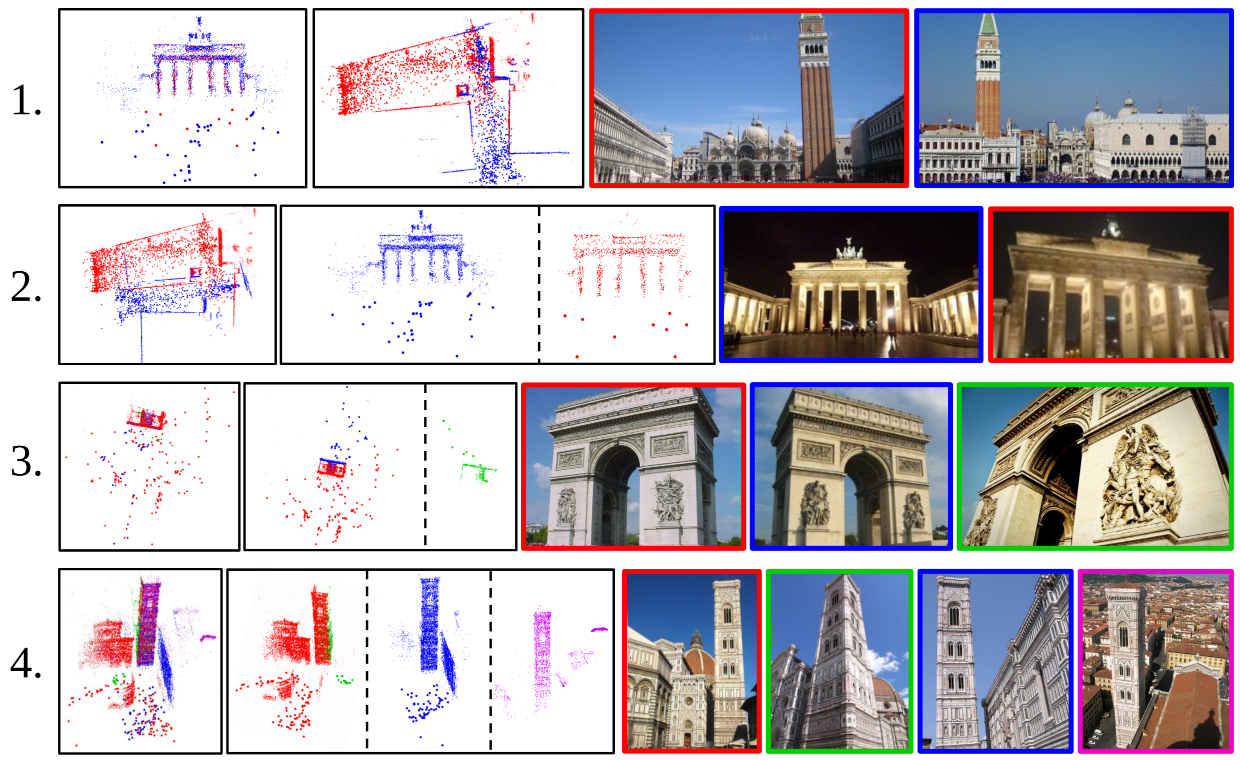

Figure 5.7: Results for datasets from Table 5.1. From left to right in each row: original model, our result, and example images colored to correspond to their sub-model. For SfM models, the smallest points are 3D structure, and larger circles are camera positions.

Triomphe, (Wilson and Snavely, 2013) correctly output a subset of the cameras facing the main facade, but discarded all cameras from the opposite side. Finally, for Giotto's Campanile, it split the original model into two components, one of which still contained errors.

For all tests our method correctly split models into independent CCG components, each generating a correct sub-model. Additionally, our MATLAB implementation is highly efficient and is a natural post-processing mechanism for SfM. Our main computational bottleneck is the worst case $O(m^4)$ sequential CCG component analysis (m is the number of cameras). In practice, however, significantly fewer than m^2 cuts are required to partition the CCG.

5.4 Conclusion

We have presented a novel method for correcting corrupted SfM reconstructions originating from non-unique, symmetric, or otherwise indistinguishable structure. Our technique leverages co-occurrence information to split the initial model into several consistent sub-models, and then is able to correctly merge them back together if permitted by the captured images. Furthermore, throughout the calculation, the set of 3D points causing the inconsistencies is identified, enabling a variety of additional applications.

CHAPTER 6: DISCUSSION

This dissertation has presented efficient methods to robustly handle the data association problem in large-scale structure-from-motion systems. In Chapter 3 we proposed a new paradigm for connected component discovery in large-scale photo collections. This method is based on a streaming framework, and we demonstrated its effectiveness on a world-scale 100 million image dataset (Shamma, 2014; Thomee et al., 2015). In Chapter 4 we proposed a state-of-the-art method to perform disambiguation in the case of duplicate scene structures. Here, our method is able to identify the errors in a reconstructed model, and correct for those errors resulting in an accurate representation of the scene. Finally, in Chapter 5, we proposed modifications to the duplicate structure disambiguation method to improve its efficiency by exploiting the co-occurrence information present in the scene's geometry.

6.1 Future Directions

In regard to the area of research to which this dissertation pertains, we propose several different avenues for future work on these topics.

