APPEARANCE-BASED VEHICLE LOCALIZATION ACROSS SEASONS IN A METRIC MAP

A Dissertation
Presented to
The Academic Faculty

by

Christopher A. Beall

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2016

APPEARANCE-BASED VEHICLE LOCALIZATION ACROSS SEASONS IN A METRIC MAP

Approved by:

Professor Frank Dellaert, Advisor
School of Interactive Computing
Georgia Institute of Technology

Professor Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Anthony Yezzi, Co-Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Henrik Christensen
School of Interactive Computing
Georgia Institute of Technology

Professor James M. Rehg
School of Interactive Computing
Georgia Institute of Technology

Professor Gabe Sibley
Department of Computer Science
University of Colorado Boulder

Date Approved: June 17, 2016

ACKNOWLEDGEMENTS

I thank my advisor Dr. Frank Dellaert for his guidance and support, without which this work would not have been possible. I also want to thank my committee for their valuable feedback during the final stages of this work.

Additionally, I wish to acknowledge my labmates who have played an important role in shaping the graduate school experience. I appreciate the many hours of fruitful discussion, and the opportunities to collaborate with all of you on interesting projects. Special thanks go to Richard, Andrew, and Luca who volunteered their own time to help me with data collection by driving Frank's camera-equipped car around campus.

Lastly, bust most importantly, I thank my family for their unconditional love, support, and patience.

Contents

List of Tables

List of Figures

SUMMARY

Great strides have been made in recent years in developing the necessary technologies to make autonomous cars a reality. However, a number of challenges remain, a major one being that of accurate vehicle localization.

This thesis presents a vision-only approach to the outdoor localization problem. The system provides for real-time, metric localization of a moving camera (on a vehicle) in a pre-built 3D map, which is inherently robust with respect to appearance changes. This is achieved by utilizing a novel spatio-temporal map (STM) representation which is built up from multiple drives worth of stereo camera data, as well as a localization algorithm which efficiently retrieves landmarks from the STM to perform appearance-based localization in real-time. The STM encodes the landmark visibility structure of the datasets which were captured to build the map, as well as landmark descriptors and observation times. This visibility structure and meta-data are then exploited for efficient localization. In addition, by splitting the STM up into a number of submaps, computational tractability is ensured during the map-building phase, as well as during localization. Experiments on real data validate that the presented method works better than conventional approaches which operate in a map built of a single dataset.

Chapter I

INTRODUCTION

Robust localization capability in changing environments is a prerequisite for any autonomous vehicle to operate in the long term. Using vision alone, this requirement proves to be quite challenging, as varying lighting and environmental conditions can drastically alter the appearance of the scene. Many existing algorithms fail when this occurs, as they typically assume a static scene, with a single representation of the world stored in the map. Localization using vision alone is attractive considering its very low cost compared to other sensor modalities. GPS is a popular solution in the context of outdoor vehicle localization and route guidance, but there are many scenarios where this can fail due to multi-path issues, drop-out, and drift, especially in urban environments. An example of an erroneous GPS trace is shown in Figure 1.

To date, a number of unsolved problems remain to make vision-based localization a reality. Most systems fall short in the face of changing scene appearance caused by differences in lighting, seasonal variation, foliage changes, weather, etc. This has become a very active area of research in recent years, and through some valiant efforts some progress has been made in addressing some of these challenges. For example, representing each place as a different experience in a topologically connected map appears to be a particularly promising

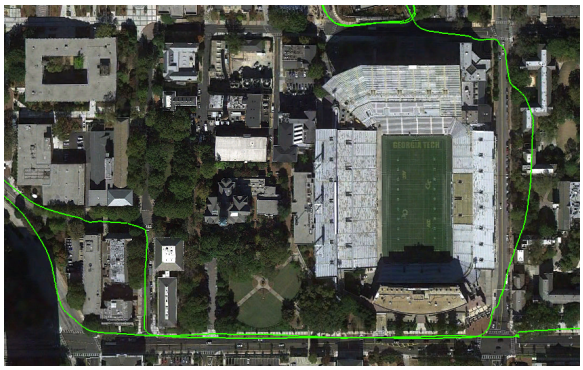


Figure 1: GPS-INS trajectory superimposed on Google Earth imagery. Severe GPS drift due to multi-path issues can be observed to the east of the stadium.

approach [21, 20], but this technique makes exact metric localization difficult as the query images are localized in several distinct visual odometry tracks. Muehlfellner et al. aimed to solve the problem by building a map containing many different datasets [62]. Their focus is on map building and the evaluation of different scoring functions to decide which landmarks to admit to the map, all in order to keep the size of the map relatively small. Experiments are limited to a parking lot and a simple loop on city roads.

In this thesis I present a methodology for real-time, metric localization of a moving camera in a pre-built 3D map, which is inherently robust with respect to appearance changes. This is achieved by utilizing a novel spatio-temporal map (STM) representation which is built up from multiple drives worth of data, as well as a snapshot recognition module (SRM), which efficiently retrieves landmarks from the STM to perform appearance-based localization in real-time. The map encodes the visibility structure of the datasets which were captured to build the map, and this information is exploited for efficient localization. The approach is validated with experiments in a map which has many intersections, and is also an order of magnitude larger than those in [62].

1.1 Thesis Statement

The spatio-temporal map and snapshot recognition modules together provide real-time, high-quality metric localization in changing environments.

1.2 Claims

In particular, I make the following three claims to support this thesis:

- **High-quality** - Localization results of the SRM have high accuracy, which is crucial for autonomous driving. A highly accurate 3D map of sparse landmarks from multiple data sequences is effective as the basis for high-quality localization. Furthermore, a highly accurate stereo visual odometry system forms the backbone of both the map-building and localization modules.

- **Efficient** - The SRM-STM localization algorithm provides localization in real-time, which is a critical requirement for autonomous driving.

- **Robust to change** - The SRM-STM localization system is robust to change because it utilizes a spatio-temporal map. By storing landmarks from multiple datasets collected at different times, it enables more robust localization than a single-dataset map in the presence of appearance changes.

1.3 Contributions

I make the following contributions as part of this thesis:

- I developed a highly accurate, real-time stereo visual odometry pipeline which is used for map building, as well as localization with the SRM.
- I developed the spatio-temporal map (STM) representation and methodology to build this map. The STM combines data from multiple seasons into a monolithic metric map, which makes localization in different conditions more robust.
- I developed the Submap-STM, which extends the STM and addresses its scalability limitations, both in run-time and storage space (memory).
- I developed a real-time visibility-based localization module for localization with the STM or Submap-STM.

1.4 Overview

The remainder of this document is organized as follows. In Chapter 2 I provide an overview of the relevant literature covering topics ranging from stereo visual odometry, to loop closure detection, smoothing and mapping, and finally localization. In Chapter 3 I present a real-time, and highly accurate visual odometry algorithm, which is an essential ingredient for the construction of the spatio-temporal map, and also for localization with the SRM-STM algorithm. The spatio-temporal map representation is introduced in Chapter 4, along with the algorithm to construct it from raw image data and GPS. In this chapter I also introduce an improved version—the Submap-STM—which addresses computational scalability limitations that exist with the monolithic STM. In Chapter 5 I describe the SRM-STM and Submap-SRM-STM localization algorithms. Final remarks are found in Chapter 6.

Chapter II

RELATED WORK

The literature survey that follows is divided into four parts. I begin with a brief overview of Visual Odometry (VO), which is a basic building block of the map building, as well as localization algorithms. This is followed by loop closing in Section 2.2. In Section 2.3 I review smoothing and mapping, and localization is presented in Section 2.4.

2.1 Visual Odometry

Camera egomotion can be computed from time-ordered image sequences, both monocular and stereo. This technique is referred to as visual odometry in the literature. In the monocular case this can be done only up to some unknown scale factor [38]. This scale ambiguity does not exist when a stereo camera is employed. In fact, results have been obtained with less than 1% error over many kilometers [66, 35, 45]. Several excellent survey/tutorial papers exist on this topic [79, 33]. I will give a brief overview of the stereo case here.

Stereo visual odometry algorithms take as input pairs of stereo rectified images obtained from the calibrated stereo rig. The process iterates as follows: In each frame, image features are extracted and stereo correspondences are established. A variety of feature detectors are suitable for this task, including those which require patch-based matching [37, 81], as well as the more recent descriptor-based detectors, such as the scale-invariant feature transform (SIFT) [56] or speeded up robust features (SURF) [8]. Descriptor-based detectors have nice scale- and rotation-invariance properties, and as such are also more suitable for the localization task described in the following sections.

Next, these features are matched into the next consecutive stereo frame, forming a set of temporal putatives. A 3-point algorithm is employed within a random sample consensus (RANSAC) framework to recover the incremental 6 degree of freedom (DOF) camera pose [38, 31]. RANSAC has become the standard tool framework to achieve robust matching in the presence of outliers. When inlier ratios become very low RANSAC can take many

iterations to find a good model. This was addressed by Chum et al. with progressive sample consensus (PROSAC) [18] , which progressively increases the sample size. This approach assumes that match putatives can be prioritized, and in the usual case the descriptor distance is suitable. In [73, 74] feature weighting is integrated into the geometric verification procedure (as opposed to a post-processing step). Composition of incremental poses yields the trajectory in the global coordinate frame. In fact, in some applications the resulting trajectory is all that is needed, or the landmark measurements may be expressed with respect to their originating reference frames, as was shown in FrameSLAM [44]. In recent years, visual-inertial solutions have become increasingly popular, as inertial measurement unit (IMU) measurements prove to be complementary to visual measurements, and can lead to more consistent results [51].

2.2 Closing the Loop

VO trajectory drift is corrected by introducing loop closures before optimizing the vehicle trajectory. An overview of map-to-map, image-to-image, and image-to-map loop closure techniques is given in [89]. Appearance based image-to-image matching is the most flexible, as it makes no assumptions about trajectory or map quality, but operates on descriptors extracted from images directly, and therefore is also applicable to image retrieval in general [67]. Large-scale SLAM systems that made use of appearance-based image matching were demonstrated in [22, 23]. The idea behind appearance-based loop-closure detection comes from text-retrieval and search. A word is analogous to a a set of feature descriptors that are very similar, while a document is analogous to a single image frame. An image is then represented by all of the descriptors (words) it contains, and similar images in a video sequence can be identified by looking for images which have similar word occurrence histograms.

2.3 Map-Building by Smoothing and Mapping

Smoothing and Mapping (SAM) refers to the process of simultaneously estimating the full set of robot or camera poses and landmark locations in the scene [26, 28]. When this technique is applied to visual image features, it is called Structure from Motion [86], or bundle adjustment [85]. The cost function is typically taken to be the reprojection error of

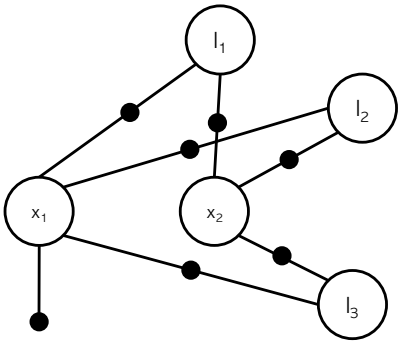


Figure 2: Factor Graph of two camera poses x_i and three landmarks l_j. The black nodes are the factors, which represent the landmark measurements taken by the respective cameras.

each landmark measurement into the image plane. Bundle adjustment has been applied to create highly accurate, city-scale reconstructions from large photo-collections [39, 82, 1].

Factor graphs offer a natural representation for the SAM problem. A factor graph is a bipartite graph containing two types of nodes: state variables and factors. In our case, the unknown camera poses $X = \{x_i | \, i \in 1...M\}$ and landmarks $L = \{l_j | \, j \in 1...N\}$ correspond to the set of state variables. The factors in the graph represent the landmark measurements $Z = \{z_k | \, k \in 1...K\}$. An exemplar factor graph is shown in Figure 2.

The non-linear cost-function to minimize is

$$\sum_{k=1}^{K} \|h_k(x_{i_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \tag{1}$$

where $h_k(\centerdot)$ is the measurement function of landmark l_j from camera x_i, and the notation $\|.\|^2_\Sigma$ represents the squared Mahalanobis distance with covariance Σ. We assume that we have normally distributed Gaussian measurement noise.

In practice one considers a linearized version of the problem, and the terms in equation 1 can be linearized as

$$\begin{aligned} & h_k(x_{i_k}, l_{j_k}) - z_k \\ \approx \quad & \left\{ h_k(x^0_{i_k}, l^0_{j_k}) + H^{i_k}_k \delta x_{i_k} + J^{j_k}_k \delta l_{j_k} \right\} - z_k \end{aligned} \tag{2}$$

where $H^{i_k}_k, J^{j_k}_k$ are the Jacobians of $h_k(\centerdot)$ with respect to x_{i_k}, l_{j_k} evaluated at $(x^0_{i_k}, l^0_{j_k})$.

