6DOF camera localization through a short image sequence

Master's thesis in Signals, Systems and Mechatronics

Jonas Garsten & Ivar Wikenstedt

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, February 2, 2020

Master's thesis February 2, 2020

6DOF camera localization through a short image sequence

Jonas Garsten & Ivar Wikenstedt

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Signal processing and Biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
Gothenburg, Sweden, February 2, 2020

6DOF camera localization through a short image sequence
Jonas Garsten & Ivar Wikenstedt

Supervisor: Torsten Sattler, Department of Electrical Engineering
Examiner: Torsten Sattler, Department of Electrical Engineering

Master's Thesis February 2, 2020
Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
SE-412 96 Gothenburg

Cover: Visualization of the full reconstruction of CMU-Seasons-Extended [51] 3D-models using Meshlab [12], where each slice is represented by a unique colorcode.

Typeset in LaTeX
Gothenburg, Sweden, February 2, 2020

6DOF camera localization through a short image sequence
Jonas Garsten & Ivar Wikenstedt
Department of Electrical Engineering
Chalmers University of Technology

Keywords: Visual localization, Structure from motion, Computer vision, Sequential localization

Acknowledgements

Thank you to Torsten Sattler our supervisor and examiner for all the discussions and feedback he assisted with. *Test early and test often.*

To Rasmus Lundin for the endless supply of coffee and Linux support.

To David Frisk for creating the LaTeX-template used in this report, licensed under *CC BY 4.0*.

Authors advice: When in doubt, run RANSAC! Jonas Garsten & Ivar Wikenstedt,

Gothenburg, February 2, 2020

Contents

Abstract

Camera localization is a core problem in the computer vision field and is a hot topic as cameras are more commonly integrated into navigation applications. With recent technological advancements it is now possible to store and process large data streams locally or in the cloud. One such example would be video streams. Common localization practices only localize one image at a time. The generalized camera describes image observations as ray origins and ray directions in 3D space. Gaining the advantage of localizing many images at the same time and can use information from many images to do so. In this paper we develop a pipeline for camera localization using the generalized camera and evaluate how localization accuracy is affected by different sequence lengths. We present results showing our pipeline outperforming state of the art localization pipelines on the Cambridge Landmarks Street and St Mary's Church datasets.

List of Figures

List of Tables

1
Introduction

Visual localization is a method of identifying a camera's location and orientation given an image from the camera, also known as the camera pose. This problem is core to computer vision with applications such as Structure from Motion (SfM) [1, 53], augmented reality (AR) [42] and simultaneous localization and mapping (SLAM) [13, 63] to name a few.

Visual localization can be done in a multitude of ways depending on the application and aim. A common approach is to find local 2D features in an image which can be compared and matched to features in a 3D map. By matching enough features between the two, the camera can be localized [32, 59]. A more recent approach that has gained popularity with the development of machine learning is to substitute conventional methods with machine learning. One can use machine learning to generate 2D features [56], match 2D and 3D features [14] or to skip these steps and localize the camera directly [29].

In this paper, we will describe a visual localization pipeline that differentiates itself by using the generalized camera model. The generalized camera model describes a match between a 2D image point and a 3D model point, hereafter referred to as 2D-3D match, as a ray origin and ray direction. As matches are described in 3D space it is possible to use matches from different images to localize many images at a time. This makes it possible to localize more than one camera at a time by combining the information of several images.

This pipeline takes an image sequence as input and estimates the camera trajectory as a local point cloud by doing Structure from Motion (SfM). The points in the local point cloud are then matched to points in a global point cloud, which is commonly referred to as a structure-based method. To find the camera trajectory in the image sequence, all cameras in the sequence are transformed into one generalized camera model. Then we localize the sequence by finding a similarity transform that maps the generalized camera to the global point cloud, based on our point correspondences.

The main question of this research is to draw a conclusion whether or not using an image sequence can improve camera pose estimation results using the generalized camera model. Additional questions that we want to answer is the impact of memory compressed 3D-point model and splitting up the query sequence into fixed sequence length.

In this report we make the following **contributions**: (*i*) A localization pipeline using a 3D structure-based method with the properties of a sequential-based method. This will use a 3D-point map and an image sequence query or otherwise known as a video to compute a trajectory. (*ii*) An evaluation of our sequential localization algorithm on datasets commonly used for localization comparison. (*iii*) An exper-

imental evaluation of the effects using different sequence lengths and compressed 3D-point clouds on our pipeline.

1.1 Related Work

This thesis main objective is to implement and evaluate how the use of generalized cameras can improve visual localization. Therefor we make use of already existing methods and theory when implementing our pipeline.

1.1.1 Ways to localize images

Camera localization has been a developing area for many years where different methods have emerged based on new areas of application and current technological advances. An intuitive localization type is 2D image-based localization, also known as image retrieval. This localization method makes use of existing information associated with images such as GPS location [7, 45] and either assigns the same position to the query images or computes the position [44] by matching features present in both images and finding a transformation between them. The challenge is to find the best matching image from the library in an efficient way. When libraries grow to hundreds of thousands or millions of images [10, 11] exhaustive matching between images becomes to slow and tasks that require localization in real time become impossible to solve. This has made the use of Bag of Words (BoW) and tree structures [7, 24, 54, 55] increasingly popular because of the increased computational speed compared to exhaustive methods. One of these real time applications is simultaneous localization and mapping (SLAM). A common problem for SLAM is compounding errors. A small error in position estimation will grow larger with time as more and more estimations are done. Image retrieval has had great success in dealing with this through loop closure [22, 65]. Loop closure occurs when a place is visited a second time and is recognized as a previously visited location.

2D image based localization relies on images with location information such as GPS tags to estimate camera poses for new image. 3D structure-based localization removes the need for poses to be attached to images and calculates the camera pose from a 3D reference model instead. Most structure-based localization algorithms make use of the same general structure of extracting features from images and matching these to points in a point cloud model where with enough matches, a position and pose can be computed. This is the same independent of geographical scope, ranging from neighborhood [60] to world wide [30] localization. Even tough geographical scope is completely different, the same localization challenges appear. In urban environments, many of the structures have similar patterns and colors such as brick walls. This occurs in rural areas as well but instead of structures it is vegetation that looks similar. With changing seasons and lighting conditions added on top, the task of finding robust and unique matches become increasingly challenging [30, 49, 66].
A wide variety of methods have been proposed to increase localization accuracy on increasingly challenging data sets. The most common way of matching features is

to match image-to-model. In [31] it is proposed that doing the opposite, matching model-to-image will increase localization accuracy. Since this method utilizes that similar images will have similar features. This is done by creating different scenes that images can be matched against. This has the potential of greatly reducing the number of feature comparisons. Similarly, [32] makes use of co-visibility between 3D points to not consider every match in isolation instead of evaluating if an image has many closely related matches. In a co-visibility graph the nodes are 3D points where two co-visible points, p_i, p_j, in the graph are bi-directional connected with edges e_{ij} and e_{ji}. The wight of the edge is calculated with

$$c_{ij} = \frac{|A_i \cap A_j|}{|A_j|} \tag{1.1}$$

where c_{ij} is the weight if edge e_{ij}, A_i and A_j is the set if database images containing point p_i and p_j.
As with image retrieval, there have been suggestions to use BoW for structure-based localization. BoW is used in computer vision to index similar images or features under words. This will make it easier to find which images are similar to one another. One proposal [48] is to use a fine vocabulary, where a BoWs vocabulary refers to the resolution of the grouping of images, in combination with co-visibility constraints to localize cameras and present results of greatly reduced memory requirements for large datasets.

A more recent development in the computer vision field is to switch out traditional practices for machine learning alternatives. Machine learning uses neural networks that are modeled after neurons in the human brain. These networks are trained on massive amounts of data to make certain connection between input data and desired output. A simple example would be that given an image as input, the network will output the probabilities of which country it was taken in. The two main applications of neural networks to camera localization are (*i*), to directly predict a cameras pose from the input image. (*ii*), to replace some part of the camera localization pipeline with a neural network. In the case of (*i*), [29] uses a convolutional neural network (CNN) to estimate camera poses and displays result of more robust pose estimation. Walch et al. [64] built on this idea by adding a long short-term memory (LSTM) [23]. Adding a LSTM helps in preventing overfitting, a phenomenon in machine learning where the network becomes adapted to the training data and its ability to make correct predictions on new data decrease.
As for *ii*, in [39] normal feature detection is replaced by synthetic view point generation that is then evaluated with the SeqSlam [40] algorithm. There is a problem in camera localization where a change in view point, such as lateral shifts, can negatively affect localization results. The proposed solution has a network trained to estimate depth maps, assigning how far objects are from the camera. With this depth map, it is possible to generate new view points that are shifted laterally. All views are then evaluated separately for better results.

In many modern application of camera localization, the images to be localized are not disordered but come from a video stream where each consecutive image should

have a lot in common. This sequential data flow is common in robots, autonomous cars and SLAM. The sequential nature of query images can be exploited by estimating movement trajectories [13] from the most recent pose estimations or to not only match one image at a time but match an entire sequence of images [40] to increase the likelihood of good matches. This method is referred to as sequential-based.