6.1.1 Extensions to the Streaming Paradigm for Connected Component Discovery

One of the first useful extensions to this work would be to make a second pass through the input dataset (assuming that it is static). Here, the intention would be to recover a higher fraction of the registerable images in the dataset. To make the processing more efficient, one could experiment with the number of nearest neighbors on which each streamed image attempts to register. For example, the first pass could be modified to only match to the first nearest neighbor, so that the output is only a set of image clusters (and not connected components). Then, the second pass could use the current existing strategy of matching to two neighbors, to allow for the linking of clusters into components. Furthermore, and trivially, the second pass through the images would not need to recompute SIFT features, avoiding the bottleneck of feature computation.

Another useful strategy of the second pass would be to focus on those sets of images that already have been clustered. Specifically, when streaming through on the second pass, the inverted index of the vocabulary tree could be initialized with the iconic images of the final clusters, and any image that does not successfully register to one of these iconic images would be immediately discarded. To improve the diversity of the recovered images, and to avoid biasing toward those images that have already been recovered, we could augment the set of existing iconic images with additional images from the cluster that are sufficiently diverse. For instance, a cluster image that successfully registered, but had a large variation in its bag-of-words representation could be used as an additional, diversified representation of the cluster, to help register those images that were not sufficiently similar to the cluster's single, original iconic image.

One of the current limitations of the streaming pipeline is that structure-from-motion is not run until the streaming connected component discovery concludes. It would be useful to visualize the recovered 3D geometry as it is being discovered, and it could actually increase the performance of the system. For instance, as opposed to waiting until the end of the streaming, structure-from-motion could be run for each cluster as it forms. To promote stable and accurate results, the reconstruction could be initialized only once a sufficient initial pair of images is found (Beder and Steffen, 2006). This could naturally be incorporated as a test between the iconic image of a cluster, and each new image that is added to it. Once the reconstruction is initialized, instead of performing essential matrix estimation for each new candidate image, a more efficient 2D-3D registration technique could be employed (yielding faster runtimes).

To represent the structure-from-motion model in the vocabulary tree, each of its 3D points can be represented either by a single SIFT descriptor from an image that observes it, or by the centers computed from a mean-shift clustering of all the SIFT descriptors viewing that point (Pollefeys et al., 2010). To avoid biasing toward the already reconstructed parts of the cluster, the unreconstructed features from the iconic image could additionally be used in the cluster's bag-of-words representation (just as the unregistered features of an iconic image are used in its representation).

One negative aspect in attempting to perform structure-from-motion during the streaming process is an increased memory burden for those clusters that have yet to be reconstructed. For instance, one scheme would be to keep the feature locations and registration information in memory for all cluster images, so that when a cluster is able to reconstruct, it can use information from all of the cluster images to refine its representation. However, storing this information will greatly increase each cluster's representation in memory, which may be prohibitive. To alleviate this issue, a cluster could discard information about its cluster images as usual, and only start to retain additional information once a successful two-view reconstruction has been initialized for the cluster.

This above strategy of leveraging structure-from-motion during the streaming process could also be easily combined with a two-pass implementation. For instance, the first pass could perform the clustering as normal, and upon completion, structure-from-motion would be run. Then, on the second pass through the dataset, efficient 2D-3D registration could be used, greatly speeding up the computation in that second pass. In this scheme, by only allowing the images in the second pass to register to existing SfM models, the focus would be on growing and combining the models, and any image that does not register to one of the models would be immediately discarded.

Another extension would be to incorporate an online learning strategy that determines what type of images are registerable in the dataset. The motivation for this is that image registration is one of the largest (if not the largest) computational bottlenecks of the system. Therefore, any improvement to its efficiency will improve the overall runtime of the system. To accomplish the goals of learning what type of images are registerable, we could use a method similar to (Raguram et al., 2012), where a classifier is trained based on the current set of registerable and unregisterable images that have been processed. Once the classifier is trained to a sufficient confidence, images in the stream that have too low of a score (as determined by the classifier) can automatically be skipped or only matched to a reduced subset of their nearest neighbors (*i.e.* their first nearest neighbor) to avoid further computation.

Continuing the goal of reducing the computational burden of image registration, we could test the usefulness of using a cascade of image registration techniques. For instance, we could first test for the existence of a valid affine transform or a global camera rotation (assuming all features are at an infinite distance) before attempting to test for an essential matrix. The reasoning for attempting these methods is that they are very easy to compute from their minimal samples, avoiding the expensive operations required to solve for an essential matrix (Nistér, 2003). Additionally, they have a smaller minimal sample size, which greatly reduces the number of required RANSAC iterations to achieve the same confidence and inlier rate. While the number of inliers to one of these simplified models would be lower, the reduced sample size can yield a reduced number of iterations and computation time.

In the case when essential matrix estimation needs to be performed between two images, estimating the essential matrix from its minimal set of five points is a non-trivial operation (Nistér, 2003). Several recent works have reported success in replacing the five-point algorithm with efficient minimal solvers (Helmke et al., 2007; Rosten et al., 2010b; Botterill et al., 2011; Lui and Drummond, 2013). In these cases, methods reported average speed improvements ranging from a few percent up to two or three times faster than the standard implementation. We wrote an initial implementation of (Lui and Drummond, 2013), and it is indeed faster, but currently only for images pairs with fewer than 70% inliers. Therefore, we could continue to investigate these methods, optimize their implementation, and even propose a hybrid approach that would switch between the standard five-point method and an iterative solver once RANSAC has reasonable confidence that the inlier rate is below a particular threshold (which would be after a particular number of RANSAC iterations).