During optimization the ordering in which variables are eliminated is crucial for performance. An approximate minimum degree (AMD) ordering is used in the GTSAM [7].

For more details on the SAM optimization process, refer to [26]. Some of the most popular software packages to solve these sorts of problems are GTSAM, Ceres, and g2O [27, 3, 47].

There have also been various efforts to improve the computational tractability. Some examples of this are application of the Schur complement trick to eliminate landmarks [2], smart factors [16], and divide and conquer approaches [64, 65].

2.4 Localization

As self-driving cars are moving closer and closer to becoming a reality, many researchers have been working on solutions to the localization problem. Many of these techniques rely primarily on vision. An extensive survey is provided in [57], and [34].

The work most relevant in terms of its application is that of Churchill et. al. [20][19][21]. Visual odometry trajectories, termed experiences, are stored each time the vehicle visits a new place and is unable to relocalize itself within existing experiences. The system keeps collecting new experiences until they become fully adequate for localization. One disadvantage of this work is that these experiences are only topologically linked, and exact metric pose recovery presents a challenge. Linegar et al. further extend this approach by introducing the concept of path memory, which encodes the robot's past use of the experience graph, and leverages this information during localization. [54].

Muehlfellner et. al take an approach which is similar in spirit to mine [62]. They build a map comprising many datasets, and optimize this offline. Only the most relevant landmarks, according to a variety of scoring functions, are included in the map, and the map is therefore a summary of what was actually observed. During localization, the most nearby landmarks are retrieved for pose recovery. Taking this a step further, landmark scoring functions are used by Dymczyuk et al. to summarize landmark maps so that the map remains at a relatively fixed size, while retaining the most useful features for robust localization[29]. The algorithm alternates between localization and offline maintenance procedures in order to perform this maintenance.

Most appearance-based localization methods operate with sparse landmark maps, but dramatic increases in computing and GPU power are beginning to make dense methods

feasibile at city scale. LIDAR is used together with cameras mounted on a vehicle to build a dense map of the survey environment in the recent work by Pascoe et al. [68]. LIDAR is not employed during localization. The camera is localized by minimizing the normalized information distance with respect to a rendered view of the dense 3D map. This approach is inherently robust to scene appearance changes, but its downside is that LIDAR is required during the mapping stage.

Another very relevant approach is that of Lategahn and Stiller, who used a pre-computed 3D map, comprising 3D landmarks and their descriptors, to localize a stereo camera without the help of GPS [49]. Given the previous known pose, all landmarks observed by the nearest camera pose used to build the map are used for descriptor matching. Lategahn et al. took a similar approach in [48], using a monocular camera during localization, and the resulting pose is refined in a filter together with IMU measurements.

Milford and Wyeth [61] introduced SeqSLAM. Rather than matching local features between images, sequences of images are compared to establish a loop closure. Image similarity is established using sum of absolute differences. Consequently, no lighting/season invariant descriptors are needed. The method works on sequences with drastically different appearance, e.g. night/day. The method does have some notable disadvantages, such as assumptions about relatively constant velocity and direction of travel, some of which were addressed in [71, 36]. Most recently, Pepperell et al. improved the approach by adding a directed graph to the image database to allow for branching at intersections, and images are rescaled to allow for greater invariance to lateral shifts, which are the primary source of pose variation in the authors' experiments [70].

Naseer et al. frame the sequence matching idea as a minimum cost network flow problem over a data association graph [**?**], and this is augmented with GPS information for better performance and the capability to handle loops in [88].

A related approach is that of Maddern et al. [59], in a system called CAT-SLAM. Sequential appearance based SLAM is enhanced with metric pose filtering to improve the performance.

Valgren et al. [87] also explored an appearance-based approach across scenes with stark

appearance changes, using SIFT/SURF descriptor matching, and tuning the parameters for optimal results. As expected, results are good on relatively similar data, but the system breaks down when snow-cover is present.

Stylianou et al. investigated the matching performance of feature descriptors over long time periods on a set of static webcams over a timespan of several years [83]. They found several trends in the data: There are seasonal variations, but they are dwarfed by changes in weather and also by changes in lighting. Specifically, they mention that matching sunny images against overcast images is more difficult than matching those taken in identical conditions. Moreover, in varying lighting conditions the feature detector often does not fire in the same place, so feature detection plays a large role in matching performance, as well. Despite these challenges, Lowry et al. showed promising results with predicting image appearance over diurnal appearance changes [58] , and Torii et al. used linear combinations of image descriptors to localize query images [84].

An evaluation of feature descriptors across different seasons was presented by Krajnik et al., and the authors also present a new bespoke feature descriptor called GRIEF [46]. McManus et al. learn place-dependent features for long-term localization by leveraging prior experiences of a place [60]. Neubert and Protzel present a system to solve the place recognition problem with the help of superpixels, combining the advantages of keypoints and fixed image patches [63].

More recently, Linegar et al. introduced a custom place-specific linear SVM classifier to recognize distinctive elements in the environment [55] . This approach uses unsupervised mining on single dataset which finds these distinctive elements.

As noted in the introduction, retrieving the best set of landmarks to match against is a major challenge, and this is especially true in the case of Structure from Motion (SfM), where unordered datasets with mostly unknown location priors are the norm. Li et al. [53] addressed this difficulty by matching 3D points to image features, rather than the more conventional 2D to 3D matching. Points with higher degree in the visibility graph are considered before points with lower degree. When a match is found, the priority of connected points is increased in the matching queue. They further improved on this by introducing

bi-directional matching in [52].

A similar approach is taken by Sattler et al. [77], where 2D-3D matching is sped up by indexing all image features into a vocabulary tree that was constructed using the 3D model, and the size of each word cluster is used as a proxy for estimated matching speed. Feature matching is prioritized according to cluster sizes. In [78] this approach is further refined with an active correspondence search in both directions. Once a match in the forward direction (image to map) is made, nearby 3D points can be used to attempt reverse matches. Hyperpoints are introduced in [76] to push the performance of this matching strategy even further.

Another interesting line of attack is reasoning about descriptor occurrence. One such approach is taken in [41, 40] by Johns et al., where robust localization is achieved by computing landmark observation likelihoods based on the number of times a landmark was observed across training runs. [42, 43]

It is standard practice to employ a RANSAC [31] framework to achieve robust matching in the presence of outliers. Formally, the goal of RANSAC is to obtain a set of inlier data points I in a set of putatives P. The algorithm randomly selects a minimal subset $S \subseteq P$, where the size of S is the minimum required to compute a desired model $M1$, e.g. in the case of camera pose estimation, $|S| = 3$. Given model $M1$ the set of inliers I is computed. If $|I|$ exceeds a predetermined threshold the algorithm terminates and a refined model $M1^*$ is computed from all inliers I. Otherwise, a new subset is randomly selected, and the algorithm iterates until a good inlier set has been found, or until a maximum number of iterations is reached, thus indicating failure.

RANSAC does not perform very well when when inlier ratios are very low, as it can take many (hundreds) iterations to find a good model. To this end, Chum et al. introduced PROSAC [18] , which progressively increases the sample size. This approach assumes that matches can be prioritized according to some quality measure, and in the usual case the descriptor distance is suitable and yields significant performance improvements. Raguram et al. improved upon this even further in [73, 74]. Feature weighting is integrated into geometric verification procedure (as opposed to post-processing step).

Choudhary and Narayanan explicitly model the visibility probability of a landmark, which leads to a probability guided matching algorithm [17]. A similar approach to the landmark retrieval problem is taken in [5, 6], under the assumption that a coarse location estimate is available for the camera to be localized. Specifically, the authors introduced a framework for predicting the visibility of landmarks in the scene. Given a new query image with a pose prior, the landmarks which were previously observed by nearby cameras are probabilistically weighted according to a distance metric which is learned in an offline step. The distance metric takes into account camera rotation and translation. This makes it trivial to ignore landmarks which were observed by a camera facing in the opposite direction, even though they are very close to the query camera. In the presented research I also assume that a coarse location estimate is available, and landmark visibility prediction is a component of the landmark retrieval algorithm.

A number of hybrid approaches for vehicle localization have been presented which combine different approaches, map sources, or sensing modalities. Senlet and Elgammal present a system for global vehicle localization which uses satellite imagery and road maps [80]. Good results have also been achieved by tracking the position of the vehicle with respect to lane markings [**?**]. Floros et al. introduced OpenStreetSLAM, an approach which used stereo visual odometry together with data from the OpenStreetMap (OSM) project for global localization [32]. Also using road maps, Brubaker et al. probabilistically localize the vehicle using OSM and visual odometry alone, but this approach fails in maps with many topological ambiguities, such as a Manhattan world [14].

Ros et al. presented an offline-online system in which 3D semantic maps were built offline, which could then be recalled during localization in real-time [75]. Cadena et al. present an approach which takes appearance, as well as geometry into account for loop closure detection by combining Bag-of-Words and CRFs, with better results than FabMap [15].

There have been a number of techniques which combine the best of topological and metric localization. For example, pose graphs with locally accurate points clouds attached to each node [24].Xu et al. demonstrated topometric localization on a road network represented as

a directed graph, each node has WI-SURF descriptor per node, as well as a road curvature descriptor. Redundant visits are removed during map building [90].

2.5 Submaps for SLAM

Submaps are often used when operating in a single map becomes computationally intractable, either due to size or computational limitations.

Bosse et al. introduced the Atlas framework, which uses a hybrid metric topological system of a pose graph with locally metric maps attached at each pose graph frame[13]. Estrada et al. presented a two-level hierarchical SLAM system, where local submaps are related to a global reference frame via conditionally independent transformations. All of the measurements are stored in the low-level submaps [30]. A similar approach with multiple overlapping submaps, each defining its location with respect to its neighbors, was presented by Leonard and Newman [50].

A large-scale stereo visual SLAM system with submaps was presented by Paz et al. [69]. Pinies and Tardos introduced a similar approach tailored specifically to monocular cameras, which operates in submaps which are conditionally independent given their coobserved landmarks[72].

Chapter III

STEREO VISUAL ODOMETRY

Visual odometry is a key component for map building and also localization. I present the specifics of my VO pipeline in this chapter. A high-level overview of the steps is shown in Figure 3. The main steps are image rectification, feature extraction, stereo feature matching, temporal feature matching, and finally pose recovery with the three-point algorithm and RANSAC. The output from the visual odometry algorithm is a camera trajectory, as well as a set of sparse landmarks and associated feature descriptors arising from the triangulation of feature points observed in both images. Both of these outputs are required in the map and localization modules of this work. The work in this chapter was initially presented in [11].

3.1 Claims

Visual odometry plays an important role during localization, as well as map building. It is therefore appropriate to think about VO in terms of the claims I make about the system as a whole. I make the following claims about VO, which are are supported by experiments with real data as shown in Section 3.4.

- **High-quality** - Stereo visual odometry provides highly accurate pose estimates.This is validated with on benchmark datasets.
- **Efficient** - The stereo visual odometry algorithm runs in real-time. This is achieved due to parallelization of the processing pipeline, which is explained in Section 3.4.2.
- **Robust to change** - Visual odometry is robust to apparent short term changes due to moving objects, as well as appearance changes. Changes induced by motion are very common when operating on city streets with vehicular traffic. This claim is validated in Section 3.4.3.