In most of the previous works, there has only been cases where a single camera is used. This does imply that research into utilizing multi-camera systems have been overlooked. Multi-camera systems are not a new idea. The topic has been researched since the early two thousands [16, 26]. Most of the work up till now have focused on using multiple cameras for SLAM [9, 26, 63] where the benefit of more cameras is a wider filed of view, enabling the multi-camera rig to analyze more structure at the same time.
In image localization where there is no rig or robot to add more cameras to the approach has instead been to try and represent cameras in a more general way to localize multiple images as if they were only a single image. This was introduced by Grossberg and Nayar [17] who proposed a way of modeling pixels in an image as raxels. A Raxel is the virtual representation of a photo sensitive element measuring the light intensity a pixel is exposed to.The idea of modeling light sensors was then further expanded upon by [43] who introduced the generalized camera model. The generalized camera model switched out raxels for Plücker vector, simplifying the camera model by, for example, removing the modulation of light intensity. Recently, Sweeney et al. [59] applied the generalized camera to large scale SfM and displayed impressive results reconstructing the city of Rome in only twenty two minutes.

1.1.2 Point Cloud Compression

With generalized cameras having the ability to localize more than one image at a time by using the combined information present in all images, it is of interest to know how this added information can be used for camera localization. One application is point cloud compression. As camera localization moves towards applications requiring real-time performance [18, 34] and the hardware in current mobile devices is not yet powerful enough for running high preforming localization algorithms in real-time. An alternative is then to offload the localization to a server [38]. The issue with this is that visual information captured by the mobile device has to be sent to the server requiring fast transfer speeds. To minimize transfer speed requirements one can use point cloud compression to reduce the total data that has to be transmitted. Point cloud compression is generally done in two ways. Either the number of points are reduced [8, 31] or the point descriptor is compressed [18, 48]. Both approaches try to achieve the same result in reducing the memory requirements for point cloud models. When selecting a subset of points the goal is to create a more compact description of a model without losing significant localization performance. In the method by S. Cao and N. Snavely [8] compression is done by incrementally selecting points which have good coverage and are distinct until all images can see k points. How good coverage a point has is not determined by how many images it is visible in but rather a probability p_{ij} of point i being visible in

image j is defined. This is similar to simply selecting the points with best coverage in the already registered images. However, it was found that using a probabilistic approach contributed to finding good feature matches. Points are then selected to maximize the probability of each images seeing k points. To increase the likelihood of picking distinct points the distance between a point and already picked points in descriptor space is also considered. Compressing the descriptors is tricky as the objective is to reduce the size while maintaining the distinctiveness of the descriptors. Intuitively, reducing the size of all descriptors would result in a less descriptive model representation and result in poor localization and [34] is an example of compression affecting localization accuracy. However, the application here is a SLAM system and the worse image localization accuracy did not significantly impact its pose estimation performance.They employ product quantization [25] which reduces a high dimensional descriptor into several descriptors of lower dimensionality. The low dimensional descriptors are clustered using k-means clustering and each cluster is then represented with a single descriptor. In the original descriptor each low dimensional descriptor will be exchanged with a descriptor representing the cluster it belongs to.
Given the increase in demand for real-time solutions in mobile applications, viewing the effects of using a compressed point cloud model in our pipeline is in our interest.

1.2 Content of thesis

This thesis is split into four sections: Theory, Method, Results and Conclusion. In Theory we will explain how camera localization works and what a generalized camera is. Method will explain the different parts of our pipeline and specify what software and methods we use. Results will include our findings and discussion regarding our results. Finally we present our findings in short in the Conclusion section and will discuss what more can be done in Future Work.

2
Theory

Visual localization has been a research topic for decades where many methods have been developed to improve localization accuracy. In this chapter the underlying theory for visual localization will be presented to give the reader information necessary to understand the topic of visual localization.
Our pipeline boils down to two steps, creating models using Structure from Motion and localizing images by comparing Structure from Motion models. The first chapters will explain how cameras are modeled and how these models can be used to calculate camera positions as well as to build 3D models of a scene. The later chapters describe how the generalized camera works and how images can be localized using this model. Lastly a method called RANSAC will be explained and why it is necessary to use RANSAC when the data is noisy or contain outliers.

2.1 Camera Model

A camera model describes the transformation of points in the world into an image. There are many different types of camera models depending on for example which type of lens a camera is using. The simplest camera model is the pinhole camera model. This camera model has a camera center C where all image rays intersect. Global points are projected, transformed to the image plane, by following the ray connecting the point to the camera center until it intersects the image plane. The transformation from world coordinates into a pinhole camera can be expressed with the projection matrix

$$P = K \begin{bmatrix} R & t \end{bmatrix} \ , \tag{2.1}$$

where R is a 3x3 rotation matrix, t is a 3x1 translation vector and K is a 3x3 calibration matrix. R and t are called the extrinsic parameters of a projection matrix containing information about camera orientation and position while K is the intrinsic parameters describing the camera. As seen in Figure 2.1 the rotation R and translation t transform global point into a camera coordinate system.
From a camera coordinate system, points can be projected by applying the camera calibration matrix K as seen in Figure 2.2. The calibration matrix for a pinhole camera looks like

$$K = \begin{bmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \ , \tag{2.2}$$

where the parameters are; focal length f, the distance between the center of projection and the image plane. Skew s, which describes tilt of the pixels in the image.

Aspect ratio γ, determining the pixel x-y ratio and finally x_0 and y_0 maps the origin from where the z-axis intersects the image plane to the upper left pixel. When applying the calibration matrix, the distances are transformed into pixel values. With this, an image pixel coordinate can be found by reprojecting its corresponding global point $\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ into the image. The projection matrix would be applied as

$$\lambda \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\bar{x}} = P \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\bar{X}} , \tag{2.3}$$

where $[x \quad y]^T$ is an image pixel coordinate and λ is a scaling parameter. This compact way of applying R,t and K requires the use of homogeneous coordinates. Homogeneous coordinates lift a N-dimensional point to a N+1 dimensional line. This allows for translations and projections to be expressed as matrix multiplications. The N-dimensional point can after a transformation be retrieved by dividing by the last entry and then dropping the last entry.
If we instead want to transform a point from an image into world coordinates, a problem occurs. To project a point, we apply P and divide with the last entry. But when transforming a point from the image into the world the last entry is not known. Calculating the last entry is not possible with a single image. For this a second image has to be added.

2.2 Finding point coordinates and projection matrices

Given two cameras with different position, if both cameras can see the same 3D point its exact position can be computed. Because the point is present in both images, there are two different lines describing where it is located in 3D space. Its position is

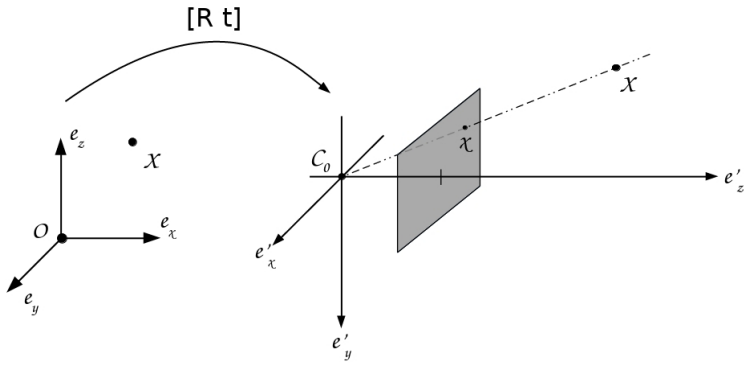

Figure 2.1: Visualization of applying the extrinsic parameters of camera matrix P will transform a 3D point from global coordinates into the camera coordinate system.

then where both lines intersect (Figure 2.3). In essence, an equation system is solved by inserting image and 3D point information in Equation (2.3). The same principle can be applied when solving camera matrices. A camera matrix has the dimensions of 3x4, containing twelve variables. We only have to solve eleven variables as it is determined up to an arbitrary scale. To solve eleven variables, eleven equations are required. The camera equation (2.3) gives three equations for every point but λ also has to be solved for each new point. The number of points required then becomes

$$3n \geq 11 + n \Rightarrow n \geq 5.5 \tag{2.4}$$

A simple approach to solve systems of linear equations is called direct linear transform (DLT) [20]. This approach expresses all equations in terms of a matrix multiplication and finds an approximate nullspace for the matrix. The DLT algorithm explained below is a slightly modified version [27] of the one in Hartley and Zissermans book [20]. First, lets express P as

$$P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{2.5}$$

where p_i, $i = 1, 2, 3$, are 1x4 row vectors from P. Insert this representation of P into (2.3) and it can be written as

$$p_1 X - \lambda x = 0 \tag{2.6}$$
$$p_2 X - \lambda y = 0 \tag{2.7}$$
$$p_3 X - \lambda = 0 \tag{2.8}$$

which in matrix form looks like

$$\begin{bmatrix} X^T & 0 & 0 & -x \\ 0 & X^T & 0 & -y \\ 0 & 0 & X^T & -1 \end{bmatrix} \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.9}$$

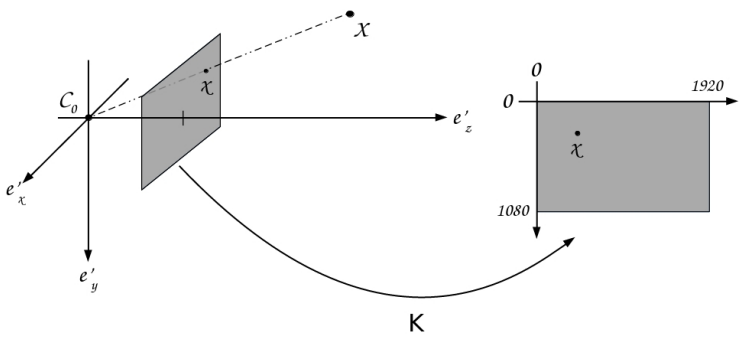

Figure 2.2: Applying the calibration matrix K to a point in camera coordinates will reproject it into the image as a pixel coordinate.