To improve the speed at which RANSAC finds a valid solution (assuming one exists), we could leverage the work by Sattler et al. (2009) in which they propose a spatial consistency filter prior to estimation in RANSAC, which they term SCRAMSAC (Sattler et al., 2009). Here, candidate feature matches are pruned when they do not have a sufficient number of nearby matches mapping to similar parts of the two images. The end result is that the overall inlier rate is typically increased, allowing RANSAC to more rapidly find the valid solution. For instance, speed improvements up to two orders of magnitude were demonstrated when employing this strategy (Sattler et al., 2009). While we would test SCRAMSAC's applicability as proposed in its paper (Sattler et al., 2009), we would also test it in conjunction with PROSAC (Chum and Matas, 2005), which leverages a better sampling scheme when some prior information is known about the quality of each of the candidate feature matches. One of the potential shortcomings we have observed with SCRAMSAC is that it can discard too many of its candidate feature matches in low-inlier scenarios. Therefore, by using the spatial consistency score leveraged in SCRAMSAC, we can sort the candidate matches, and use them in a PROSAC-style sampling. This would yield the benefits of the spatial consistency filter, while at the same time preserving the entire candidate match set. When testing the use of PROSAC, however, we would need to take care to avoid biasing the results toward degenerate configurations, as mentioned in Sattler et al. (2009).

Apart from image registration (RANSAC and essential matrix estimation), feature computation is the other large bottleneck of the streaming system even when using multiple GPUs and a GPU-enabled SIFT implementation (Wu, 2007). Therefore, by using a feature which is more easy to compute, the computational burden of this stage in the pipeline could be greatly reduced, as well as reducing the need for multiple, expensive GPUs. Several binary features have recently been proposed, some of which provide excellent performance in certain scenarios (Heinly et al., 2012). Specifically, we would investigate the use of BRIEF (Calonder et al., 2010) in conjunction with Harris corners (Harris and Stephens, 1988). Both of these are very easy to compute, though in combination, no scale or rotation invariance is provided. Our motivation here is that in large-scale crowdsourced photo collections, many people will take photos in either portrait or landscape orientation, obviating the need for rotation invariance (except to find correspondences between these two orientations). Furthermore, with such a large collection of images, there will be many different views of the same scene at different scales, which could be linked transitively when using a detector/descriptor pairing that is not scale invariant. Therefore, the streaming system could be run using this feature combination, and only once useful image clusters are found, would SIFT features (Lowe, 2004; Wu, 2007) be extracted to link clusters of widely differing scales or orientations.

If a descriptor that was not rotation invariant were to be used for image registration, essential matrix estimation could be modified to take advantage of this fact. Specifically, because the descriptors enforce a similar 2D rotation between the images, we can assume that there is an insignificant amount of relative rotation around the camera's viewing directions. Because this reduces the number of degrees of freedom in the relative pose (from five to four), a smaller minimal sample size could be used in RANSAC, yielding a faster runtime.

6.1.2 Extensions to Ambiguous Structure Disambiguation

Currently, one area of weakness in the ambiguous structure disambiguation methods is the scalability and runtime. Even though the method of Chapter 4 has linear complexity in the number of cameras, and the method of Chapter 5 has linear complexity in the number of points, the computation time can still be non-trivial for large datasets. Specifically, we would like to maintain the excellent disambiguation ability of the first method, but drastically improve its speed (as opposed to the lcc based method which sacrifices some disambiguation ability). One modification that could be made would be to leverage the edge costs in the minimum spanning tree. Currently, once the minimum spanning tree is constructed, only its structure is used and its weights are ignored. Upon inspection, the edge with the most conflicting observations is also usually the edge with the highest cost in the minimum spanning tree. Therefore, the edge cost could be used as a prior, biasing the search toward these edges.

Another way to speed up the method of Chapter 4 is through a breadth-first search of the camera pairs associated with each minimum spanning tree edge. Currently, each edge in the minimum spanning tree results in up to $s = 100$ camera pairs being evaluated for conflicting observations. A smaller number of camera pairs could initially be evaluated at each minimum spanning tree edge, and then only those that show the most conflict could be selected for further evaluation, in a recursive, breadth-first manner. This search could also be biased by the minimum spanning tree edge weights as mentioned above, to provide another prior as to which edges to focus the evaluation.

Another way to address the efficiency of the duplicate structure disambiguation is to expand upon the extension proposed in Section 3.2.3. Here, as opposed to processing the entire model at once, independent components are identified using the mean-shift algorithm over the sparse 3D geometry (projected to 2D space). While this worked for the datasets on which it was tested, a more rigorous formulation and evaluation would prove useful. For instance, as opposed to using mean-shift, images could be more explicitly grouped based on the visibility graph structure, so that cameras observing common content are the ones used for the disambiguation. Furthermore, these groupings could allow the algorithm to run in parallel, as each group could be evaluated independently for conflict, and then a global merge operation could correct for any discovered errors.

6.1.3 Additional Research Directions

We now propose two more general research topics related to structure-from-motion systems.