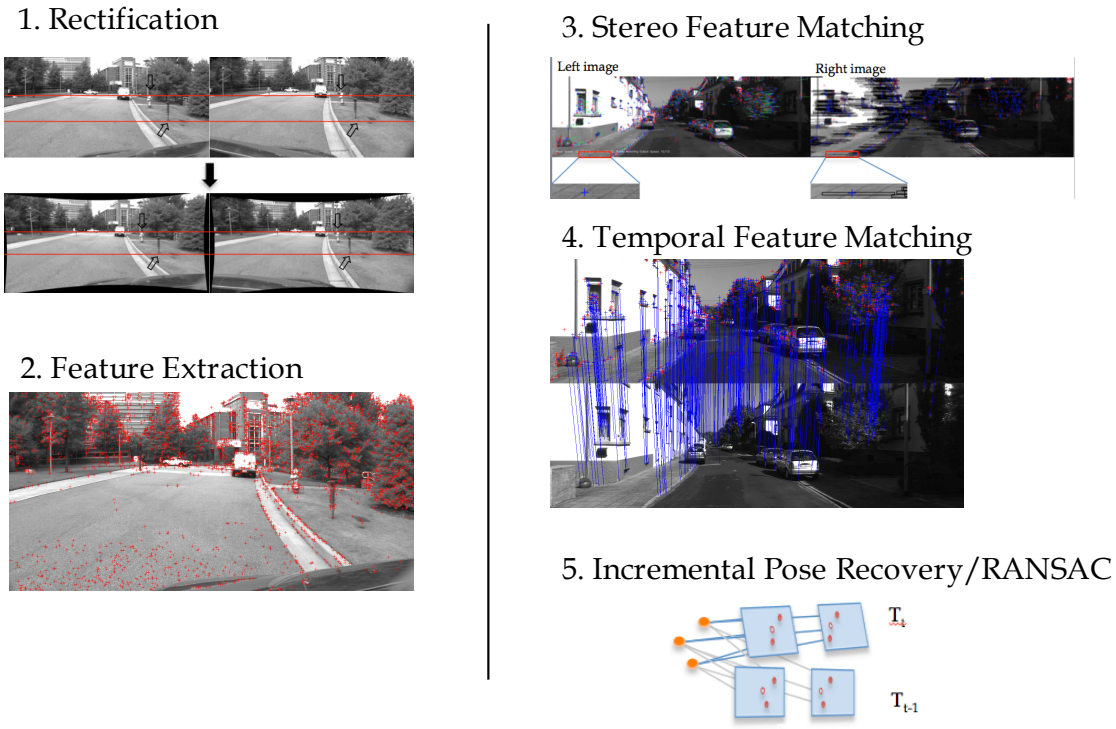


Figure 3: Overview of the visual odometry pipeline. 1. Image rectification 2. Feature extraction in the left and right images 3. Feature matching between the left and right images, constrained to a small bounding box 4. Temporal feature matching in the left image, also constrained to a bounding box, and landmark triangulation 5. Incremental pose recovery between consecutive frames with the three-point algorithm in the context of RANSAC.

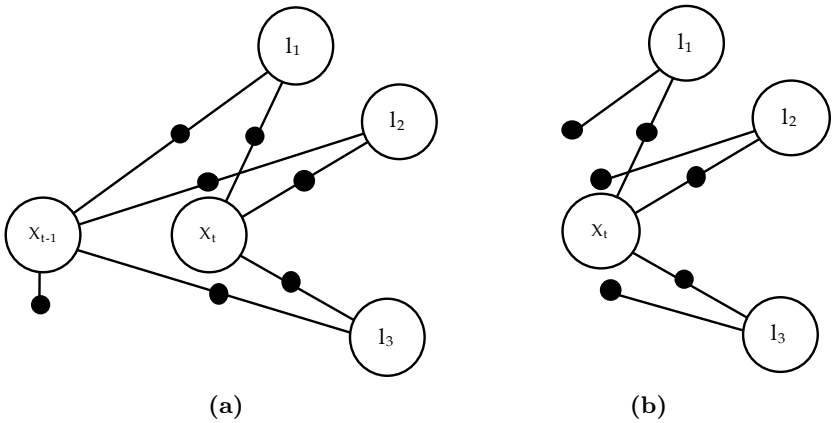


Figure 4: (a) Factor graph for the three point pose recovery problem with binary projection factors between each of the poses and landmarks, and a pose prior on x_{t-1}. (b) An improved version of the problem, taking advantage of the fact that x_{t-1} can be assumed as fixed, since we are only interested in the relative pose of x_t with respect to x_{t-1}. This reduces the binary factors involving x_{t-1} to unary priors on the landmarks.

3.2 Feature Detection and Stereo Matching

For each rectified stereo image pair, features are extracted and matched across the stereo pair. Matches are only retained if they are mutually optimal according to the ratio test [56], and fall within a tight threshold ($1px$) of the epipolar line, which is a horizontal scan-line for rectified images. In practice, the search region for stereo correspondences can be restricted even further to a bounding box of a limited width, as disparities above a certain threshold are unlikely to occur. Next, stereo matches with very small disparity ($< 0.5px$) are discarded as their depth uncertainty is very large, and 3D landmarks $(X, Y, Z)^\top$are then triangulated using the known stereo calibration. A 2D feature point (u_L, v_L) taken from the left camera with an associated disparity $d = u_L - u_R$ can then be reprojected to 3D coordinates in the left camera coordinate frame:

$$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (u_L - c_x)\, Z/f_x \\ (u_R - c_y)\, Z/f_y \\ bf_x/d \end{bmatrix} \quad (3)$$

3.3 Temporal Matching and Incremental Pose Recovery

Features are then matched temporally to form a set of putative matches. Depending on the frame rate of the camera, the putative matches could only have translated a limited distance in the image, and therefore this search is also restricted to a bounding box. The incremental 3D transformation $T \in SE(3)$ which expresses the current camera pose x_t with respect to the previous pose x_{t-1} is recovered by way of applying a three point algorithm within a RANSAC framework [31, 38]. A minimum set of three landmarks is needed, which can be represented with a small factor graph containing two poses and three landmarks, as shown in Fig 4. In practice, and to gain a slight computational advantage, I assume that camera pose x_{t-1} is fixed, and we only need to optimize over the landmarks and camera pose x_t. Landmarks are initialized from the stereo triangulation, and x_t is initialized with the previous incremental motion. The factor graph then captures a non-linear least-squares problem

$$\Theta^* \triangleq \arg\min_{\Theta} \sum_{m=1}^{M} \|h_m(\mathbf{x}_{i_m}, \mathbf{l}_{j_m}) - \mathbf{z}_m\|^2_{\Sigma_m} \tag{4}$$

where $h_m(\cdot)$ is the measurement function of landmark l_{j_m} from pose r_{i_m}, and M is the total number of measurements, and $\mathbf{r} \in \mathbf{R}$ and $\mathbf{l} \in \mathbf{L}$ .The measurements are denoted by $\mathbf{z}_m = (u_L, u_R, v)$, where u_L and u_R are the horizontal pixel coordinates, and v the vertical pixel coordinate, all of which result from the projection of a tracked 3D point into the stereo pair. Only one value is needed for v because the stereo rig is rectified, and hence $v_L = v_R$. The measurement function $h_m(\cdot)$ takes a landmark l in world coordinates, transforms it into the left camera coordinate frame to obtain p_i, and then projects this point into the stereo pair according to

$$u_L = \frac{f_x X}{Z} + c_x, u_R = \frac{f_x (X - T_x)}{Z} + c_x, v = \frac{f_y Y}{Z} + c_y \tag{5}$$

where T_x is the baseline between the stereo cameras. This is shown in Figure 5.Finally, repeated composition of T with the previous camera pose yields the camera trajectory in the global coordinate frame.

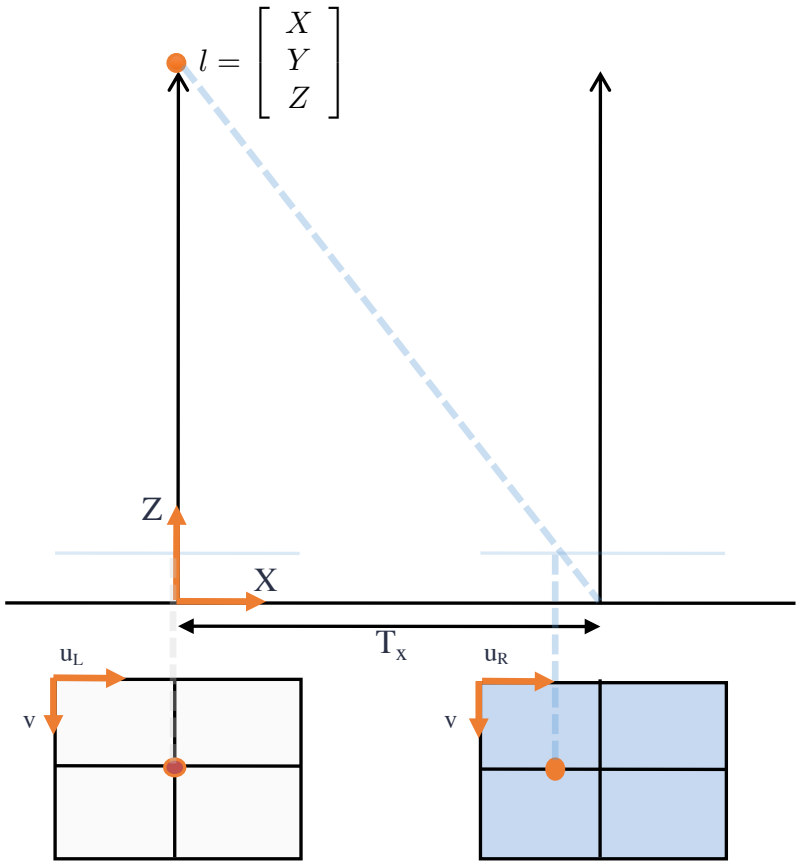


Figure 5: Schematic showing the projection of landmark l into a stereo camera, where the 3D scene is viewed top-down, and the image planes are shown from the front.

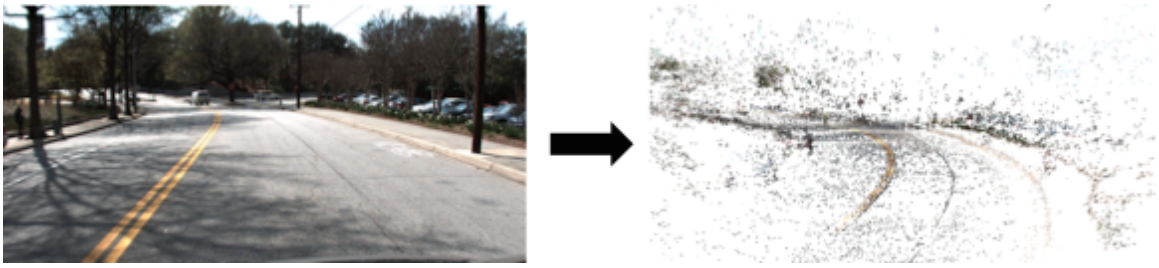


Figure 6: Sparse point cloud generated generated by running visual odometry and transforming the triangulated stereo points into the global coordinate frame.

Features which are successfully tracked for at least two consecutive frames, called feature tracklets, are recorded for later use along with their feature descriptors. As these feature tracklets have been proven to be geometrically consistent across at least two frames they are excellent candidates for inclusion in the map used for localization. Figure 6 shows an example of a sparse point cloud generated by my VO pipeline. The resulting camera trajectory, together with the accepted landmarks is optimized later as described in Section 4.5.

3.4 Results

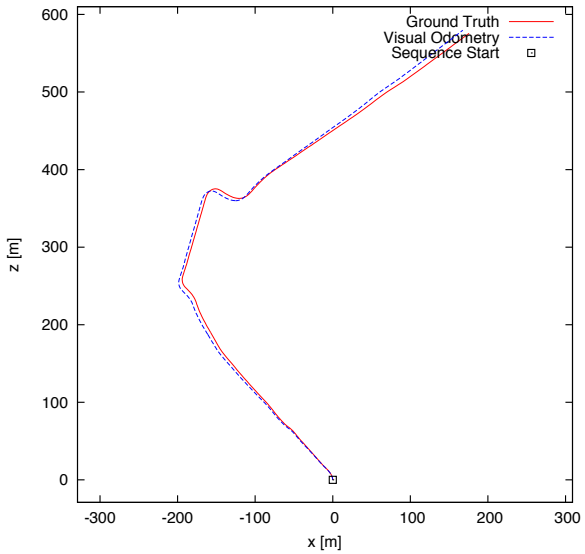
The visual odometry algorithm I have discussed in the previous sections was first used in the context of underwater reconstruction [11]. Since then, I have improved upon it by making it faster and more accurate, as detailed in the results that I describe here. The results reinforce the claims that visual odometry is of high quality, real-time, and robust to change, which are all necessary properties for it to function effectively as part of the SRM-STM localization system.

3.4.1 High Quality

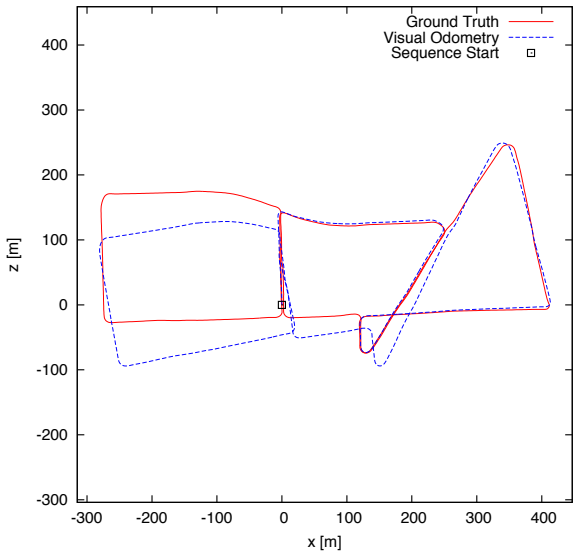
The stereo visual odometry algorithm I have presented held the top-ranking spot in the KITTI VO benchmark among competing approaches which operate only frame-to-frame [35], although it was not real-time at that time. Visual odometry accuracy is easily evaluated in terms of accumulated translational and rotational error, averaged over different distances of a ground truth dataset, as well as computation time. Figure 7shows results on the KITTI benchmark [35], where my VO pipeline achieved top-ranking performance of 2.54% translational error, and 0.0078 deg/m of rotational error at the time of submission in 2012. The benchmark reports errors averaged over 5, 10, 50, 100, 150, ... 400m.