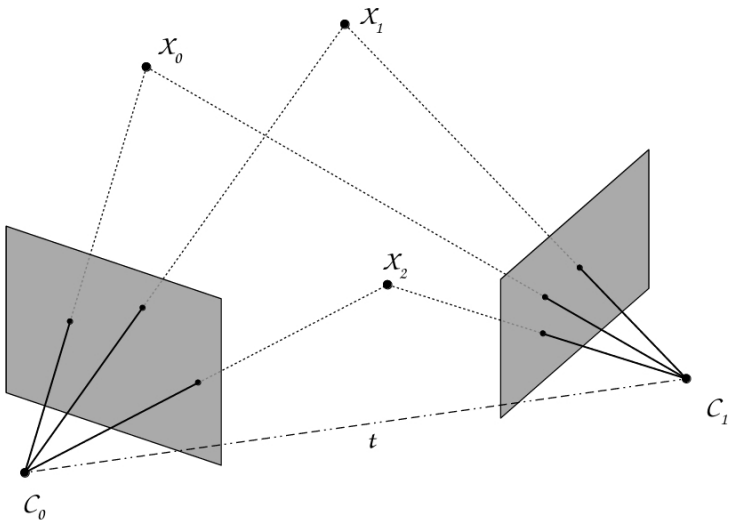

Figure 2.3: Visualization of how 3D point locations can be determined by finding correspondences in two separate image and evaluate where the rays intersect.

This is the matrix for only a single point. To find P at least 5.5 points are required according to equation 2.4. Since you can not use partial points, six points are required. If we include more points in equation 2.9 the matrix will be on the form

$$\underbrace{\begin{bmatrix} X_1^T & 0 & 0 & -x_1 & 0 & 0 & \cdots \\ 0 & X_1^T & 0 & -y_1 & 0 & 0 & \cdots \\ 0 & 0 & X_1^T & -1 & 0 & 0 & \cdots \\ X_2^T & 0 & 0 & 0 & -x_2 & 0 & \cdots \\ 0 & X_2^T & 0 & 0 & -y_2 & 0 & \cdots \\ 0 & 0 & X_2^T & 0 & -1 & 0 & \cdots \\ X_3^T & 0 & 0 & 0 & 0 & -x_3 & \cdots \\ 0 & X_3^T & 0 & 0 & 0 & -y_3 & \cdots \\ 0 & 0 & X_3^T & 0 & 0 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{bmatrix}}_{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}. \tag{2.10}$$

This equation can be solved by finding a non-zero vector in the nullspace of M. Since the scale of the solution is arbitrary the constraint $||v||^2 = 1$ can be applied. This system will most likely not have an exact solution and a least squares solution is found by minimizing

$$\min_{||v||^2=1} ||Mv||^2. \tag{2.11}$$

2.3 A Structure from Motion Pipeline

A Structure from Motion (SfM) pipeline is widely used to reconstruct 3D point models from images [1, 53, 59]. A SfM pipeline is divided into the following steps:

1. Extract image features and compute descriptors. To compare images against each other, it is possible to simply use individual pixel values or a blurred version to see if general patterns match. However, this yields poor results and unique features are instead computed.

2. Match descriptors. With sufficiently unique features, they can now be matched to determine if the same feature is present in multiple images.

3. Estimate camera positions and create point cloud model. Matched features can now be triangulated into 3D points. This can be done either by Incremental SfM [1] or Global SfM [41].

Feature extraction is commonly done by using SIFT [33] descriptors. This is done by sliding a Difference of Gaussian filter over the image and detecting local extrema for the filter. Descriptors [3, 33, 46] are used to describe feature points. A descriptor does not have to be made in any particular way but a common descriptor such as SIFT [33] uses gradient vectors to describe features. As seen in Figure 2.5, gradients are created by analyzing a small portion of the image and creating gradients to nearby pixels based on changes in light intensity. All gradients are then placed into the bins of multiple histograms which are concatenate to form a descriptor. When creating a descriptor, the goal is to make it as robust and unique as possible. A robust descriptor will not change in different lighting and weather conditions. This is an important characteristic for matching features in images taken over long periods of time. Uniqueness in the other hand is important as to not match descriptors which do not represent the same point, so called false positives.

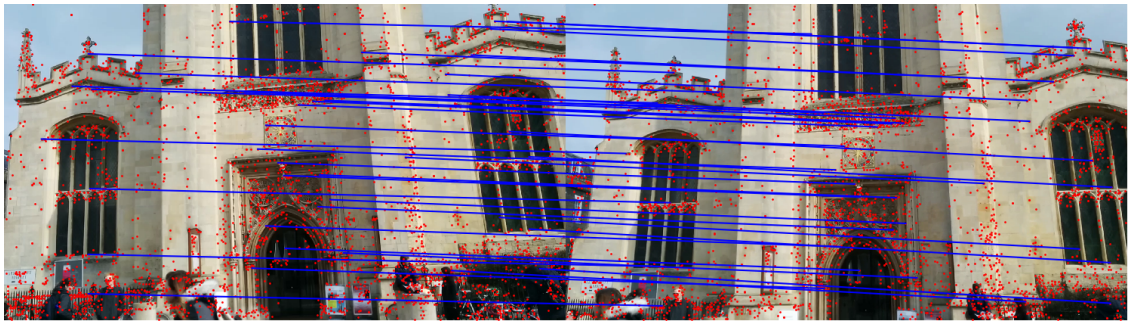

Figure 2.4: Extracted features (red dots) and the matched features between images (blue line). Both images are from the dataset Cambridge Landmarks St Mary's Church [29]. Feature extraction and matching was done with COLMAP [53].

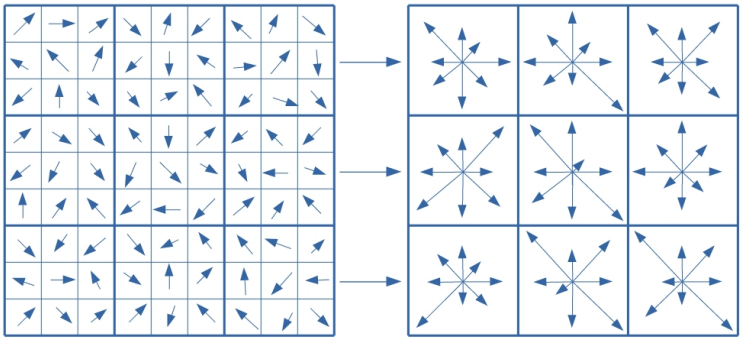

Figure 2.5: Image gradient computation into keypoint descriptors.

Matching descriptors to one another is a simple task in theory. As with simple 3D points, comparing descriptors is done by computing the Euclidean distance between them. What may not be obvious is that with high dimensional data such as descriptors, this distance will be large even to close matches [2]. Close matches will have many dimensions that match well but some will not and this is enough for distances to grow large. High dimensional data is also usually gathered in large quantities because of the larger space they reside in. This makes matching all descriptors a non trivial task that can take lots of time. To decrease the time spent matching, approximate search [35] is used instead of exhaustively looking for the best match. Approximate search can be done by arranging all data into a k-d tree (k-dimensional tree) [4]. Arranging the data in a tree structure makes it easier to only search data that is similar. Starting from the top of the tree, similar data can be found by following the path down the tree which has the best matches. To further ensure that good matches are found it is common to use Lowe´s ratio test [33] to discard matches that are probable to be false. The ratio test takes the ratio of the distance between the first and second nearest neighbors of the descriptor to be matched and evaluates if the ratio is less than a pre-defined threshold. The first nearest neighbor is the most similar descriptor to the matched descriptor while the second nearest neighbor is the second most similar descriptor. This is expressed as

$$\frac{|d_i - d_{N1}|^2}{|d_i - d_{N2}|^2} \leq tol \tag{2.12}$$

where d_i is the descriptor to be matched, d_{N1} and d_{N2} are the first and second nearest neighbor matches and tol is the predefined tolerance. A match passes the ratio test if the ratio is below the threshold. Employing the ratio test ensures that matched descriptors are more similar to each other than to other descriptors. This filters out matches that are unlikely to be correct.