The first deals with many of the points brought up in Section 6.1.1, where image registration is a major bottleneck in structure-from-motion systems. Specifically, the issue when two candidate images will not register, and thus consume the maximum number of RANSAC iterations. While some of the proposed strategies will address this issue, efficient data association in large, unorganized photo collections is an open topic. Therefore, we propose to leverage feature-based (similarity of visual words, similarity of underlying feature descriptors, *etc.*), geometry-based (similarity of spatial distribution of features, recurrence of common feature clusters, *etc.*), and classifier-based (learn models on useful feature and geometry-based configurations) methods to put better priors on the ability of two images to successfully register. This would be in a similar vein of work as Li et al. (2009); Lou et al. (2012); Mills (2013); Hartmann et al. (2014); Schönberger et al. (2015a).

The second topic deals with errors in the reconstructed structure-from-motion model. While we already proposed a post-processing step to correct for errors due to duplicate or ambiguous structure, these are not the only errors that are present. For instance, popular scenes that are imaged both during daytime and nighttime hours often result in separate 3D structures, or superimposed surfaces within the same model. Additionally, temporary structures that are captured in many photos, such as scaffolding used for restoration, can cause the previously proposed duplicate structure disambiguation methods to incorrectly segment the model. Given the success of the conflicting observations measure in the disambiguation of duplicate structure, and its underpinnings in reasoning about the geometric layout of the underlying scene, we propose to investigate additional reasoning or criteria which could be added to the structure-from-motion process in order to avoid the types of issues listed above. Here, as opposed to blindly triangulating 3D points and registering cameras, additional geometric scene validity would be enforced to constrain the reconstruction (Furukawa et al., 2009; Gallup et al., 2010b; Cohen et al., 2012; Wu et al., 2012; Bao et al., 2013; Häne et al., 2013, 2014). One potential way to incorporate these types of constraints in structure-from-motion is through the simultaneous use of dense estimation techniques (such as multi-view stereo methods). The incorporation of this information could allow a reconstruction method to better verify if a newly aligned image conforms to the currently reconstructed scene, as several methods either directly report (or could be modified to report) on the confidence of the estimated depth information, or the reliable subset of images used to estimate the final depth for a particular pixel or region (Strecha et al., 2006; Goesele et al., 2007; Furukawa and Ponce, 2010; Zheng et al., 2014a).

BIBLIOGRAPHY

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(11).

Agarwal, S., Furukawa, Y., Snavely, N., Curless, B., Seitz, S. M., and Szeliski, R. (2010a). Reconstructing Rome. *IEEE Computer*, 43(6).

Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., and Szeliski, R. (2011). Building Rome in a Day. *Communications of the ACM*, 54(10).

Agarwal, S., Snavely, N., Seitz, S. M., and Szeliski, R. (2010b). Bundle Adjustment in the Large. In *European Conference on Computer Vision (ECCV)*.

Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building Rome in a Day. In *IEEE International Conference on Computer Vision (ICCV)*.

Aholt, C., Agarwal, S., and Thomas, R. (2012). A QCQP Approach to Triangulation. In *European Conference on Computer Vision (ECCV)*.

Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). FREAK: Fast Retina Keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Amazon (2015). Amazon EC2 Pricing. http://aws.amazon.com/ec2/pricing.

Arandjelović, R. and Zisserman, A. (2012). Three Things Everyone Should Know to Improve Object Retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ardeshir, S., Zamir, A. R., Torroella, A., and Shah, M. (2014). GIS-Assisted Object Detection and Geospatial Localization. In *European Conference on Computer Vision (ECCV)*.

Arthur, D. and Vassilvitskii, S. (2007). `k-means++`: The Advantages of Careful Seeding. In *ACM-SIAM Symposium on Discrete Algorithms*.

Bansal, M., Daniilidis, K., and Sawhney, H. (2012). Ultra-wide Baseline Facade Matching for Geo-Localization. In *European Conference on Computer Vision (ECCV) Workshops*.

Bao, S. Y., Chandraker, M., Lin, Y., and Savarese, S. (2013). Dense Object Reconstruction with Semantic Priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110.

Beder, C. and Steffen, R. (2006). Determining an Initial Image Pair for Fixing the Scale of a 3D Reconstruction from an Image Sequence. In *Symposium for Pattern Recognition (DAGM)*.

Botterill, T., Mills, S., and Green, R. (2011). Fast RANSAC Hypothesis Generation for Essential Matrix Estimation. In *International Conference on Digital Image Computing: Techniques and Applications*.

Brown, M., Hua, G., and Winder, S. (2010). Discriminative Learning of Local Image Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(1).

Bujnak, M., Kukelova, Z., and Pajdla, T. (2008). A General Solution to the P4P Problem for Camera with Unknown Focal Length. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Calonder, M., Lepetit, V., and Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision (ECCV)*.

Cao, S. and Snavely, N. (2013). Graph-Based Discriminative Learning for Location Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ceylan, D., Mitra, N. J., Zheng, Y., and Pauly, M. (2014). Coupled Structure-from-Motion and 3D Symmetry Detection for Urban Facades. *ACM Transactions on Graphics (TOG), Proceedings of SIGGRAPH*, 33(1).

Charikar, M., Chen, K., and Farach-Colton, M. (2002). Finding Frequent Items in Data Streams. In *International Colloquium on Automata, Languages, and Programming*.

Chatterjee, A. and Govindu, V. M. (2013). Efficient and Robust Large-Scale Rotation Averaging. In *IEEE International Conference on Computer Vision (ICCV)*.

Cho, M. and Lee, K. M. (2009). Bilateral Symmetry Detection via Symmetry-Growing. In *British Machine Vision Conference (BMVC)*.