3.4.2 Efficient

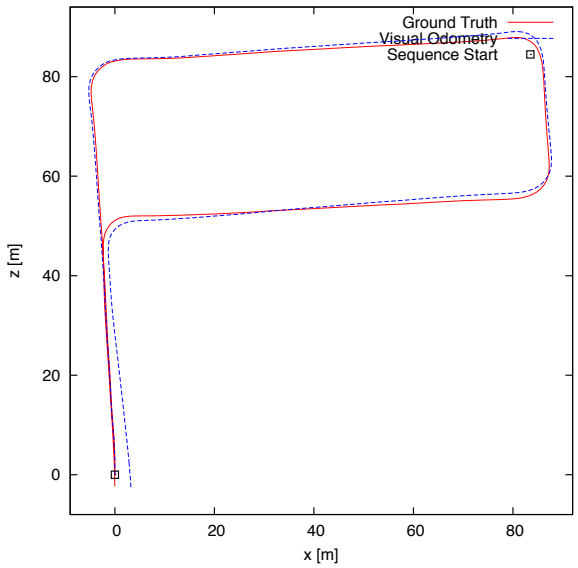
The VO algorithm I developed is highly parallelized, and runs in real-time ($\sim 10Hz$) for a variety of feature descriptors and image resolutions. Feature extraction is by far the most expensive operation, followed by feature matching. Both of these operations can be



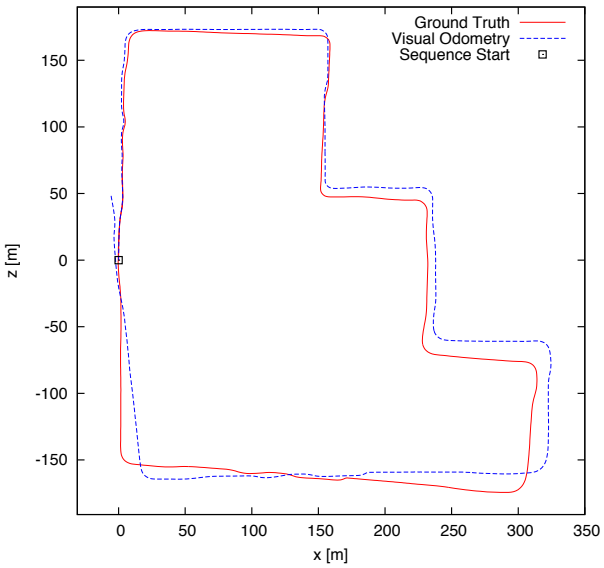
(a) Test Sequence 11



(b) Test Sequence 13



(c) Test Sequence 14



(d) Test Sequence 14

Figure 7: Visual odometry trajectories generated with my pipeline evaluated against the KITTI benchmark [35]. A few degrees of rotational error are accumulated over the course of thousands of poses.

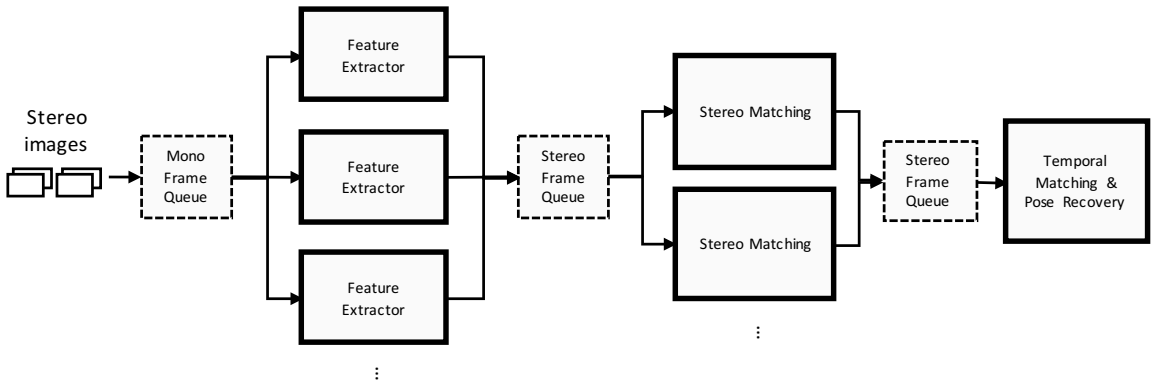


Figure 8: Visual Odometry pipeline. Individual stereo frames progress through a parallel bank of feature detectors. Stereo frames are then reassembled, and move on to paralleized stereo matching. The last stage is temporal matching and incremental pose recovery with the three-point algorithm in a RANSAC framework.

Table 1: Performance of the visual odometry pipeline on KITTI and Georgia Tech datasets with all elements of the pipeline running on the CPU on an 8 core Intel i7 chip at 4Ghz.

	Resolution	FAST/BRIEF (fps)	SIFT (fps)
KITTI	1241×376	16	11
GT	1380×480	22	9

parallelized, as shown in Figure 8. I have implemented the pipeline so that feature extraction is carried out in parallel threads, of which I create as many as are needed. The ideal number of feature extractors depends on the frame rate, the image resolution, as well as the choice of feature descriptor (SIFT,SURF,FAST, etc.). Feature extraction and description may be offloaded to the GPU for even better performance. Stereo matching is also carried out in parallel. Finally, at the end of the pipeline there is a single thread running temporal matching and RANSAC/pose recovery.

Quantitative results with the entire pipeline running on the CPU are shown in Table 1 for a selection of datasets and feature descriptors.

3.4.3 Robust to Change

The VO algorithm as I have described it so far already performs very well in relatively static scenes due to the use of RANSAC, but special care must be taken in more challenging scenarios. Highly dynamic scenes, for example heavy automobile traffic, or substantial variations in lighting conditions can cause problems. Uneven lighting can lead to nonuniform feature distribution, which in turn can result in biased rotation estimates. Figure 9 shows

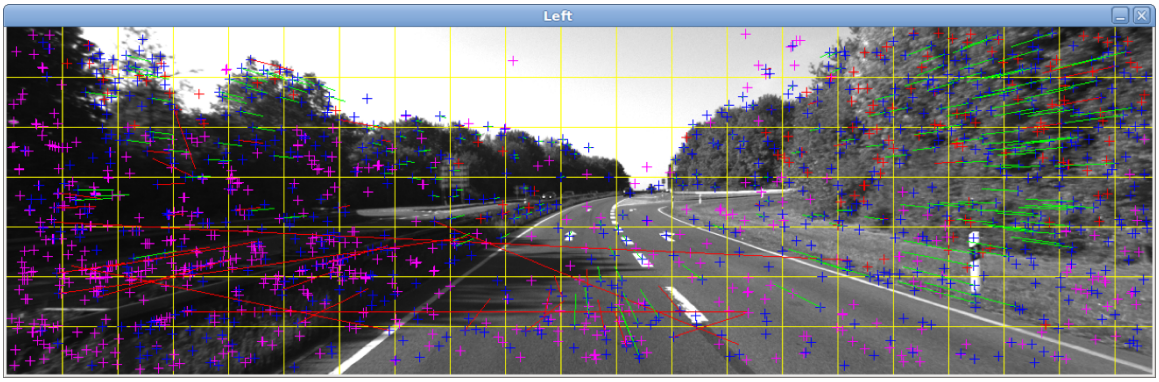


Figure 9: Feature tracking with feature binning. The left side of the image is much darker than the right, so a uniform distribution is achieved by dividing the image into a grid, and retaining at least n features in each grid cell.

an example where the left side of the image is much darker than the right, which results in far fewer features in that portion of the image. To prevent this, I divide the image into a grid, and after running feature extraction with much lower thresholds, I retain only the k strongest features in each grid cell. Stereo matching and pose recovery then proceed as usual. Experimentation showed that this yields much better results on some data, and feature binning was critical to achieving the top-ranking results on the benchmark results reported in Section 3.4.

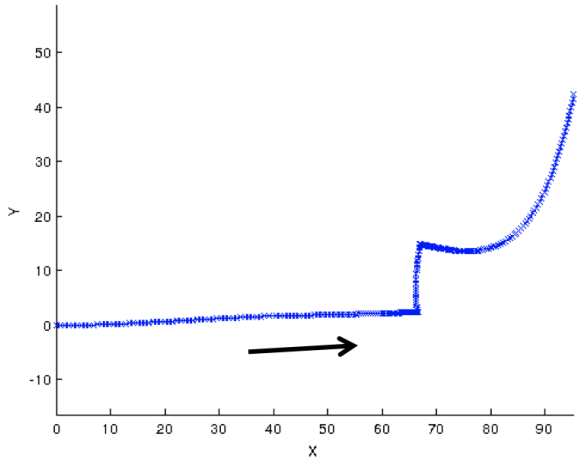
3.5 Summary

In this chapter I have presented the visual odometry system which is used as part of localization with the SRM, as well as during map building. The VO pipeline presented here outputs a camera trajectory, as well as sparse feature tracklets. These outputs are used in the next chapter for map building. The outputs are also used in the complete localization system. With the results I have shown that it satisfies these claims:

- **High-quality** - Results form the KITTI benchmark have shown that VO achieves accuracy of 2.54% translational error, and 0.0078 deg/m of rotational error, as averaged over several different vehicle trajectory lengths. This level of accuracy is more than enough to initialize camera trajectories for map building, and also to serve as source of incremental pose updates during localization.

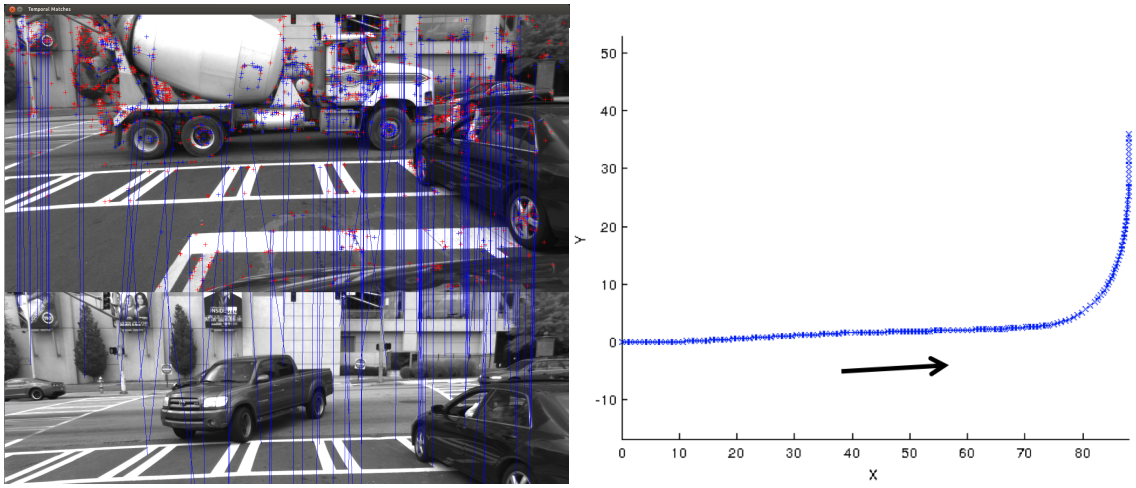
- **Efficient** - The stereo visual odometry algorithm runs at up to 11fps with SIFT, and

(a) **(b)**



(c)

Figure 10: Pose recovery failure caused by a large vehicle moving laterally through the field of view while our data collection vehicle is stopped at an intersection (a) Temporal putative matches between features (red) are shown in blue (b) Estimated inlier flow from previous frame is highlighted in green (c) Vehicle trajectory indicating an inconsistent sideways jump.



(a) **(b)**

Figure 11: Correct trajectory result after incorporating motion model rejection for incremental pose recovery, together with key-framing.

up to 22fps with FAST/BRIEF on CPU. The speed depends on a number of variables such as the feature detector, the image size, and the number of features detected. As a result, the parameters can easily be tuned to achieve the desired performance.

- **Robust to change** - The visual odometry algorithm has been made robust to change thanks to keyframing and motion-model pose rejection. As a result, the VO pipeline is able to process all of the data I have collected without any failures.

In the next chapter the output from VO will be used to build large scale 3D maps which are used as a basis for localization.