Creating a point cloud model can be done in two ways, incrementally or globally. Incremental SfM [53] adds one image at a time to the reconstruction, incrementally expanding the model. Global SfM [41] estimates all camera positions at the same time by first constructing a graph connecting cameras viewing the same scene. Only incremental SfM will be explained here.

Incremental SfM first finds a good starting image pair from which an initial model can be created. Given this initial model, one by one images are added to the model. Each new image has to see n 3D points already in the model for a camera pose to be computed. The number of points depend on how much information is available about the camera and also determines which PnP solver to be used [67]. New point correspondences can then be triangulated and added to the point cloud. As more images are added, small errors will accumulate distorting the model. It is therefor common to perform optimization on both 3D point and camera poses with set intervals.

Optimization or more specifically bundle adjustment [62] is performed to minimize growing errors by shifting the position of 3D points and, depending on how much information about the camera is known, tuning the projection matrix. Localization error is usually quantified as the reprojection error of 3D points. It is computed by taking the square of all 2D image points subtracted by their respectively reprojected 3D point correspondence as

$$e = \sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\left(x_{ij} - \frac{p_i^1\mathbf{X}_j}{p_i^3\mathbf{X}_j}, y_{ij} - \frac{p_i^2\mathbf{X}_j}{p_i^3\mathbf{X}_j}\right)\right\|^2 \tag{2.13}$$

where $\begin{bmatrix} x_{ij} & y_{ij} \end{bmatrix}$ is the 2D image point in camera i corresponding to 3D point j.
p_i^k, $k = 1, 2, 3$ are the rows of the projection matrix P for camera i in Equation (2.5).
Unfortunately, minimizing Equation (2.13) is not simple as this is a non-linear least squares problem. Therefor gradient decent methods are used for local optimization.

2.4 The Generalized Camera

The generalized camera describes image observations as rays of light. These rays have two components, one vector for the direction in which the ray is pointing and and a point which lies on the ray. This parameterization is not unique to generalized cameras and can be used for pinhole cameras as well. Though, there are distinct differences between them. A pinhole camera has a center of projection/camera center. In Figure 2.6 this would be C_i, $i = 1, 2, 3$. A center of projection is where all rays intersect. A generalized camera does not have this constraint. In theory, every single ray could have its own center of projection. Another distinguishing feature is the lack of individual camera coordinate systems. For normal multi-camera systems such as the one in Figure 2.6, the pose of each individual camera has to be computed separately because the same rotation and translation will not transform 3D point correctly into all different camera coordinate systems. If the system is instead formulated as a generalized camera (Figure 2.7), all rays will be in the same coordinate frame and a single transformation is needed to find a solution.

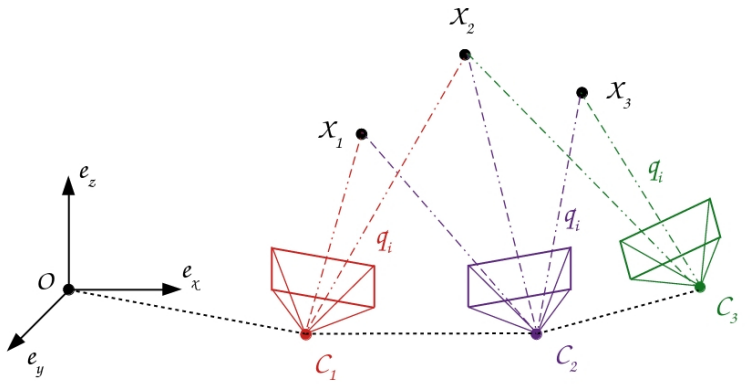

Figure 2.6: A graphical visualization of multiple cameras viewing the same points.

2.4.1 Pose estimation using the Generalized Camera

As mentioned before, generalized cameras represent image observations as rays of light. A ray-point correspondence can be expressed as

$$sq_i + \alpha_i \bar{r}_i = Rr_i + t, \quad i = 1, 2, 3, ... \tag{2.14}$$

where s, R, t is the scale, rotation and translation for the sought transformation from world coordinates into the generalized camera coordinate system [58]. The ray has ray origin q_i and ray direction $\bar{r}_i$. α_i stretches the rays to meet the 3D point r_i matched to ray i such that $\alpha_i = ||Rr_i + t - sq_i||$. The rotation can be parameterized using Cayley-Gibbs-Rodriguez to be represented with three unknowns. Translation plus scale are four more unknowns. When the query is constructed it is arbitrary scaled. Since the task is to localize in regards to the model, the scale needs to be solved for. For each correspondence an α also has to be solved. This result in $7 + n$ total unknown variables, where n is the amount of points used. The problem of finding a transformation which lets 3D points coincide with rays can be expressed

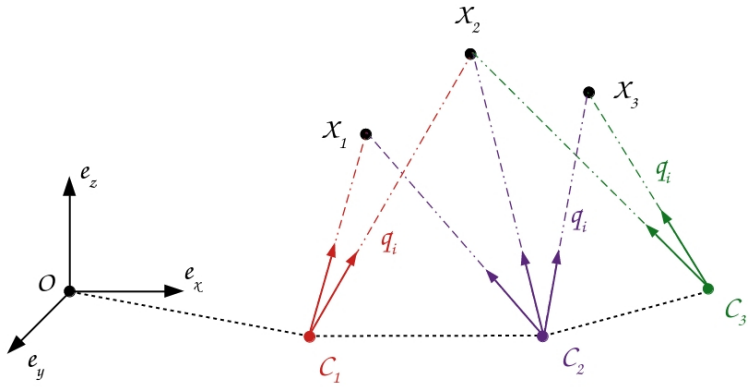

Figure 2.7: A graphical visualization of how the cameras in Figure 2.6 could be represented as a single generalized camera.

as minimizing the cost function

$$C(R,t,s) = \sum_{i=1}^{n} ||\bar{r}_i - \frac{1}{\alpha_i}(Rr_i + t - sq_i)||^2 \ . \tag{2.15}$$

We have one slight problem. For every correspondence added a new equation is introduced but so is a new unknown variable. Fortunately, it is possible to exploit linear dependence between unknowns and rewrite the equation in terms of only the rotation R. Equation (2.14) can be rewritten as

$$\underbrace{\begin{bmatrix} \bar{r}_1 & & & q_1 & -I \\ & \ddots & & \vdots & \vdots \\ & & \bar{r}_n & q_n & -I \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ s \\ t \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} R & & \\ & \ddots & \\ & & R \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} r_i \\ \vdots \\ r_n \end{bmatrix}}_{b} \tag{2.16}$$

$$\Leftrightarrow Ax = Wb \tag{2.17}$$

where x contain all linearly dependent variables to be eliminated. Manipulating Equation (2.17) it is possible to express x as

$$x = (A^T A)^{-1} A^T W b = \begin{bmatrix} U \\ S \\ V \end{bmatrix} Wb \tag{2.18}$$

where U,S,V are constant matrices constructed from $(A^T A)^{-1} A^T$. By expressing t, s, α in terms of U, S, V, W, b it is now possible to rewrite Equation (2.14) and (2.15) to only include the unknown rotation. There are now more equations than unknowns and it is possible to find a least squares solution. Also observe that the number of unknowns now is independent from the number of observations making it scalable in terms of number of points used by the solver. For a more detailed explanation, read [58].

2.5 Random Sample Consensus

Random Sample Consensus (RANSAC) [15] is a method for picking random data samples in a way that ensures an outlier free subset of data to a certain degree. In many optimization tasks, the data is not perfect. It contains noisy measurements and outliers. Optimizing for all points can then be challenging. Outliers can be far off from a correct measurement, contributing with large errors to the cost function, drowning the effect of errors from correct measurements. The RANSAC method is widely used to try and find a subset of data that only contains inliers.

Given a set of data points and a solver requiring s points to compute a solution, RANSAC randomly picks a subset of s data points N number of times. The idea is that in a set of points some will be outliers and some will be inliers. It is desired

to pick a subset of points which is outlier free, resulting in the computation of a correct solution. The number of times a subset has to be picked is described by

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)} \tag{2.19}$$

where N is the number of iteration, s is how many points are needed to find a solution, e is the ratio between outliers and all matched points and p represents the desired probability of finding the best model.
The outlier ratio e is hard to determine and scarcely available. It can therefor be necessary to update the distribution based on inlier results calculated during runtime. The process of calculating inliers is done every iteration of RANSAC to evaluate if the subset of points picked resulted in a good solution. To determine if a 3D point is an inlier, the point is reprojected using (2.3). Then the distance between the reprojected point and the image feature which the point is matched against is computed. If this distance is smaller that a pre-determined threshold, it is considered an inlier. This procedure is done for all points and will give the number of inliers for a given solution. The number of inliers is used to calculate a new outlier ratio and is then inserted back in (2.19) to calculate the number of iterations needed.