Cho, M., Shin, Y. M., and Lee, K. M. (2010). Unsupervised Detection and Segmentation of Identical Objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chum, O. and Matas, J. (2005). Matching with PROSAC - Progressive Sample Consensus. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chum, O. and Matas, J. (2010). Large-Scale Discovery of Spatially Related Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(2).

Chum, O., Mikulík, A., Perdoch, M., and Matas, J. (2011). Total Recall II: Query Expansion Revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chum, O., Philbin, J., Sivic, J., Isard, M., and Zisserman, A. (2007). Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval. In *IEEE International Conference on Computer Vision (ICCV)*.

Cohen, A., Zach, C., Sinha, S. N., and Pollefeys, M. (2012). Discovering and Exploiting 3D Symmetries in Structure from Motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cormode, G. and Muthukrishnan, S. (2005a). An Improved Data Stream Summary: The Count-Min Sketch and its Applications. *Journal of Algorithms*, 55(1).

Cormode, G. and Muthukrishnan, S. (2005b). What's Hot and What's Not: Tracking Most Frequent Items Dynamically. *ACM Transactions on Database Systems*, 30(1).

Crandall, D., Owens, A., Snavely, N., and Huttenlocher, D. (2011). Discrete-Continuous Optimization for Large-Scale Structure from Motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Crandall, D. J., Owens, A., Snavely, N., and Huttenlocher, D. P. (2012). SfM with MRFs: Discrete-Continuous Optimization for Large-Scale Structure from Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(12).

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6).

Dellaert, F., Seitz, S., Thorpe, C., and Thrun, S. (2000). Structure from Motion without Correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Duda, R. O. and Hart, P. E. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM (CACM)*, 15(1).

Farenzena, M., Fusiello, A., and Gherardi, R. (2009). Structure-and-Motion Pipeline on a Hierarchical Cluster Tree. In *IEEE International Conference on Computer Vision (ICCV) Workshops*.

Fischler, M. A. and Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM (CACM)*, 24(6).

Frahm, J.-M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., and Pollefeys, M. (2010). Building Rome on a Cloudless Day. In *European Conference on Computer Vision (ECCV)*.

Frahm, J.-M., Heinly, J., Zheng, E., Dunn, E., Fite-Georgel, P., and Pollefeys, M. (2013). Geo-Registered 3D Models from Crowdsourced Image Collections. *Geo-Spatial Information Science*, 16(1).

Frahm, J.-M. and Pollefeys, M. (2006). RANSAC for (Quasi-)Degenerate Data (QDEGSAC). In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009). Manhattan-World Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Towards Internet-Scale Multi-View Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Furukawa, Y. and Ponce, J. (2010). Accurate, Dense, and Robust Multi-View Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(8).

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining Data Streams: A Review. *ACM SIGMOD Record*, 34(2).

Gallup, D., Frahm, J.-M., and Pollefeys, M. (2010a). A Heightmap Model for Efficient 3D Reconstruction from Street-Level Video. In *3D Data Processing, Visualization, and Transmission (3DPVT)*.

Gallup, D., Frahm, J.-M., and Pollefeys, M. (2010b). Piecewise Planar and Non-Planar Stereo for Urban Scene Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gallup, D., Pollefeys, M., and Frahm, J.-M. (2010c). 3D Reconstruction Using an n-Layer Heightmap. In *German Association for Pattern Recognition (DAGM)*.

Gao, X.-S., Hou, X.-R., Tang, J., and Cheng, H.-F. (2003). Complete Solution Classification for the Perspective-Three-Point Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(8).

Gherardi, R., Farenzena, M., and Fusiello, A. (2010). Improving the Efficiency of Hierarchical Structure-and-Motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. (2007). Multi-View Stereo for Community Photo Collections. In *IEEE International Conference on Computer Vision (ICCV)*.

Google (2015). Google Earth Map Data: US Department of State Geographer, Image Landsat, GeoBasis-DE/BKG.

Goyal, A. and Daumé, III, H. (2011). Lossy Conservative Update (LCU) Sketch: Succinct Approximate Count Storage. In *AAAI Conference on Artificial Intelligence*.

Han, X., Leung, T., Jia, Y., Sukthankar, R., and Berg, A. C. (2015). MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Häne, C., Savinov, N., and Pollefeys, M. (2014). Class Specific 3D Object Shape Priors Using Surface Normals. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Häne, C., Zach, C., Cohen, A., Angst, R., and Pollefeys, M. (2013). Joint 3D Scene Reconstruction and Class Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*.

Hartley, R. and Sturm, P. (1996). Triangulation.

Hartley, R. I. (1997). In Defense of the Eight-Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(6).

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

Hartmann, W., Havlena, M., and Schindler, K. (2014). Predicting Matchability. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Havlena, M. and Schindler, K. (2014). VocMatch: Efficient Multiview Correspondence for Structure from Motion. In *European Conference on Computer Vision (ECCV)*.

Hays, J. and Efros, A. A. (2008). `IM2GPS`: Estimating Geographic Information from a Single Image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heath, K., Gelfand, N., Ovsjanikov, M., Aanjaneya, M., and Guibas, L. J. (2010). Image Webs: Computing and Exploiting Connectivity in Image Collections. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heinly, J., Dunn, E., and Frahm, J.-M. (2012). Comparative Evaluation of Binary Features. In *European Conference on Computer Vision (ECCV)*.