Chapter IV

MAP BUILDING

Accurate localization requires a highly accurate map of the environment against which localization will be performed. I show in this chapter that such a map can be constructed by combining data from multiple data sequences collected at different times into a spatio-temporal map (STM). Again, I begin this chapter by outlining the claims that to be made about the STM and Submap-STM presented in this chapter, followed by the data collection apparatus and methodology. I then discuss the map building procedure that was used to build the spatio-temporal map, followed by results which validate the claims which I make about the STM. The work in this chapter was initially presented in [10, 9, 4].

4.1 Claims

I make the following claims about the spatio-temporal map, and these are supported by results in Sec. 4.7.

- **High-quality** - The map built of multiple datasets is of high quality as measured by reprojection errors. Results which support this claim are shown in Section 4.7.1.
- **Efficient** - The Submap-STM is efficient to construct because the size of each submap is bounded. Submapping ensures ensures that this remains the case even as the total coverage area and number of datasets in the map grows. Furthermore, the Submap-STM supports real-time retrieval of landmarks and descriptors during localization, which is necessary for real-time operation of the overall system. Results to support this claim are shown in Section 4.7.2.
- **Robust to change** - The spatio-temporal map is inherently robust to change because it incorporates landmarks observed at different times of year. This is demonstrated by the distribution of landmarks in the map, as well as the localization experiments in

(a) **(b)**

Figure 12: Picture of front stereo camera assembly mounted on car.

Table 2: Data collection equipment.

Name/Description	Quantity
Point Grey FL3-GE-14S3M-C GigE monochrome camera	2
Point Grey FL3-GE-14S3C-C GigE color camera	1
Microstrain 3DM-GX3-45 GPS-INS unit	1
Laptop for data collection	1
Gigabit Ethernet switch	1

the following chapter. Section 4.7.3 shows results which demonstrate that landmarks are evenly distributed.

4.2 Data collection

The data were collected using PointGrey Flea3 cameras running at 10Hz, mounted on a conventional car rack along with a 3DM-GX3-45 Microstrain GPS-INS unit, which outputs data at up to 100Hz. The data collection equipment is summarized in Tab. 2. There are two monochrome cameras to form a stereo-pair, as well as a color camera which is used solely for visualization purposes. All sensors are connected to a single laptop for data collection while the car is driven around GT campus. The GPS-INS unit records GPS, as well as an EKF-filtered solution which includes attitude and heading. The GPS-INS data was interpolated and sub-sampled in a post-processing step to line up with image timestamps. A diagram showing how everything was connected is shown in Figure 13.

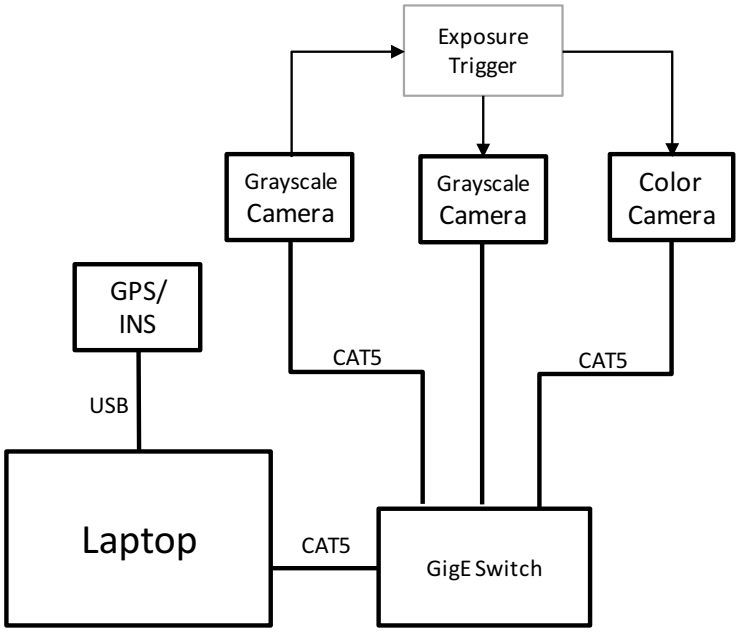


Figure 13: Data collection wiring schematic. The GPS/INS unit is connected directly to the laptop through USB. The GigE cameras are connected to a network switch which is connected to the laptop. One of the cameras is designated the master camera, and triggers exposure of the other cameras through hardwired circuit.

Table 3: Georgia Tech campus stereo and IMU datasets, along with notes that specify limitations or quality issues due to hardware or software failure.

Label	Date	Number of frames	Resolution	Notes
A	2012/08/13-A	10292	1380×480	North campus only
B	2012/08/13-B	11199	1380×480	North campus only
C	2012/08/14	29685	1380×480	Outside of campus
D	2012/09/02	29996	1380×480	
E	2012/09/11-A	22663	1380×480	
F	2012/09/11-B	20373	1380×480	
G	2012/10/06-A	21460	1380×480	Poor GPS and calibration
H	2012/10/06-B	25204	1380×480	Poor GPS and calibration
I	2012/11/20	20389	1380×480	Partial GPS
J	2013/02/05	25525	1380×480	
K	2013/04/02	23091	1384×680	
L	2013/08/01	21691	1384×680	
M	2014/04/03	21282	1384×680	
N	2015/03/25	24995	1384×680	

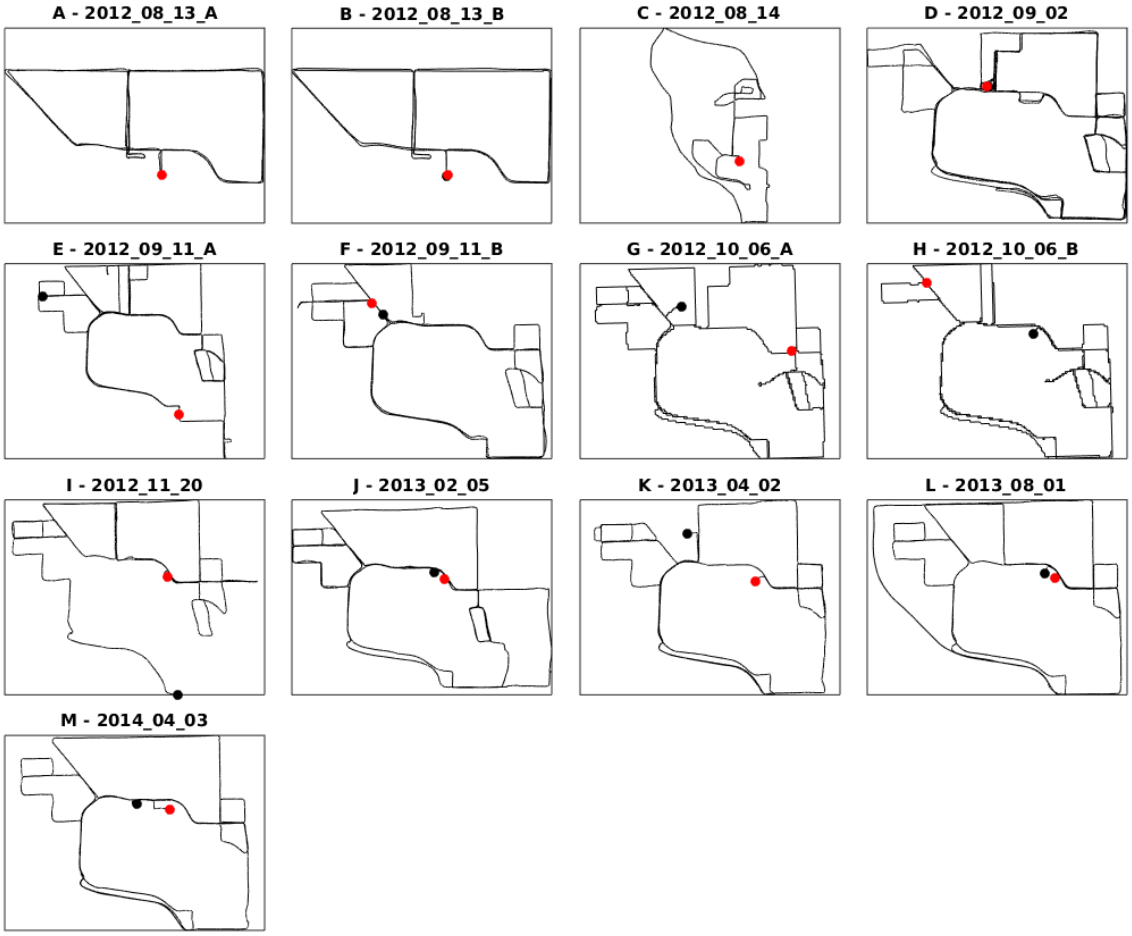


Figure 14: GPS traces for a subset of the Georgia Tech datasets. The black dot indicates the starting position, and the red dot the last frame. The start and end positions coincide for some datasets, which is indicated by a red dot only.

Over one million images have been collected. Table 3 shows the data collected, as well as the approximate size of each dataset. Some of the GPS traces are shown in Figure 14.

A computer with an i7-3400 processor, 32GB of RAM and 4TB of storage is used for running the presented algorithms on the obtained data. The algorithms make use of a number of open-source software packages, including the Georgia Tech Smoothing and Mapping library developed in our lab, as well as the popular computer vision library OpenCV.

GPS data—and especially elevation data from GPS—is noisy. So I preprocess the GPS data by correcting the elevation from digital elevation models (DEM) obtained from the United States Geological Survey (USGS). I also compared this with Google Map data, and while they are very close most of the time, USGS data is easier to work with as it can be

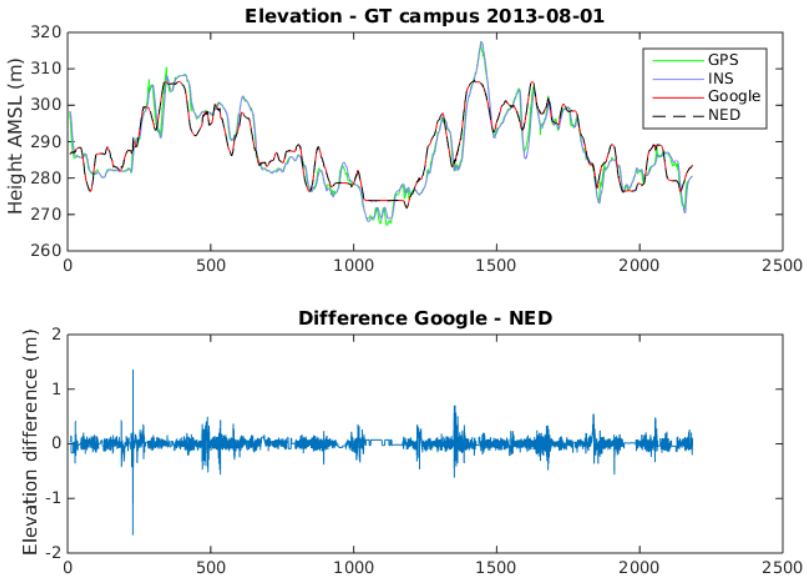


Figure 15: Raw GPS and INS solutions compared to elevation data obtained from Google Maps and USGS surveys.

stored locally. Figure 15 shows a subset of dataset L and its elevation compared to the DEM sources.

Each data sequence is also individually calibrated at the beginning of each data collection run with a rigidly mounted checkerboard pattern, as shown in Figure 16

4.3 Map Building Overview

I now describe the procedure to build the map used for localization. This map consists of landmarks $L \in \mathbb{R}^3$, as well as the camera poses $X \in SE(3)$ from which these landmarks were observed. Each landmark has associated with it feature descriptors, as well as the times at which the landmark was observed. As shown in Figure 17the map also contains a lookup table which maps camera poses to landmarks. This map will be used for localization, so it must fulfill a number of requirements:

- Metrically accurate
- Good spatial coverage of landmarks
- Good coverage of different datasets to capture the varying appearance

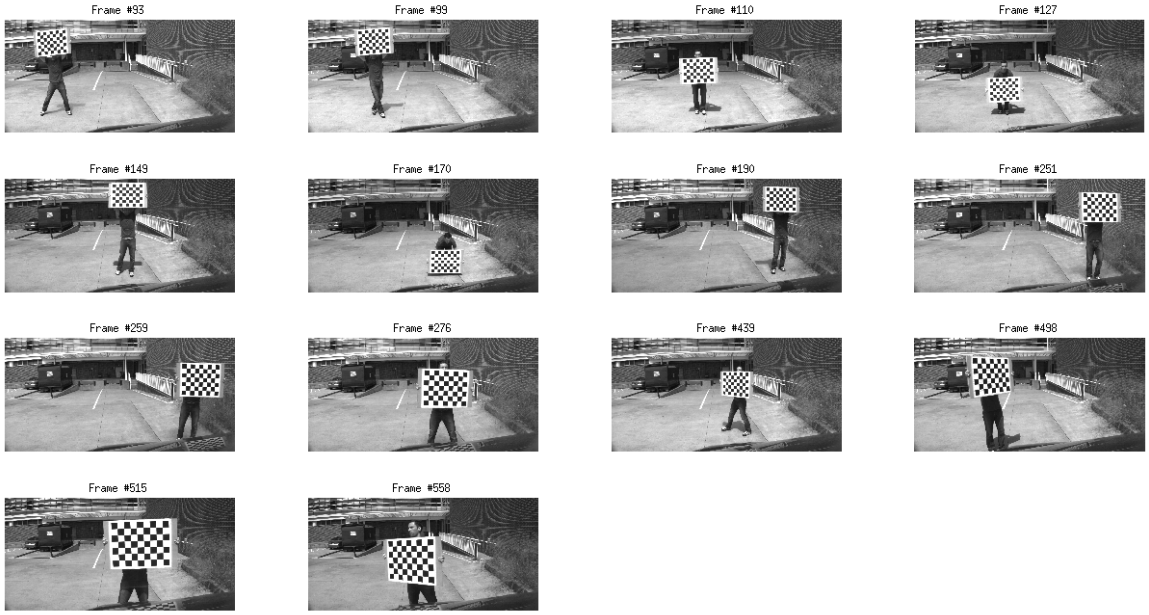


Figure 16: Camera calibration with a rigid calibration grid. The grid is moved around the overlapping viewing of the stereo cameras, and its known dimensions are then used to recover the intrinsic and extrinsic parameters.