2.5.1 RANSAC with Td,d test

The Td,d test [36] is a small addition to the normal RANSAC procedure. In normal RANSAC each solution will be evaluated on all data points. This is computationally expensive and shown to be unnecessary. Instead, the Td,d test uses a very smaller subset d of data points for a first preliminary evaluation. If the preliminary evaluation suggests that a good solution has been found by all points in d being inliers, then all other data is also evaluated to determine if a new best solution has been found. The number of iterations needed can be formulated as picking $s + d$ inlier points as

$$N = \frac{\log(1-p)}{\log(1-(1-e)^{s+d})} \tag{2.20}$$

where d is the number of points to evaluate the solution on. It was shown in [36] that choosing $d << N$ is enough to confidently tell if a solution is correct or not.

3
Method

The task at hand is to create a localization pipeline using a 3D structure-based method with the properties of a sequential-based method. By using a 3D-point model and a query image sequence, multiple camera pose estimates are computed and evaluated. The pipeline is built with a modular structure in mind so that if desired, a part can be swapped without influencing the remaining modules. Given a query of sequential images and a model, a camera pose estimations are computed for every image in the sequence. This chapter will cover how the pipeline is constructed as shown in figure 3.1.

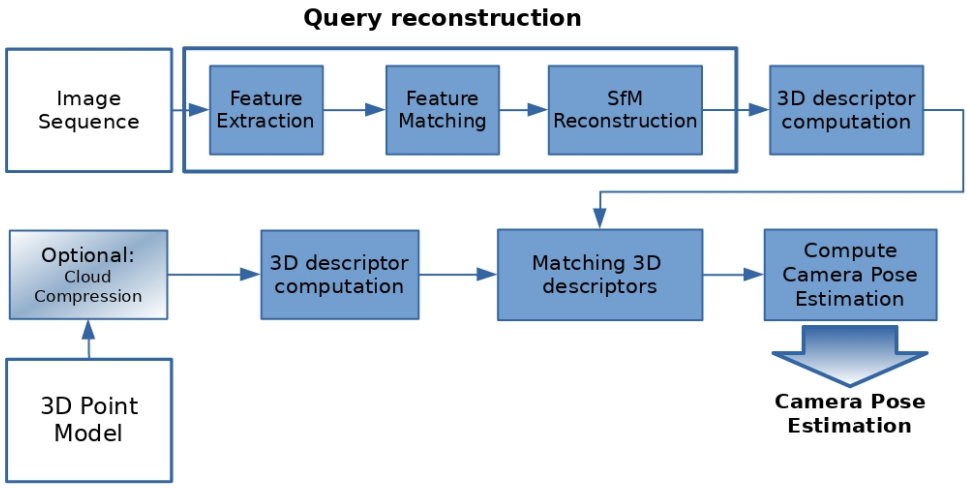

Figure 3.1: This is an overview of the pipeline. Given a sequence of images a query reconstruction is created. From the reconstruction, a 3D descriptor is computed and matched against the models 3D descriptors.

Query reconstruction is a collective term for the process of computing a 3D SfM reconstruction. Given a set of images, features are extracted from all images. All features are matched against each other and then computed into a Structure from Motion reconstruction. Feature extraction, feature 2D-3D matching and model generation is done with COLMAP [53]. Since COLMAP outputs descriptors for 2D features and not for the reconstructed 3D points, a descriptor for each 3D point is calculated by creating an elementwise mean descriptor from all feature descriptors mapped to each 3D point. The 3D descriptor is then matched with the precomputed model 3D descriptor where a ratio test is used for outlier rejection. 3D-3D

matching is done using a kd-tree search algorithm for nearest neighbor matching. Using the matched points, a camera pose is computed using the generalized camera solver $gDLS$ [58].

It is also possible to include compression in the pipeline. Either the pipeline is run from start to finish with an added step of compressing the model point cloud before the matching step or using previous query reconstruction to only re-run the descriptor matching and camera pose estimation steps in Figure 3.1 with a compressed point cloud for the model. No implementation is available for running compressed descriptors.
Below we will cover each module in detail starting with query reconstruction followed by computing 3D descriptors, matching 3D descriptors, camera pose estimation, compression of 3D model and the special case of using a sequence length of one.

3.1 Query Reconstruction

The query is reconstructed using COLMAP [53]. Given a sequence of images COLMAP extracts feature points and 128-dimensional descriptors that describes the respective point. COLMAP implements SIFT [33] to extract feature descriptors in each image. This process is represented as *Feature Extraction* in Figure 3.1. The data is then processed in *Feature Matching* by matching the descriptors between the images, illustrated in Figure 2.4. The final step of the reconstruction is computing the Structure from Motion model using the matched data. Using the matched features between images 3D points are triangulated, resulting in a 3D reconstruction and estimates of the cameras pose for each query image in the query-coordinate frame obtained from the local reconstruction. The query-coordinate frame will from here on be referred to as *query frame*. The resulting reconstruction is defined up to an arbitrary scaling factor in its own query frame. As a last step, all 2D features which are not assigned to a 3D point after the COLMAP reconstruction will be assigned to a 3D point on the image plane and transformed into the query model using the poses estimated from SfM. If only the points that were triangulated from SfM were to be used in the estimation of the generalized camera pose, it would result in to few rays and the solver would estimate significantly less accurate camera poses. As the solver utilizes ray origins and ray directions the depth of a point is not needed. However, the COLMAP reconstruction is still necessary to determine all camera poses in the query frame.

3.2 Compute 3D Descriptors

The resulting data from the reconstructions provides 3D points in query coordinate frame. The model and query are defined in their own frames from here on referred to as world and query coordinate frame. These 3D points are only described in space (X,Y,Z) and do not have a descriptor assigned to them. This is why the 3D descriptor $\bar{d}$ is computed, by taking the mean of the feature point descriptors d_i that have been matched to the 3D point. The resulting descriptor $\bar{d} = \frac{1}{n}\sum_{i=0}^{n} d_i$

is assigned to the 3D point. The mean is used since the corresponding feature descriptors should be similar due to matching. This results in a 128-long vector describing the 3D-point. This is done for the query and model, where the model is computed beforehand.

3.3 Matching 3D Descriptors

With 3D descriptors created for both query and model the task is to find matches between these two models. We use the existing library FLANN, Fast Library for Approximate Nearest Neighbors [35]. FLANN is used to find the most similar model descriptor to all query descriptors, however due to the amount and dimensionality of descriptors, a kd-tree search is implemented to reduce computation time. FLANN outputs the indexes that correspond to the most similar descriptors in the model and the distance between the descriptors, which is the vector norm of the difference between the query $\bar{d}_i$ and model d_{NN} descriptors. Since there is a possibility that some matches are outliers Lowe's ratio test [33] is implemented as described in Equation (2.12).

3.3.1 Implementation

The FLANN library was chosen as it is highly customizable, in our method a kd-tree search is used. The default settings are used, with the exceptions of trees changed from 1 to 10 and branches from 32 to 100. FLANN is set to search for the two nearest neighbors. The tolerance used in Lowe's ratio test is 0.75.

3.4 Camera Pose Estimation

Using the matched data and the camera poses from the query reconstruction, the final step is to compute the camera pose estimation in world coordinate frame. For this we use primarily the gDLS (generalized Direct Least Squares) solver [58] and in the unique case when the image sequence is one image, the DlsPnp (Direct least squares Perspective n Point) [21] solver is used. The solvers implementations in Theia SfM library [57] are used. To handle outliers in the matches RANSAC with Td,d test [37] is implemented.

3.4.1 Implementation

To use the solvers the matches needs to be converted into ray origins and ray directions. This is done by taking the difference between the point and camera center

$$q_{i,j} = p_i - C_j \tag{3.1}$$

where $q_{i,j}$ is the ray origin, p_i is the 3D point coordinate and C_j is the camera center/ray origin, the light ray is then normalized. At least 4 light rays and 4 ray origins are necessary to solve with both gDLS and DlsPnp. In the description for the solver in the Theia SfM library [57] it says "*Theoretically, up to 27 solutions*

may be returned, but in practice only 4 real solutions arise". Each solution contains a scale s, rotation R and translation t representing a similarity transform between query frame and world frame. Since there are outliers in the matching RANSAC with Td,d test [37] is implemented to find the best solution. For a set amount of iterations RANSAC samples five light rays and their ray origins where four of these light rays and ray origins are used to compute solutions using the gDLS solver. The resulting solutions are then applied to the fifth rays corresponding 3D point, the point is transformed and reprojected down onto the image. If the reprojection error is less then a set tolerance (we used 10 pixels) the light ray and ray origin is considered good enough for the solution to be evaluated on all rays. By transforming the query point cloud and reprojecting all points we evaluate how many inliers there are. The solution that produces the most inliers after all RANSAC iterations will be presented as *best_solution*. The choice of using RANSAC was due to the amount of light rays and ray origins, to decreased the computation time for each iteration RANSAC with Td,d test is used. However since the amount of sampled light rays increase, based on Equation 2.19 the amount of iterations necessary for a given probability of picking an outlier free subset is also increased. After the iterations, the *best_solution* defined as (s, R, t) is then applied to the camera pose estimates (R_q, t_q) in query frame, resulting in the desired output (R_{est}, t_{est}) where $R_{est} = R_q R$ and $t_{est} = R_{est} t + \frac{1}{s} t_q$. The code used for computing the camera poses are presented as pseudo-code in Algorithm 1.