Heinly, J., Dunn, E., and Frahm, J.-M. (2014a). Correcting for Duplicate Scene Structure in Sparse 3D Reconstruction. In *European Conference on Computer Vision (ECCV)*.

Heinly, J., Dunn, E., and Frahm, J.-M. (2014b). Recovering Correct Reconstructions from Indistinguishable Geometry. In *International Conference on 3D Vision (3DV)*.

Heinly, J., Schönberger, J. L., Dunn, E., and Frahm, J.-M. (2015). Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Helmke, U., Hüper, K., Lee, P. Y., and Moore, J. (2007). Essential Matrix Estimation Using Gauss-Newton Iterations on a Manifold. *International Journal of Computer Vision (IJCV)*, 74(2).

Irschara, A., Zach, C., and Bischof, H. (2007). Towards Wiki-Based Dense City Modeling. In *International Conference on Computer Vision (ICCV) Workshop on Virtual Representations and Modeling of Large-Scale Environments (VRML)*.

Irschara, A., Zach, C., Frahm, J.-M., and Bischof, H. (2009). From Structure-from-Motion Point Clouds to Fast Location Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Isenburg, M. and Lindstrom, P. (2005). Streaming Meshes. In *Proceedings of IEEE Visualization*.

Jégou, H. and Chum, O. (2012). Negative Evidences and Co-occurences in Image Retrieval: The Benefit of PCA and Whitening. In *European Conference on Computer Vision (ECCV)*.

Ji, D., Dunn, E., and Frahm, J.-M. (2014). 3D Reconstruction of Dynamic Textures in Crowd Sourced Data. In *European Conference on Computer Vision (ECCV)*.

Jiang, N., Cui, Z., and Tan, P. (2013). A Global Linear Method for Camera Pose Registration. In *IEEE International Conference on Computer Vision (ICCV).*

Jiang, N., Tan, P., and Cheong, L.-F. (2011). Multi-View Repetitive Structure Detection. In *IEEE International Conference on Computer Vision (ICCV).*

Jiang, N., Tan, P., and Cheong, L.-F. (2012). Seeing Double Without Confusion: Structure-from-motion in Highly Ambiguous Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Johnson, S. C. (1967). Hierarchical Clustering Schemes. *Psychometrika*, 32(3).

Kalantidis, Y., Tolias, G., Avrithis, Y., Phinikettos, M., Spyrou, E., Mylonas, P., and Kollias, S. (2011). VIRaL: Visual Image Retrieval and Localization. *Multimedia Tools and Applications*, 51.

Kang, L., Wu, L., and Yang, Y.-H. (2014). Robust Multi-View L_2 Triangulation via Optimal Inlier Selection and 3D Structure Refinement. *Pattern Recognition*, 47(9).

Karp, R. M., Shenker, S., and Papadimitriou, C. H. (2003). A Simple Algorithm for Finding Frequent Elements in Streams and Bags. *ACM Transactions on Database Systems*, 28(1).

Kim, S. J., Gallup, D., Frahm, J.-M., Akbarzadeh, A., Yang, Q., Yang, R., Nistér, D., and Pollefeys, M. (2007). Gain Adaptive Real-Time Stereo Streaming. In *International Conference on Computer Vision Systems (ICVS).*

Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR).*

Klingner, B., Martin, D., and Roseborough, J. (2013). Street View Motion-from-Structure-from-Motion. In *IEEE International Conference on Computer Vision (ICCV).*

Köser, K., Zach, C., and Pollefeys, M. (2011). Dense 3D Reconstruction of Symmetric Scenes from a Single Image. In *German Association for Pattern Recognition (DAGM).*

Kushnir, M. and Shimshoni, I. (2014). Epipolar Geometry Estimation for Urban Scenes with Repetitive Structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(12).

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Lee, S. and Liu, Y. (2012). Curved Glide-Reflection Symmetry Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(2).

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). EPnP: An Accurate $O(n)$ Solution to the PnP Problem. *International Journal of Computer Vision (IJCV)*, 82(2).

Leutenegger, S., Chli, M., and Siegwart, R. (2011). BRISK: Binary Robust Invariant Scalable Keypoints. In *IEEE International Conference on Computer Vision (ICCV)*.

Li, Q., Wang, G., Liu, J., and Chen, S. (2009). Robust Scale-Invariant Feature Matching for Remote Sensing Image Registration. *IEEE Geoscience and Remote Sensing Letters*, 6(2).

Li, X., Wu, C., Zach, C., Lazebnik, S., and Frahm, J.-M. (2008). Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. In *European Conference on Computer Vision (ECCV)*.

Li, Y., Snavely, N., Huttenlocher, D., and Fua, P. (2012). Worldwide Pose Estimation Using 3D Point Clouds. In *European Conference on Computer Vision (ECCV)*.

Liu, J. and Liu, Y. (2013). GRASP Recurring Patterns from a Single View. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, Y., Hel-Or, H., Kaplan, C. S., and Gool, L. V. (2010). Computational Symmetry in Computer Vision and Computer Graphics. *Foundations and Trends in Computer Graphics and Vision*, 5(1-2).

Longuet-Higgins, H. (1981). A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature*, 293.

Lou, Y., Snavely, N., and Gehrke, J. (2012). MatchMiner: Efficient Spanning Structure Mining in Large Image Collections. In *European Conference on Computer Vision (ECCV)*.