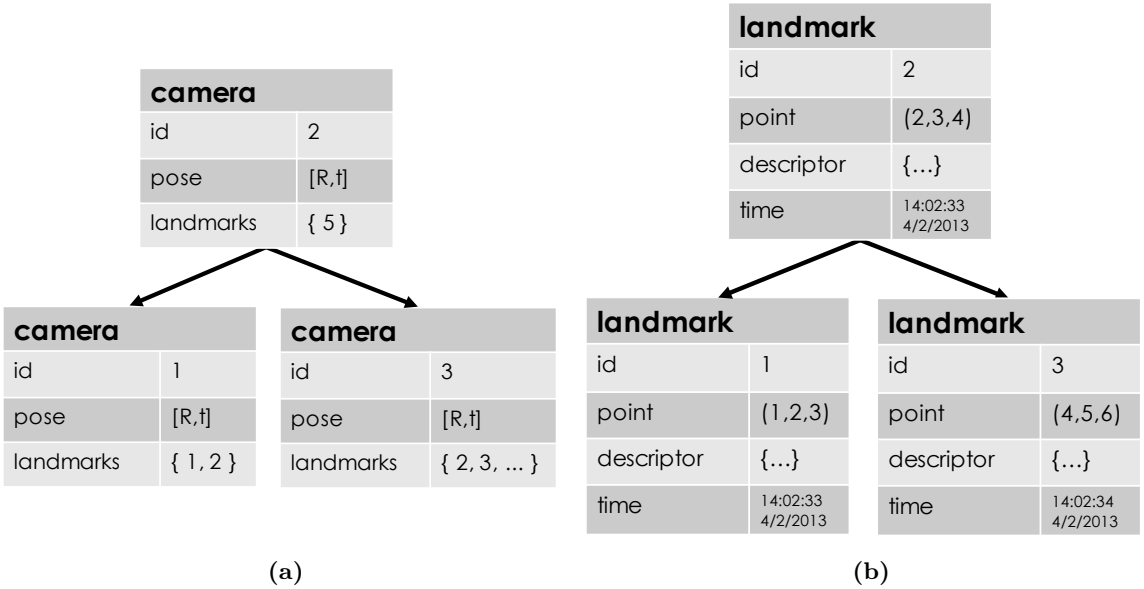


Figure 17: Data structures used for the STM. (a) Cameras are stored in a binary tree, where each camera node contains an id, 3D pose, and list of landmarks observed from this camera. (b) Landmarks are stored in a tree, and each is represented by an id, 3D point, feature descriptor and observation time.

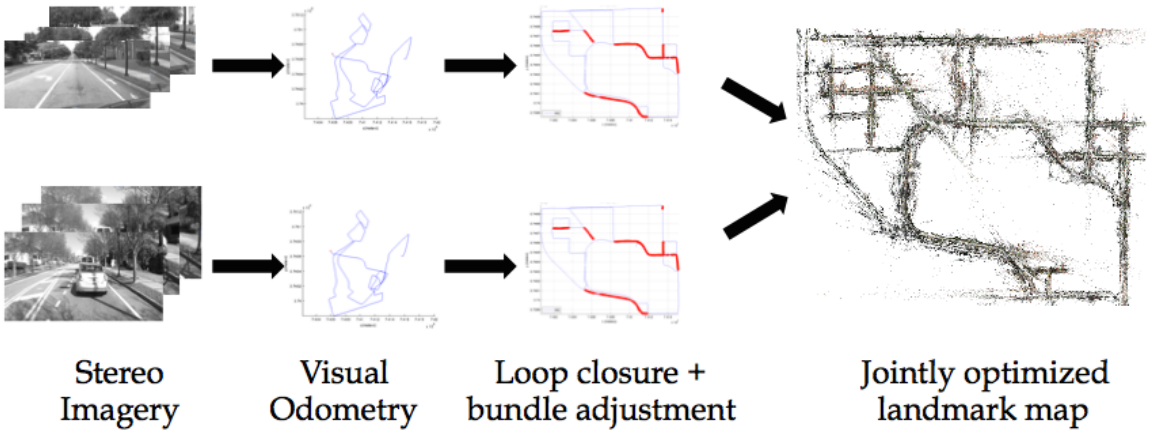


Figure 18: STM building pipeline for two image sequences. After running visual odometry on each data sequence, loops are detected and geometrically verified. This is followed by loop closure detection between the individual data sequences and another round of optimization to obtain the globally consistent STM.

This map encodes information about space, as well as time, which is why I call it the spatio-temporal map (STM). A high-level representation of the STM-building algorithm is shown in Figure 18. After running stereo visual odometry on each data sequence, loops are detected and geometrically verified. This is followed by loop closure detection between the individual data sequences and another round of optimization to obtain the globally consistent STM. This basic approach has scalability issues, which I address in Sec. 4.6 by introducing submaps.

4.4 Closing the Loop

Loop closure detection serves several functions. Loop closure detection is necessary to correct for drift accumulated by the VO algorithm, which appears as map alignment errors when parts of the trajectory are revisited. The loop closures serve as additional constraints during map optimization, as described in the following section. As parts of the map are revisited, loop closure detection identifies matching landmarks and prevents duplicate entries in the map.

There are essentially two steps to most loop closure detection approaches:

1. Detecting a set of potential loop closures candidates. There are several ways to do this. Appearance-based, or with help of GPS are two of the ways discussed below.

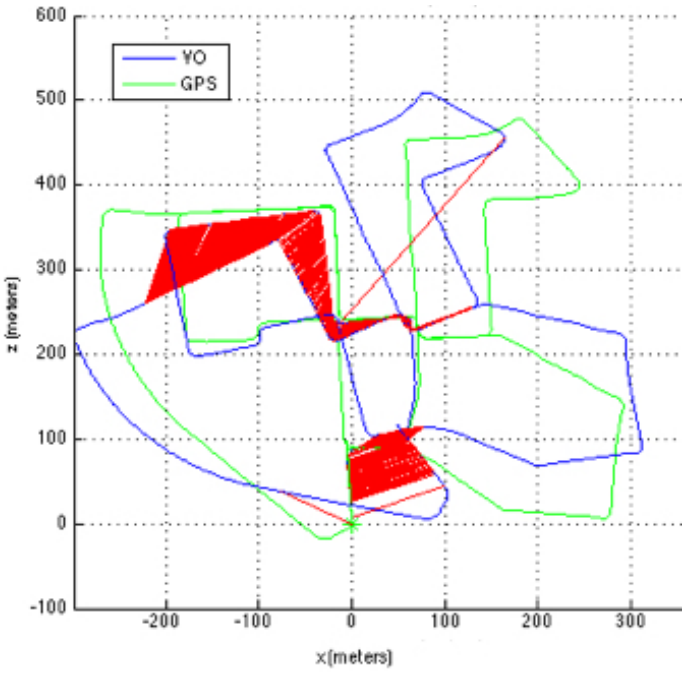


Figure 19: Appearance-based loop closure result including a false loop closure. The visual odometry trajectory is shown in blue, GPS in green, and loop closures are shown in red.

2. Geometrically verify loop closure candidates. This follows the same steps as incremental pose recovery with the three-point algorithm in the context of RANSAC.

Care must be taken to avoid/detect incorrect loop closures, as these may cause catastrophic failure during map optimization.

4.4.1 Appearance-based Loop Closure

Loop closure detection algorithms based on vocabulary trees and bag of words are as in [22, 67], yield satisfactory results within some of my sequences, but due to perceptual aliasing and other effects, many loop closures are actually missed. Figures 19 and 20 show an example of this approach on one of the KITTI sequences. Figure 19includes an incorrect loop closure which results in catastrophic failure during map optimization, due to the global inconsistency introduced by this loop closure. After careful tuning of parameters and inlier thresholds I was able to obtain the result in Figure 20.

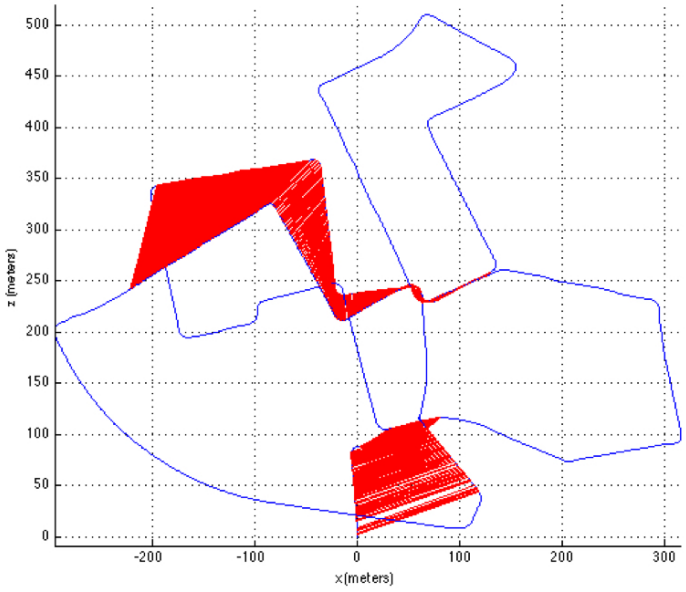


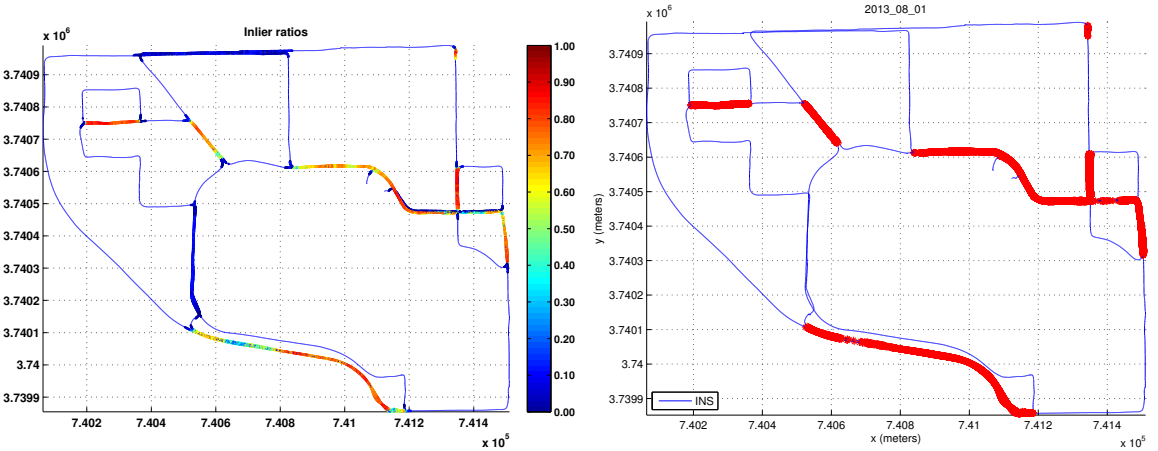
Figure 20: Appearance-based loop closure result. The visual odometry trajectory is shown in blue, and loop closures are shown in red.

4.4.2 Loop Closure Detection With GPS

The data we have collected has GPS measurements for each frame, so I use this to identify likely loop closures and which frames are likely to match, and geometrically verify those. Using GPS also has the added benefit of eliminating grossly erroneous matches as shown in Figure **??**, and it also reduces the likelihood of missed loop closures due to perceptual aliasing. Given the GPS readings, I take a brute force approach, which is acceptable during the map-building stage as I am not concerned about real-time performance here. For each frame in the data sequence I find the k nearest neighbor camera frames, and attempt feature matching and geometric verification for each one. A loop closure is accepted when a minimum number of inliers is found. Loop closure landmark observations are recorded to be incorporated into the map during the following optimization step (Sec. 4.5). Figure 22 shows an example of GPS-based loop closure detection applied to one of the data sequences.