Algorithm 1 Solver with RANSAC

```
Convert matches into light rays and ray origins
for Amount of RANSAC iterations do
    Sample 5 random query rays and corresponding model 3D points
    Use rays 1-4 in gDLS
    for all solutions do
        Apply solution to fifth point
        Compute reprojection error of transformed fifth point
        if reprojection error ≤ threshold then
            inliers = 0
            for all query points do
                Apply solution
                Compute reprojection error
                if reprojection error ≤ threshold then
                    inliers = inliers + 1
            if inliers > best_inliers then
                update best_inliers
                update best_solution
Apply best_solution to query
```

3.5 Compressing 3D Point Cloud

When compressing the 3D model the total amount of 3D points are reduced. We specify the compression as how many points are left compared to the original amount. For example if there originally are one hundred points and after compression only ten are left, this would be a compression rate of ten percent, thus reducing the necessary static storage for the model into 1/10 of the original size. It would be preferable if this percentage could be specified an input to the compression algorithm but it introduces some unwanted problems. When compressing data the goal is to preserve as much information as possible in a small set of all data. We could simply pick the first $x\%$ of points based on how many images observe the point as a measure. However, this could lead to the points being picked all ending up in the same region of the model. This could happen if there are more images depicting the same scene or if one scene contain more structural variance in the model dataset. This would result in a compressed model with partial or no coverage for some areas. We prevent this from happening by instead specifying a minimum amount of points that all images must see, similar to the method by Li et al. [31]. First, points are sorted and picked after how many images they are present in. As points are being picked and an image reaches the minimum number of points required, it will no longer count toward the number of images a point is present in and all points are resorted. This continues until all images can see the minimum amount of points or all points have been evaluated since it's possible that some images view less then the minimum requested amount of points.

3.6 Special case: One Image

When using a single image COLMAP can not create a query reconstruction. Instead, a 3D point is created for every image feature. This point is placed in the image plane, a z-distance of one, and the same x and y pixel value as the image feature. It is then transformed with the inverse calibration matrix to be placed in the query coordinate system. Using one images also results in all rays having the same ray origin. gDLS can not find a solution if all rays have the same origin. Instead DlsPnP is used.

4
Results

In this chapter we will present our localization results for different sequence lengths as as well as how overlapping images and compressed data affect localization using generalized cameras.

4.1 Datasets

To verify that our pipeline achieves significant results we have evaluated our pipeline on the Cambridge Landmarks dataset [29] as it contains scenes of different sizes and is widely used in the visual localization community. All scenes in the dataset were recorded roughly under the same weather conditions.

4.1.1 Cambridge Landmarks St Mary´s Church

In the St Mary's Church dataset all images come in a number of image sequences. It contains a prebuilt model reconstructed from images intended to be used for training of neural networks. It also contains test images that have not been used to create the model. In total there are around 1500 training images and 500 test images to evaluate on. As St Mary's Church is a small dataset of a structure with well defined features it is used as a first benchmark to determine if a localization algorithm is able to localize images in a relatively easy scene.

4.1.2 Cambrige Landmarks Street

The Street scene shows King's Parade street in Cambridge. It contains roughly 3000 training images which the included model is reconstructed from. There are also nearly 3000 test images. The training images are divided into four sequences; west, east, north and south. In each sequence the camera is only pointed in the direction of the sequence name. The test images are instead taken by walking up and down King's Parade constantly rotating. Street poses a more challenging localization problem as it features many repeated features over a larger area and a greater variety of depth at which structure are located.

4.2 Evaluation

We evaluate our pipeline implementation on the above described datasets. The results will be presented in two ways. First to evaluate the impact of different sequence

length our results from the St Mary's and Street will be presented in the same way as [51] to observe the trends of increasing sequence length. Our results will also be presented as median positional and orientation error of the pose estimation to compare results against other camera localization methods [52]. More specifically we will compare against some deep learning approaches, PoseNet with geometric loss function [28], MapNet [6], AnchorNet [47]; and some structure based approaches, DSAC++ [5], and ActiveSearch [50] as well as the image retrieval method DenseVLAD [61] as this will be a good selection of methods highlighting localization results from different approaches. Additionally to this we will evaluate the impact of using a compressed 3D point model with the initial assumption that by using more images the point cloud can be compressed to a fraction of its initial size without significantly impacting localization results.

In [51] each image is placed in bins depending how accurately it was localized in positional error computed by the Euclidean distance and the orientation error (Xm, Y^o). There are three bins with thresholds $(0.25m, 2^o)$, $(0.5m, 5^o)$ and $(5m, 10^o)$ labeled fine-, medium- and coarse-precision. An image is placed in all or none of the bins depending on how accurately it was localized. We will measure pose accuracy as the absolute orientation and position error between the estimation and the ground truth. The orientation error $|\alpha|$ is measured by standard practice [19] and is computed as $2\cos|\alpha| = \text{trace}(R_{\text{gt}}^{-1}R_{\text{est}})$ where $|\alpha|$ is the minimum rotation angle to align the estimated rotation matrix R_{est} to the ground truth rotation R_{gt}. The position error is measured as the Euclidean distance between the ground truth camera origin c_{gt} and c_{est} the estimated camera origin, $||c_{\text{est}} - c_{\text{gt}}||_2$.

The localization results for our pipeline on the Street and St Mary's dataset shown in bins can be seen in Table 4.1 where different sequence lengths are evaluated on the two datasets. Of note is that the St Mary's dataset contains three sequences of images with two of them consisting of less than a hundred images. When localizing images with sequence lengths of eighty five and above, the entire sequence was then localized at the same time. There is a clear trend between sequence lengths where results improve with each step up in sequence length until a certain point where results start to fluctuate up and down. For Street this happens for longer sequence lengths than St Mary's. For both datasets the use of sequences vastly outperforms the results from the single image solver. However for some sequence lengths the single image case performs better. This is likely the result of some sequences not being reconstructed and therefor the images in the sequence are not localized.

In Table 4.2 the results are presented as median position and orientation error to be compared with state of the art localization results presented in [52]. We manage to outperform the best Street localization result in position while our pipeline is only surpassed by ActiveSearch in orientation error for a sequence length of eighty. This is presented in Table 4.6 and visualized in Figure 4.1 and 4.2.

An advantage of localizing sequences of images is that we can overlap sequences with each other thus the images exist in multiple queries. This increases the likelihood of the image existing in a reconstruction and therefor being localized. When presented with two or more poses we take the one with the most inliers, computed by pro-

	Street	St Mary's
m	.25/.50/5.0	.25/.50/5.0
deg	2/5/10	2/5/10
1	12.8/23.4/37.9	0/3.9/97.8
5	2.2/5.7/25.2	5.5/15.8/72.3
10	4.3/9.9/33.7	9.8/27.1/89.8
15	4.0/9.0/26.8	8.1/23.1/89.3
20	2.8/8.6/33.5	4.9/18.6/88.7
25	4.1/11.7/34.5	17.3/34.5/88.7
30	4.3/12.8/38.2	15.0/38.2/94.5
35	7.4/17.4/42.0	16.2/46.2/90.6
40	5.2/11.3/20.2	23.1/54.5/98.3
45	11.3/26.1/56.3	21.2/48.5/99.2
50	10.8/23.6/43.0	27.8/60.0/99.4
55	6.6/19.1/46.5	**35.3/69.4/99.6**
60	8.7/21.7/51.7	25.4/59.2/91.5
65	5.3/17.1/46.5	25.2/62.2/99.4
70	5.6/19.8/52.9	24.2/62.8/99.1
75	**13.1**/29.5/62.0	28.9/69.0/99.2
80	11.0/**30.3/67.1**	28.4/64.5/99.1
85	11.3/29.0/56.7	28.9/63.3/99.4
90	11.1/21.9/48.2	20.5/54.3/95.3
95	12.6/28.6/60.5	20.7/61.3/99.2
100	10.7/23.9/52.4	22.4/63.0/98.9

Table 4.1: This table shows localization results on the Street and St Mary's datasets for different sequence lengths where all images are assigned to bins depending on how accurately they were localized. Presented is the percentage of images clearing the threshold for each bin. In bold are the best localization results for Street and St Mary's dataset respectively.

	Street	St Mary's
	Median position [m]/ orientation [^{o}] error	Median position [m]/ orientation [^{o}] error
1	17.690/68.290	0.82/1.05
5	30.249/54.616	2.11/2.07
10	15.911/38.968	0.91/1.44
15	16.190/41.723	1.03/1.43
20	19.448/38.515	1.19/1.75
25	6.601/11.711	0.67/1.22
30	11.087/14.759	0.66/0.99
35	4.521/8.274	0.54/1.16
40	3.443/12.327	0.43/0.80
45	3.077/6.326	0.51/0.84
50	3.246/6.570	0.40/0.71
55	1.956/3.302	**0.35/0.60**
60	1.459/4.147	0.40/0.76
65	1.945/4.068	0.39/0.82
70	1.586/4.983	0.40/0.77
75	0.968/**2.066**	0.36/0.69
80	**0.82**/2.458	0.39/0.81
85	0.853/2.530	0.39/0.71
90	2.856/5.467	0.46/0.89
95	1.096/3.386	0.40/1.10
100	1.486/3.390	0.40/0.84

Table 4.2: Localization results from our pipeline on Street and St Mary's datasets shown as median position and orientation error.

jecting the matched points onto the query image. In Table 4.4 we see that by using overlapped sequences we improve our previous best results by over ten percentage points in all bins compared to not using overlapped sequences. The best median position and orientation error is also halved in Table 4.5.