Lowe, D. G. (2004). Distinctive Image features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2).

Lui, V. and Drummond, T. (2013). An Iterative 5-pt Algorithm for Fast and Robust Essential Matrix Estimation. In *British Machine Vision Conference (BMVC)*.

Manku, G. S. and Motwani, R. (2002). Approximate Frequency Counts over Data Streams. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*.

Martinec, D. and Pajdla, T. (2007). Robust Rotation and Translation Estimation in Multiview Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference (BMVC)*.

Matzen, K. and Snavely, N. (2014). Scene Chronology. In *European Conference on Computer Vision (ECCV)*.

Meeker, M. (2014). Internet Trends 2014 - Code Conference. http://www.kpcb.com/internet-trends.

Memories (2014). 1000 Memories Blog. http://blog.1000memories.com.

Mikolajczyk, K. and Schmid, C. (2004). Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision (IJCV)*, 60(1).

Mikolajczyk, K. and Schmid, C. (2005). A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10).

Mills, S. (2013). Relative Orientation and Scale for Improved Feature Matching. In *IEEE International Conference on Image Processing (ICIP)*.

Nistér, D. (2003). An Efficient Solution to the Five-Point Relative Pose Problem. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nistér, D. and Stewénius, H. (2006). Scalable Recognition with a Vocabulary Tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Oliva, A. and Torralba, A. (2001). Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision (IJCV)*, 42(3).

Park, H. S., Shiratori, T., Matthews, I., and Sheikh, Y. (2010). 3D Reconstruction of a Moving Point from a Series of 2D Projections. In *European Conference on Computer Vision (ECCV)*.

Pauly, M., Mitra, N., Wallner, J., Pottmann, H., and Guibas, L. (2008). Discovering Structural Regularity in 3D Geometry. *ACM Transactions on Graphics (TOG), Proceedings of SIGGRAPH*, 27(3).

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pollefeys, M., Frahm, J.-M., Fraundorfer, F., Zach, C., Wu, C., Clipp, B., and Gallup, D. (2010). Challenges in Wide-Area Structure-from-Motion. *IPSJ Transactions on Computer Vision and Applications*, 2.

Raginsky, M. and Lazebnik, S. (2009). Locality-Sensitive Binary Codes from Shift-Invariant Kernels. In *Advances in Neural Information Processing Systems (NIPS)*.

Raguram, R., Chum, O., Pollefeys, M., Matas, J., and Frahm, J.-M. (2013). USAC: A Universal Framework for Random Sample Consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(8).

Raguram, R., Frahm, J.-M., and Pollefeys, M. (2008). A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In *European Conference on Computer Vision (ECCV)*.

Raguram, R., Tighe, J., and Frahm, J.-M. (2012). Improved Geometric Verification for Large Scale Landmark Image Collections. In *British Machine Vision Conference (BMVC)*.

Raguram, R., Wu, C., Frahm, J.-M., and Lazebnik, S. (2011). Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. *International Journal of Computer Vision (IJCV)*, 95(3).

Roberts, R., Sinha, S. N., Szeliski, R., and Steedly, D. (2011). Structure from Motion for Scenes with Large Duplicate Structures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rosten, E. and Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision (ECCV)*.

Rosten, E., Porter, R., and Drummond, T. (2010a). Faster and Better: A Machine Learning Approach to Corner Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32.

Rosten, E., Reitmayr, G., and Drummond, T. (2010b). Improved RANSAC Performance Using Simple, Iterative Minimal-Set Solvers. arXiv.

Rousseeuw, P. J. and Leroy, A. M. (2005). *Robust Regression and Outlier Detection*. John Wiley & Sons.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An Efficient Alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*.

Sattler, T., Leibe, B., and Kobbelt, L. (2009). SCRAMSAC: Improving RANSAC's Efficiency with a Spatial Consistency Filter. In *IEEE International Conference on Computer Vision (ICCV)*.

Sattler, T., Leibe, B., and Kobbelt, L. (2011). Fast Image-Based Localization using Direct 2D-to-3D Matching. In *IEEE International Conference on Computer Vision (ICCV)*.

Savasere, A., Omiecinski, E., and Navathe, S. B. (1995). An Efficient Algorithm for Mining Association Rules in Large Databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*.

Schaffalitzky, F. and Zisserman, A. (2002). Multi-View Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps?". In *European Conference on Computer Vision (ECCV)*.

Schindler, G., Krishnamurthy, P., Lublinerman, R., Liu, Y., and Dellaert, F. (2008). Detecting and Matching Repeated Patterns for Automatic Geo-tagging in Urban Environments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schönberger, J. L., Berg, A. C., and Frahm, J.-M. (2015a). PAIGE: PAirwise Image Geometry Encoding for Improved Efficiency in Structure-from-Motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schönberger, J. L., Radenović, F., Chum, O., and Frahm, J.-M. (2015b). From Single Image Query to Detailed 3D Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shamma, D. A. (2014). One Hundred Million Creative Commons Flickr Images for Research. http://labs.yahoo.com/news/yfcc100m.

Shi, J. and Malik, J. (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8).

Shum, H.-Y., Ke, Q., and Zhang, Z. (1999). Efficient Bundle Adjustment with Virtual Key Frames: A Hierarchical Approach to Multi-Frame Structure from Motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sinha, S. N., Ramnath, K., and Szeliski, R. (2012). Detecting and Reconstructing 3D Mirror Symmetric Objects. In *European Conference on Computer Vision (ECCV)*.