To combine multiple datasets into the STM, I repeat this procedure and also apply it in between data sequences. An example of this brute-force matching between sequences is shown in Figure21 This image has significant appearance changes, including lighting, foliage, and vehicular occlusions, but in this instance geometric verification was still successful.

Figure 21: Successful loop closure and pose recovery on challenging imagery between frames from sequences K (top) and L (bottom). There are notable differences in lighting, foliage, as well as vehicular occlusions. Putative matches are shown in blue, and accepted inlier matches are shown in green.



(a) **(b)**

Figure 22: Loop closure results for sequence L. (a) Inlier ratios (b) Accepted loop closures exceeding the inlier threshold and minimum inlier count are shown in red.

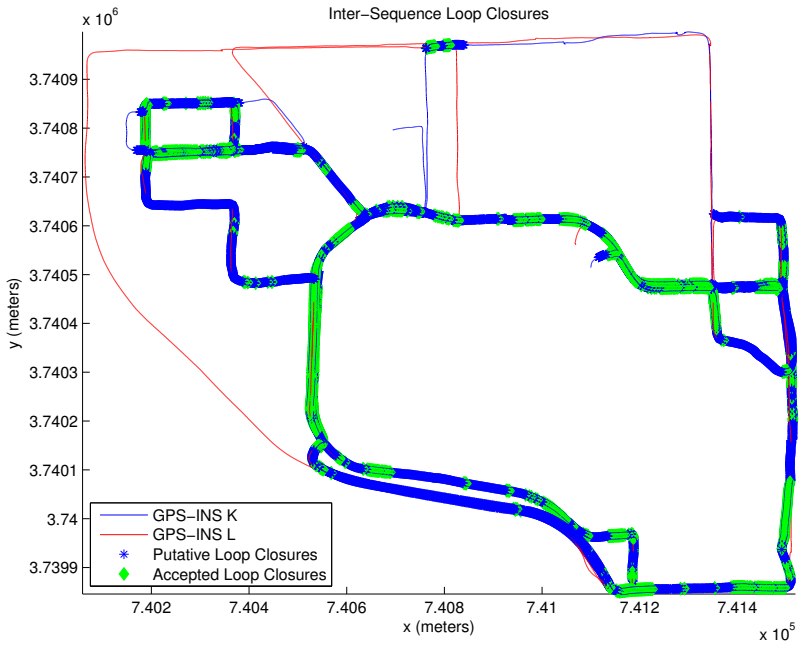


Figure 23: Loop closures between sequences K & L. Poses where loop closure is deemed possible (and attempted) are shown in blue, and accepted loop closures are shown in green.

It is crucial to identify as many loop closures between sequences as possible, since that will yield the most consistent 3D map. Missed loop closure detections could also cause multiple inclusions of landmarks in the database, consequently leading to poor localization results. On the other hand, accepting false loop closures can lead to catastrophic map errors. Through experimentation I found that for our datasets a minimum inlier ratio of 0.5, and a minimum RANSAC inlier count of 10 yields satisfactory results. This is much lower than the inlier ratios typically observed in VO, which are often as high as 95%. Figure 23 shows the loop closure result between sequences K and L.

4.5 Map Optimization

After loop closure detection, I run a full bundle adjustment step on each data sequence individually, before combining them into the STM, where I apply bundle adjustment once more. The STM is optimized by applying nonlinear Levenberg-Marquardt optimization such that I minimize the reprojection error of each landmark into each of the cameras which have observed it. Specifically, I minimize the non-linear cost function

$$\sum_{k=1}^{K} \|h_k(x_{i_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \tag{6}$$

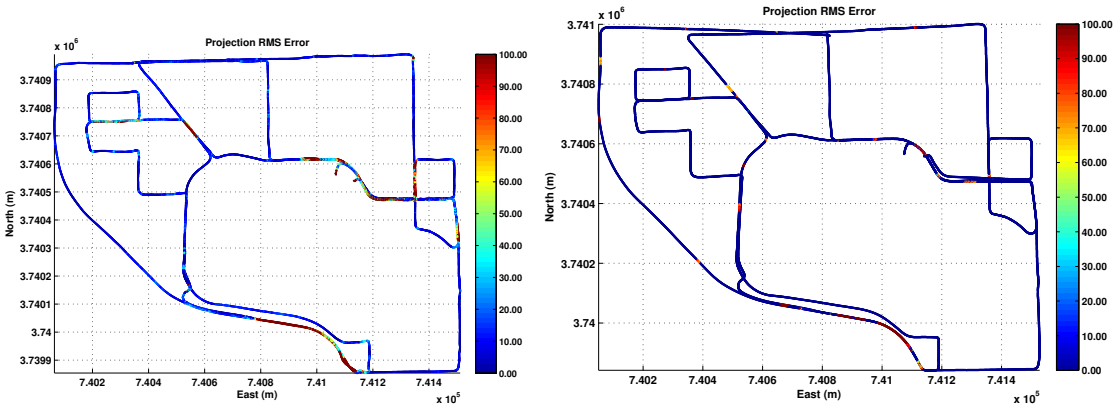


Figure 24: Sequence L, per-camera RMS errors. (a) Initialization (b) Optimized.

where $h_k(\centerdot)$ is the projection function of landmark l_j from camera x_i, and the notation $\|.\|^2_\Sigma$ represents the squared Mahalanobis distance with covariance Σ. I assume that we have normally distributed Gaussian measurement noise.

Large scale nonlinear optimization problems are very sensitive to initialization. The presence of outlier feature tracks, or inconsistent VO trajectories can easily result in divergence. Camera poses X^s are initialized from GPS, and landmarks L^s are initialized by transforming each from the observed camera frame to the global frame. I also add weak GPS priors to camera poses to accurately geo-register the map. All coordinates are expressed in the local UTM frame. Loop closures introduce many new measurements into the graph, and these are expected to have high reprojection errors before optimization. The Huber cost function is used to achieve robustness, and to cope with potential outliers. Root mean square projection errors per camera, before and after optimization, are shown in Figure 24.

Two or more maps are combined in a final optimization step. Here, cameras and landmark variables are initialized from the individual map optimization results. Landmarks which were observed in one or more sequences (as detected by the previous loop closure stage) are reconciled and represented in the map just once. Furthermore, due to memory constraints I only retain landmarks which were tracked for at least three consecutive frames. An example of two datasets being optimized together is shown in Figure 25 At over 2.2

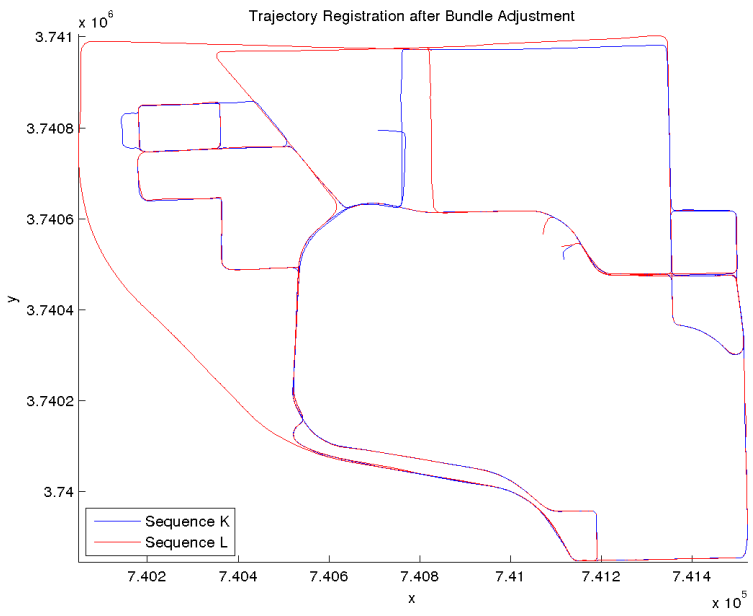


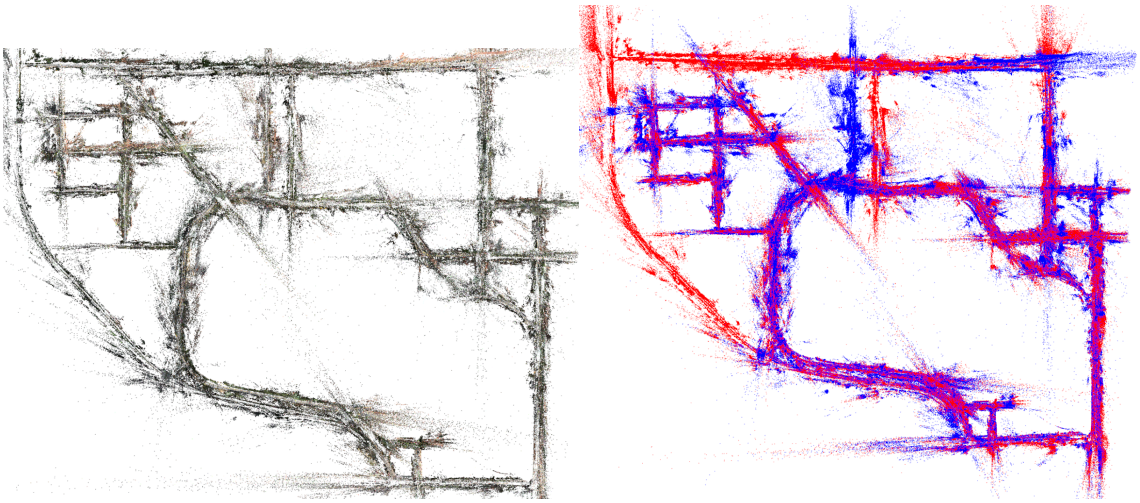
Figure 25: Optimized camera trajectories after full bundle adjustment of over 12 million factors and over 2.2 million variables.

million variables (cameras and landmarks), with over 12 million observations, the optimization requires just over 31GB of memory, and so it just barely fits into RAM on a typically configured desktop PC. This huge memory footprint is due to the very large cliques which are induced in the factor graph during elimination, resulting in large amounts of infill in our otherwise sparse system of equations. Sec. 4.6 describes a submap-based approach to work around this problem. A color visualization is shown in Figure 26a, and Figure 26b shows all landmarks in sequence K shown in blue, and landmarks observed in sequence L in red. The complete map, inclusive of feature descriptors has a size of approximately 1.4GB on disk.

4.6 Scalability through Submaps

To overcome issues of scalability—memory and time—associated with optimizing the STM, I now introduce the Submap-STM, which breaks the STM up into roughly street-sized segments. With the ultimate goal of highly accurate, real-time localization in mind, a number of criteria drove the design of the Submap-STM:

- The Submap-STM must retain the same level of accuracy as the monolithic STM, without alignment errors between submaps.



(a) **(b)**

Figure 26: Landmark point cloud comprising sequences K and L with (a) actual color (b) points in blue and red from sequences K and L, respectively.

- Submap selection during localization must be unambiguous, correct and real-time.
- Vehicle pose transitions from one submap to the next during localization must be error free and real-time.
- Submaps must capture the same visibility structure as the monolithic STM. Some landmarks near the boundary of a submap may appear in multiple submaps to satisfy this criterion.
- The number of cuts between submaps should be as small as possible.

These requirements lead to few deliberate design choices. To minimize the possibility of alignment errors, as well as submap transition ambiguities during localization, submaps must not meet at intersections. Intersections are potentially complicated during localization because a decision must be made about which submap to load next. Of course this is not a problem when the route is planned ahead of time and known to the localization algorithm. Nonetheless, avoiding intersections helps to satisfy all the other design requirements. The resulting algorithm for submap creation is shown in Algorithm 4.1.

This algorithm has several nice properties: Each submap begins N meters after passing through an intersection, and it continues through the next intersection by N meters in each

Algorithm 4.1 Submap assignment of camera frames

Input: Interpolated GPS poses for stereo frames
Input: GPS coordinates of intersections
Input: Distance N
Output: submaps with pose assignments

```
submaps ← ∅                                              ▷ Initialize set of submaps
submap ← ∅                                               ▷ Initialize empty submap
for each GPS pose g do
    intersection ← findNearestIntersection(g)
    if just passed through intersection and distance(g,intersection) > N then
        submap ← submaps.getSubmap(g, intersection)      ▷ submap with same intersection origin and similar heading as g
        if submap = NULL then
            submap ← createNewSubmap(intersection)
            submaps.insertSubmap(submap)
        end if
    end if
    submap.insertPose(g)
end for
```

possible direction, containing all the poses and landmarks for all possible turning directions at that intersection. Consequently, traversal of intersections during localization is no more complicated than with the monolithic STM. Furthermore, choosing the correct submap to move into after passing through an intersection is trivial, as the map building algorithm also yields a directed graph of possible submap transitions. In addition, and this is a quite important detail, opposite directions of travel of the same street are represented in two separate submaps. In other words, submaps explicitly encode the fact that cameras facing in the opposite direction are very unlikely to observe the same landmarks, and those cameras and landmarks are members of separate submaps.