When comparing our pipeline on the St Mary's dataset, Figure 4.1, our method preforms great in comparison to PoseNet, MapNet and DenseVLAD, but are outperformed by ActiveSearch and DSAC++.
As for the Street dataset, Figure 4.2, our method outperforms both AnchorNet and DenseVLAD, using sequences with no overlap our method reaches the same level of accuracy in position error as ActiveSearch but is outperformed in orientation error. Using overlapping sequences our method reaches an equal oritentional accuracy as that of ActiveSearch and outperforms all methods in positional accuracy.

In Table 4.3 we have also evaluated if our localization results are correlating with the reconstruction errors from COLMAP. If these are compared to the localization

	Street	St Mary's
	Median position error [m]	Median position error [m]
5	15.8	**1.3**
10	14.3	1.8
15	65.7	1.6
20	29.9	2.5
25	12.4	2.1
30	20.3	1.8
35	13.8	1.8
40	19.2	1.8
45	6.7	3.4
50	16.8	2.1
55	9.1	1.9
60	6.4	1.9
65	16.2	2.4
70	6.5	2.5
75	**3.6**	2.2
80	6.7	2.7
85	8.2	3.6
90	11.3	5.0
95	9.3	3.9
100	13.3	3.1

Table 4.3: Pipeline reconstructions compared to ground truth poses for reconstruction drift evaluation. The camera with the largest pose drift is selected for each sequence. Reconstruction drift is then determined as the median of these values for each sequence length.

	Street - Overlap	Street
m	.25/.50/5.0	.25/.50/5.0
deg	2/5/10	2/5/10
10	9.1/22.61/54.47	4.3/9.9/33.7
20	8.52/23.20/60.90	2.8/8.6/33.5
30	10.26/25.08/61.31	4.3/12.8/38.2
40	14.27/31.95/64.49	5.2/11.3/20.2
50	19.30/39.28/73.25	10.8/23.6/43.0
60	22.20/44.13/75.13	8.7/21.7/51.7
70	23.30/46.9/82.83	5.6/19.8/52.9
80	21.35/46.73/80.6	**11.0/30.3/67.1**
90	24.50/49.74/85.63	11.1/21.9/48.2
100	**29.73/54.36/87.92**	10.7/23.9/52.4

Table 4.4: This table shows localization results for the Street dataset where a sequence overlaps the previous sequence with n/2 images if the sequence length is n.

results we see that reconstructions for Street have reconstruction errors roughly increasing and decreasing in a similar pattern as the localization errors for different sequence lengths. This does however not seem to be the case for St Mary's Church as the smallest reconstruction errors are for shorter sequence lengths in comparison the sequence lengths achieving the best localization results. Except for the sequence length five, the reconstruction errors are similar until the start to grow for longer sequence lengths. The observed trends indicate that reconstruction errors is a factor that can't be ignored if one want to achieve good localization results for larger datasets.

4.2.1 Results Model compression

When testing our pipeline on compressed 3d point clouds as described in Section 3.5 we never managed to get results even in the vicinity of our other localization results. Our tests were performed on the Street dataset which 3d point cloud had been compressed to 50%. To try and get good matches with the compressed point cloud different values between 0.3 and 0.8 for the threshold in Lowe's ratio test was tested but none of the values tried managed to achieve presentable localization results.

	Street - Overlap	Street
	Median position [m]/ orientation [^{o}] error	Median position [m]/ orientation [^{o}] error
10	2.750/3.210	15.911/38.968
20	1.730/2.810	19.448/38.515
30	1.605/3.042	11.087/14.759
40	1.163/2.312	3.443/12.327
50	0.726/1.794	3.246/6.570
60	0.620/1.419	1.459/4.147
70	0.541/1.239	1.586/4.983
80	0.555/1.382	**0.82/2.458**
90	0.500/1.164	2.856/5.467
100	**0.435/0.983**	1.486/3.4

Table 4.5: Same localization results as Table 4.4 but shown as median position/orientation error.

	Median error			
	Street		St Mary's	
	position [m]	orientation [^{o}]	position [m]	orientation [^{o}]
PoseNet	20.3	25.5	1.57	3.32
MapNet	-	-	2.00	4.53
DenseVLAD	5.16	23.5	2.31	8.00
ActiveSearch	0.85	0.8	0.19	0.54
DSAC++	-	-	0.13	0.4
AnchorNet	7.86	24.2	1.04	2.69
Our method	0.82	2.46	0.35	0.60
Our method w. overlap	0.435	0.983	-	-

Table 4.6: This table shows the median errors of the state of the art localization pipelines, PoseNet [28], MapNet [6], DenseVLAD [61], ActiveSearch [50], DSAC++ [5], AnchorNet [47], DenseVLAD [61], that we compare against for the Cambridge Landmarks data sets

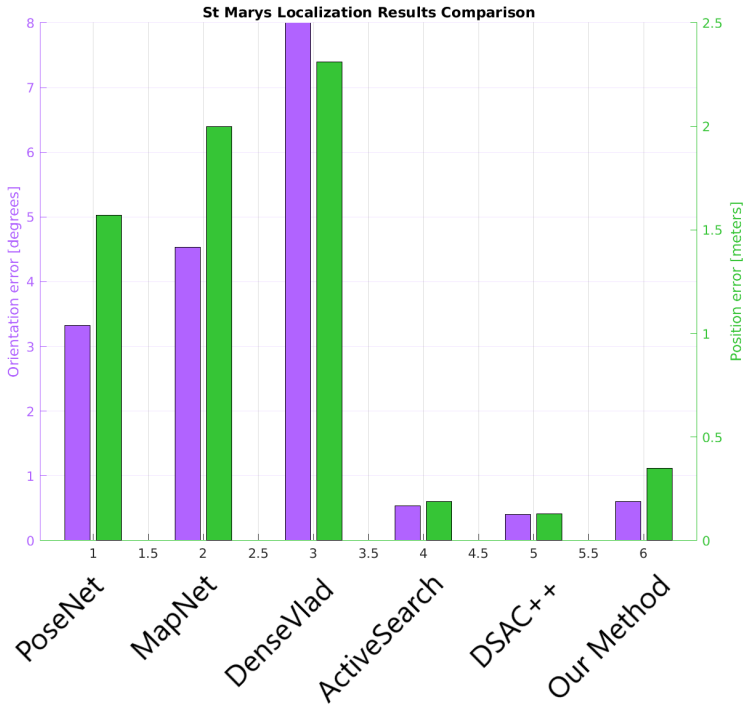

Figure 4.1: Comparison on the dataset Cambridge Landmarks: St Mary's Church dataset [29] between our method and PoseNet [28], MapNet [6], DenseVLAD [61], ActiveSearch [50], DSAC++ [5].

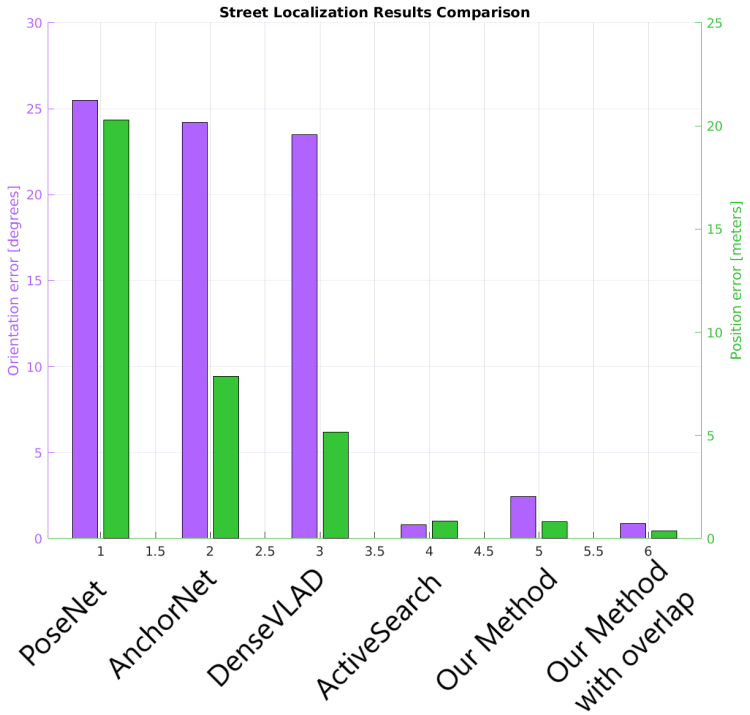

Figure 4.2: Comparison on the dataset Cambridge Landmarks: Street dataset [29] between our method and AnchorNet [47], DenseVLAD [61] and ActiveSearch [50].