Sinha, S. N., Steedly, D., and Szeliski, R. (2010). A Multi-Stage Linear Approach to Structure from Motion. In *European Conference on Computer Vision (ECCV) Workshop on Reconstruction and Modeling of Large-Scale 3D Virtual Environments*.

Snavely, N., Garg, R., Seitz, S. M., and Szeliski, R. (2008a). Finding Paths through the World's Photos. *ACM Transactions on Graphics (TOG), Proceedings of SIGGRAPH*, 27(3).

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo Tourism: Exploring Photo Collections in 3D. *ACM Transactions on Graphics (TOG), Proceedings of SIGGRAPH*, 25(3).

Snavely, N., Seitz, S. M., and Szeliski, R. (2007). Modeling the World from Internet Photo Collections. *International Journal of Computer Vision (IJCV)*, 80(2).

Snavely, N., Seitz, S. M., and Szeliski, R. (2008b). Skeletal Graphs for Efficient Structure from Motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Stewénius, H., Gunderson, S. H., and Pilet, J. (2012). Size Matters: Exhaustive Geometric Verification for Image Retrieval. In *European Conference on Computer Vision (ECCV)*.

Strecha, C., Fransens, R., and Gool, L. V. (2006). Combined Depth and Outlier Estimation in Multi-View Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Strecha, C., Pylvänäinen, T., and Fua, P. (2010). Dynamic and Scalable Large Scale Image Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2015). The New Data and New Challenges in Multimedia Research. *arXiv:1503.01817 [cs.MM]*.

Toivonen, H. (1994). Sampling Large Databases for Association Rules. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*.

Tola, E., Lepetit, V., and Fua, P. (2010). DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(5).

Torii, A., Sivic, J., Pajdla, T., and Okutomi, M. (2013). Visual Place Recognition with Repetitive Structures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 Million Tiny Images: A Large Dataset for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(11).

Trzcinski, T. and Lepetit, V. (2012). Efficient Discriminative Projections for Compact Binary Descriptors. In *European Conference on Computer Vision (ECCV)*.

Turcot, P. and Lowe, D. (2009). Better Matching with Fewer Features: The Selection of Useful Features in Large Database Recognition Problems. *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data (WS-LAVD)*.

Žbontar, J. and LeCun, Y. (2015). Computing the Stereo Matching Cost with a Convolutional Neural Network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Watts, D. J. and Strogatz, S. H. (1998). Collective Dynamics of 'Small-World' Networks. *Nature*, 393(6684).

Weyand, T., Tsai, C.-Y., and Leibe, B. (2015). Fixing WTFs: Detecting Image Matches Caused by Watermarks, Timestamps, and Frames in Internet Photos. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

Wilson, K. and Snavely, N. (2013). Network Principles for SfM: Disambiguating Repeated Structures with Local Context. *IEEE International Conference on Computer Vision (ICCV)*.

Wilson, K. and Snavely, N. (2014). Robust Global Translations with 1DSfM. In *European Conference on Computer Vision (ECCV)*.

Wu, C. (2007). SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT). http://cs.unc.edu/˜ccwu/siftgpu.

Wu, C. (2013). Towards Linear-Time Incremental Structure from Motion. In *International Conference on 3D Vision (3DV)*.

Wu, C., Agarwal, S., Curless, B., and Seitz, S. M. (2012). Schematic Surface Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, C., Frahm, J.-M., and Pollefeys, M. (2011). Repetition-Based Dense Single-View Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

YouTube (2014). Statistics. https://www.youtube.com/yt/press/statistics.html.

Zach, C., Irschara, A., and Bischof, H. (2008). What Can Missing Correspondences Tell Us about 3D Structure and Motion? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zach, C., Klopschitz, M., and Pollefeys, M. (2010). Disambiguating Visual Relations Using Loop Constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zamir, A. R., Ardeshir, S., and Shah, M. (2014). GPS-Tag Refinement using Random Walks with an Adaptive Damping Factor. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zelnik-Manor, L. and Perona, P. (2004). Self-Tuning Spectral Clustering. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhang, S., Tian, Q., Hua, G., Zhou, W., Huang, Q., Li, H., and Gao, W. (2011). Modeling Spatial and Semantic Cues for Large-Scale Near-Duplicated Image Retrieval. *Computer Vision and Image Understanding*, 115(3).

Zhao, P. and Quan, L. (2011). Translation Symmetry Detection in a Fronto-Parallel View. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhao, P., Yang, L., Zhang, H., and Quan, L. (2012). Per-Pixel Translational Symmetry Detection, Optimization, and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zheng, E., Dunn, E., Jojic, V., and Frahm, J.-M. (2014a). PatchMatch Based Joint View Selection and Depthmap Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zheng, E., Dunn, E., Raguram, R., and Frahm, J.-M. (2012). Efficient and Scalable Depthmap Fusion. In *British Machine Vision Conference (BMVC)*.

Zheng, E., Wang, K., Dunn, E., and Frahm, J.-M. (2014b). Joint Object Class Sequencing and Trajectory Triangulation (JOST). In *European Conference on Computer Vision (ECCV)*.

Zhu, G., Yan, S., and Ma, Y. (2010). Image Tag Refinement Towards Low-Rank, Content-Tag Prior and Error Sparsity. In *International Conference on Multimedia*.