5
Conclusion

A localization pipeline has been implemented that utilizes a generalized camera model for image localization. As the pipeline makes use of generalized cameras it can localize an entire image sequence at once which has become highly relevant in many modern localization applications that make use of video streams for capturing visual information. It achieves significant results on the Street dataset by localizing images more accurately than state of the art localization pipelines when using overlapping sequences. On the St Mary's dataset it perform as well but not better than other pipelines when using non overlapping sequences. This is likely due to that the dataset covers only one primary structure. So when using a short image sequence some queries will not be reconstructed and thus result in a decresed accuracy. By testing different sequence lengths it is clear that sequence lengths of fifty or more is desirable for accurate localization for this dataset. On the Street dataset a sequence length of eighty images was used to reach the peak accuracy. Using overlapping image sequences increased our pipelines accuracy and more accurate localization could be achieved for shorter sequences. This is likely due to the images existing in additional subsets and increases the likelihood that the image exists in a complete reconstruction and thus computed a camera pose estimate. When testing the pipeline in compressed datasets no presentable results were gathered.

6
Future Work

In this thesis we have presented some results for camera localization with generalized cameras. The developed pipeline has a lot of potential for improvements as it is only a first implementation where the amount of optimization has been minimal. The gDLS solver is implemented in a way to maximize the likelihood of good results at the cost of long run times. We believe that the run time could be significantly shorter with a relatively small amount of adjustments and changes.
The gDLS solver solutions did not receive any bundle adjustment which should improve results to some degree.
We hypothesized that using a generalized camera could help in localization with compressed databases but we did not achieve any presentable results when testing our pipeline on a compressed model of the Street dataset. This could be explored further to narrow down the issue. It is possible that the fault is in our implementation of selecting 3D points or that Street simply is a challenging dataset to achieve good results using model compression.
As applications for localization using image sequences usually require real-time performance from the localization pipeline, the next step would be to create a pipeline capable of real-time localization and record benchmarks for comparison to other localization methods. We believe this could be possible by incrementally including new images and discarding old ones from a single query reconstruction. By doing so the SfM calculation would be significantly faster and previously good matches between the query and reference model can be utilized when calculating the new pose.

Bibliography

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79. IEEE, 9 2009.

[2] C. Aggarwal, A. Hinneburg, and D. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In J. Van den Bussche and V. Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 6 2008.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 1975.

[5] E. Brachmann and C. Rother. Learning Less is More - 6D Camera Localization via 3D Surface Regression. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[6] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. Geometry-Aware Learning of Maps for Camera Localization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[7] S. Cao and N. Snavely. Graph-Based Discriminative Learning for Location Recognition. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 700–707, 2013.

[8] S. Cao and N. Snavely. Minimal scene descriptions from structure from motion models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014.

[9] G. Carrera, A. Angeli, and A. J. Davison. SLAM-based automatic extrinsic calibration of a multi-camera rig. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2011.

[10] W.-T. Chu, X.-Y. Zheng, and D.-S. Ding. Image2Weather: A Large-Scale Image Dataset for Weather Property Estimation. In *2016 IEEE Second International Conference on Multimedia Big Data (BigMM)*, pages 137–144, 5 2016.

[11] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 5 2007.

[12] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

[13] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 6 2007.

[14] M. Feng, S. Hu, M. H. Ang, and G. H. Lee. 2D3D-MatchNet: Learning to Match Keypoints Across 2D Image and 3D Point Cloud. *2019 International Conference on Robotics and Automation (ICRA)*, pages 4790–4796, 2019.

[15] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 2002.

[16] J.-M. Frahm, K. Köser, and R. Koch. Pose Estimation for Multi-camera Systems. In H. H. Rasmussen, Carl Edward B{\"u}lthoff, B. Sch{\"o}lkopf, and M. A. Giese, editors, *Pattern Recognition*, volume 3175, pages 286–293, Berlin, 5 2004. Springer Berlin Heidelberg.

[17] M. Grossberg and S. Nayar. A general imaging model and a method for finding its parameters. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, pages 108–115, 2001.

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

[19] R. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation Averaging. *International Journal of Computer Vision*, 103(3):267–305, 7 2013.

[20] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2 edition, 2004.

[21] J. A. Hesch and S. I. Roumeliotis. A Direct Least-Squares (DLS) method for PnP. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[22] K. L. Ho and P. Newman. Loop closure detection in SLAM by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54(9):740–749, 9 2006.

[23] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997.

[24] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2599–2606, 2009.

[25] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[26] M. Kaess and F. Dellaert. Visual SLAM with a Multi-Camera Rig. *Technology*, 2006.

[27] F. Kahl. Lecture 3: Camera Calibration, DLT, SVD, 2018.

[28] A. Kendall and R. Cipolla. Geometric Loss Functions for Camera Pose Regression With Deep Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2017.

[29] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946. IEEE, 12 2015.

[30] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3D point clouds. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. ECCV, 2012.

[31] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010.

[32] L. Liu, H. Li, and Y. Dai. Efficient Global 2D-3D Matching for Camera Localization in a Large-Scale 3D Map. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[33] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 11 2004.

[34] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart. Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization. In *Robotics: Science and Systems XI*. Robotics: Science and Systems Foundation, 7 2015.

[35] Marius Muja and David G. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 36(11):2227–2240, 2014.

[36] J. Matas and O. Chum. Randomized RANSAC with Td,d test. In *Image and Vision Computing*, 2004.

[37] J. r. Matas and O. r. Chum. Randomized RANSAC. In *Proceedings of the CVWW'02*, pages 49–58, 2002.

[38] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-DOF localization on mobile devices. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[39] M. Milford, S. Lowry, N. Sunderhauf, S. Shirazi, E. Pepperell, B. Upcroft, C. Shen, G. Lin, F. Liu, C. Cadena, and I. Reid. Sequence searching with deep-learnt depth for condition- and viewpoint-invariant route-based place recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 18–25. IEEE, 6 2015.

[40] M. J. Milford and G. F. Wyeth. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE International Conference on Robotics and Automation*, pages 1643–1649. IEEE, 5 2012.

[41] P. Moulon, P. Monasse, and R. Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[42] R. Paucher and M. Turk. Location-based augmented reality on mobile phones. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, 2010.

[43] R. Pless. Using many cameras as one. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages 587–93. IEEE Comput. Soc, 2003.

[44] D. Robertson and R. Cipolla. An Image-Based System for Urban Navigation. In A. Hoppe, S. Barman, and T. Ellis, editors, *Proceedings of the British Machine Vision Conference*, pages 1–84. BMVA Press, 2004.

[45] A. Roshan Zamir and M. Shah. Accurate Image Localization Based on Google Maps Street View. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, pages 255–268, Berlin, 2010. Springer Berlin Heidelberg.

[46] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571. IEEE, 11 2011.

[47] S. Saha, G. Varma, and C. V. Jawahar. Improved Visual Relocalization by Discovering Anchor Points. In *British Machine Vision Conference 2018*, page 164, 2018.

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

[49] T. Sattler, M. Havlena, K. Schindler, and M. Pollefeys. Large-Scale Location Recognition and the Geometric Burstiness Problem. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1582–1590. IEEE, 6 2016.

[50] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[51] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. *Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions*. CVPR 2018, 2018.

[52] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. Understanding the Limitations of CNN-Based Absolute Camera Pose Regression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2019.

[53] J. L. Schonberger and J.-M. Frahm. Structure-from-Motion Revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[54] X. Shen, Z. L. Lin, J. Brandt, S. Avidan, and Y. Wu. Object retrieval and localization with spatially-constrained similarity measure and k-NN re-ranking. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3013–3020, 2012.

[55] Sivic and Zisserman. Video Google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003.

[56] N. Sünderhauf, F. Dayoub, S. Shirazi, B. Upcroft, and M. Milford. On the Performance of ConvNet Features for Place Recognition. *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 1 2015.

[57] C. Sweeney. Theia Multiview Geometry Library: Tutorial & Reference. http://theia-sfm.org.

[58] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. gDLS: A Scalable Solution to the Generalized Pose and Scale Problem. In *Computer Vision – ECCV 2014*, pages 16–31, 2014.

[59] C. Sweeney, V. Fragoso, T. Hollerer, and M. Turk. Large scale SfM with the distributed camera model. In *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, 2016.

[60] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. InLoc: Indoor Visual Localization with Dense Matching and View Synthesis. *Proc. CVPR*, 3 2018.

[61] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1808–1817, 10 2015.

[62] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. *Workshop on Vision Algorithms*, 1999.

[63] S. Urban and S. Hinz. MultiCol-SLAM - A Modular Real-Time Multi-Camera SLAM System. *arXiv preprint arXiv:1610.07336*, 2016.

[64] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using LSTMs for structured feature correlation. *ICCV*, 2017.

[65] Z. Xin, X. Cui, J. Zhang, Y. Yang, and Y. Wang. Visual place recognition with CNNs: From global to partial. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6. IEEE, 11 2017.

[66] B. Zeisl, T. Sattler, and M. Pollefeys. Camera pose voting for large-scale image-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[67] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi. Revisiting the PnP problem: A fast, general and optimal solution. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.