Applying this algorithm to four datasets (J,K,L,M) yields a Submap-STM containing 45 individual submaps as shown in Figure 27.

The procedure to build each Submap-STM differs from the algorithm to construct the monolithic STM in the following way. First, putative loop closures within each submap are identified from GPS, and then geometrically verified. Each submap is then individually optimized, which involves all of the camera poses and landmarks observed in this submap. This takes a few minutes per submap, as shown in Section 4.7.2. Next, the camera poses

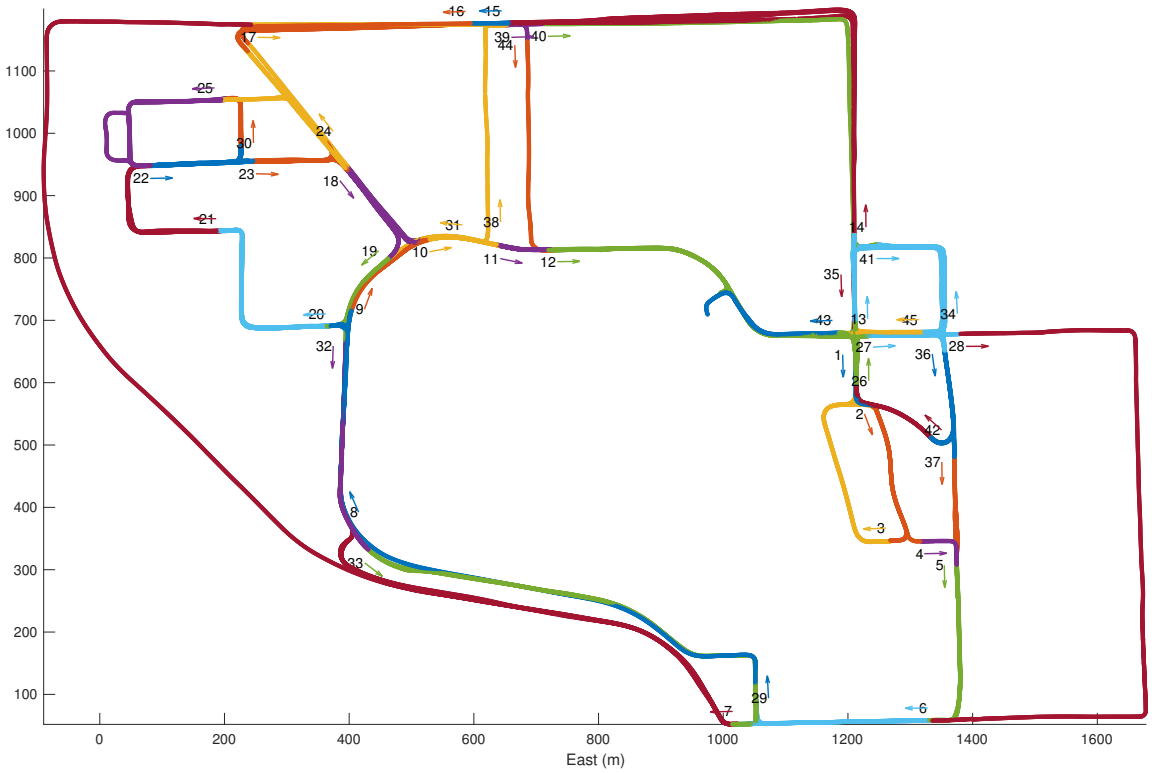


Figure 27: Optimized Submap-STM comprising 45 individual submaps. Labels and arrows indicate the starting point and direction of each submap.

from all submaps are reduced to a pose graph (including connections between submaps), and this pose graph is optimized to bring all submaps back into perfect alignment. Separator variables for each submap are then constrained in a final re-optimization step per submap.

In addition to submapping, other techniques can be used to reduce the computational complexity of optimization. For example, I have implemented a stereo factor version of the smart projection factor first introduced in [16]. This smart stereo factor uses the Schur complement trick to algebraically remove the landmarks from the actual optimization, leaving only camera poses, which means we are now optimizing a much smaller problem. However, this approach comes at a cost of additional bookkeeping overhead for each factor (one per landmark), and this overhead negated any benefits that were expected due the landmark elimination.

4.7 Results

In this section I show the results for STM and Submap-STM construction. Again, the results are organized according to the claims of the thesis. I first verify that the map building procedure yields maps of high quality as indicated by landmark reprojection errors. Next, I show that constructing the Submap-STM is fast, and finally that it is inherently robust to change.

4.7.1 High Quality

The GPS traces collected with the data are not accurate enough to serve as ground truth, as was shown in Figure 1. No other ground truth data is available. However, root mean square projection errors give a very good picture of the local consistency of the map, which is really the metric that matters most for localization. Figure 28 shows RMSE for the Submap-STM, containing four different datasets.

The individual optimized dataset trajectories which constitute the full Submap-STM are shown in Figure30.

The optimized camera poses from the first version of the STM were also used as the basis for dense reconstruction work in collaboration with Pablo Alcantarilla in [4]. A screenshot of the dense reconstruction result is shown in Figure 31.

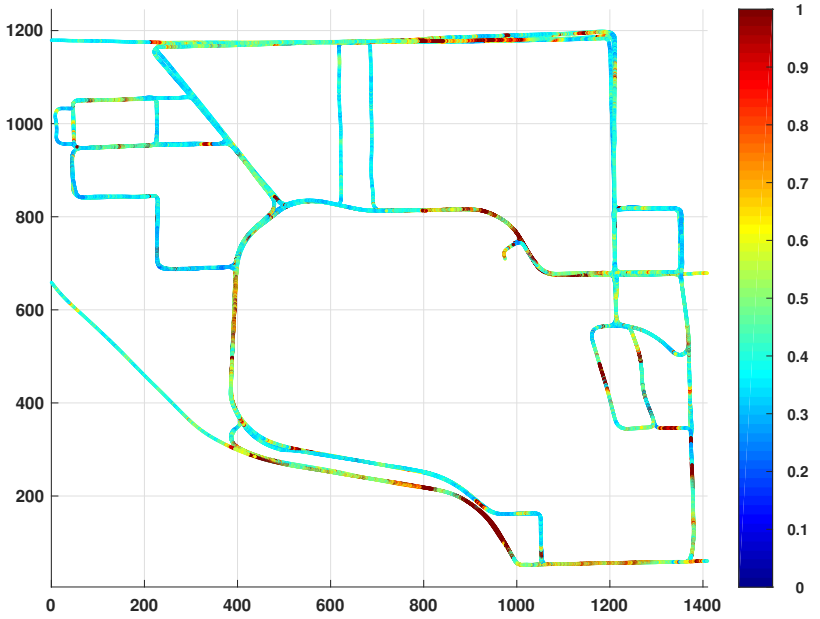


Figure 28: Root mean square projection errors per camera for the fully optimized Submap-STM.

Figure 29: Complete Submap-STM superimposed on Google Earth for scale. The map covers an area of almost $2km^2$.

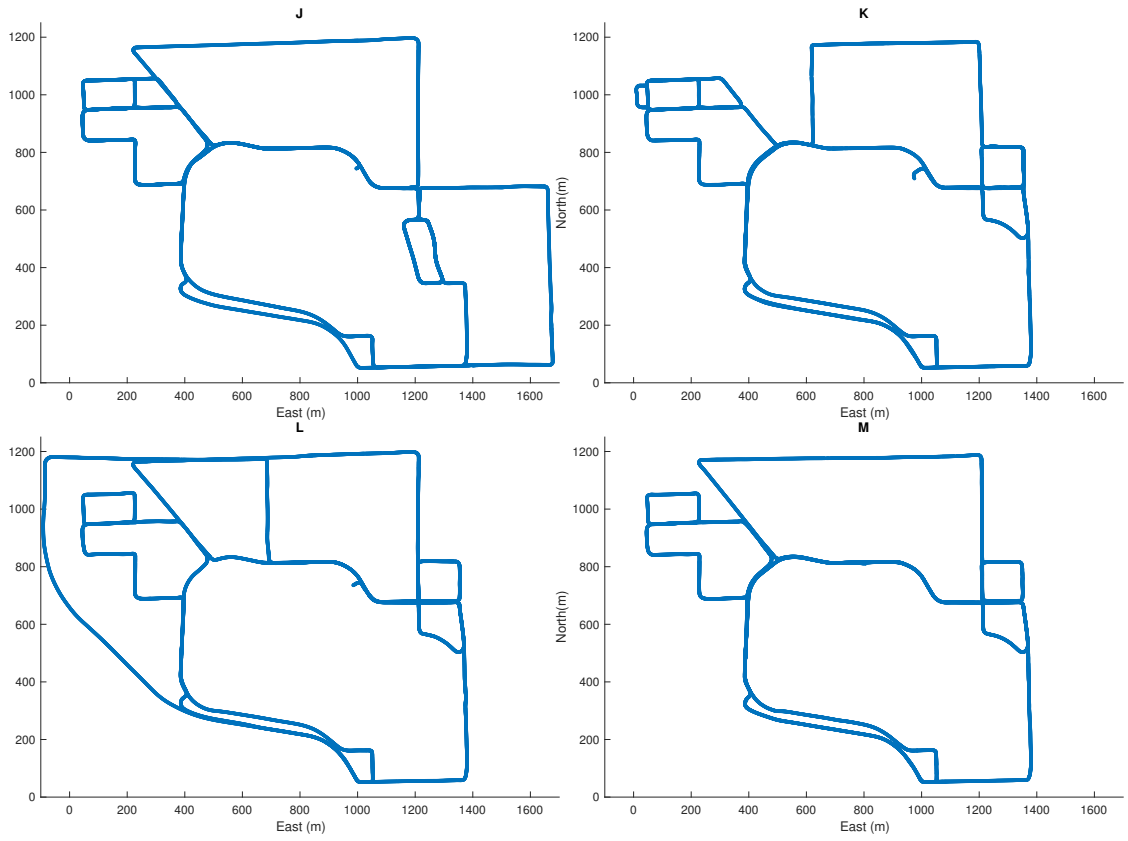


Figure 30: Datasets J K L M trajectories shown separately.

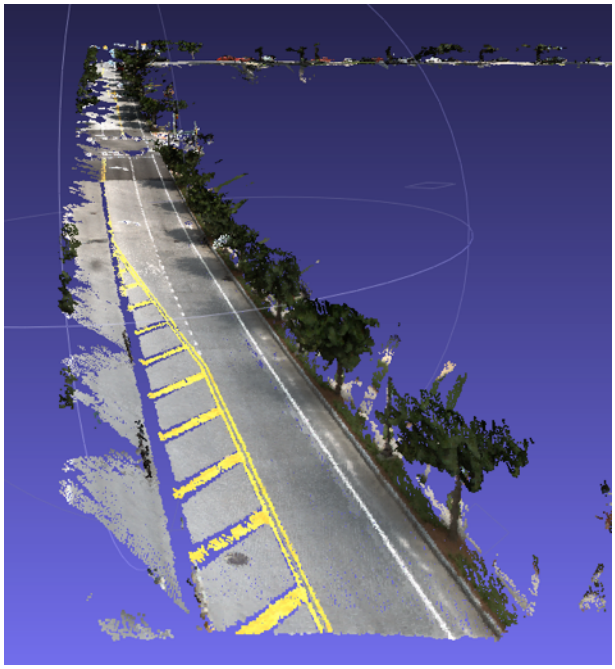


Figure 31: Dense reconstruction based on the camera poses resulting from large scale map optimization. A dense stereo algorithm is used to backproject all image pixels to 3D, and dynamic objects are then filtered out with a consistency check across a sliding window of three adjacent frames.

4.7.2 Efficient

Submap-STM optimization is much faster than optimizing a single monolithic map. Taking submap 12 as an example, which has 3812 camera poses, 465194 variables in total, and 1761264 measurements, requires 392s to optimize. In contrast, optimizing the largest possible monolithic map (which fits within 32GB of system memory without paging), takes several hours. So, submap optimization is not only faster, but it could be carried out in parallel on many machines, if desired. Monolithic map optimization, on the other hand, is not easily distributed across multiple nodes.

Other advantages of the Submap-STM are their smaller size, which reduces storage requirements.

4.7.3 Robust to Change

The map is robust to change because it incorporates landmarks observed in different data sequences collected at different times of the year. Figure 32 shows the distribution on the map, and Table 4 shows quantitative results. Figure 33 shows a zoomed in view of one city

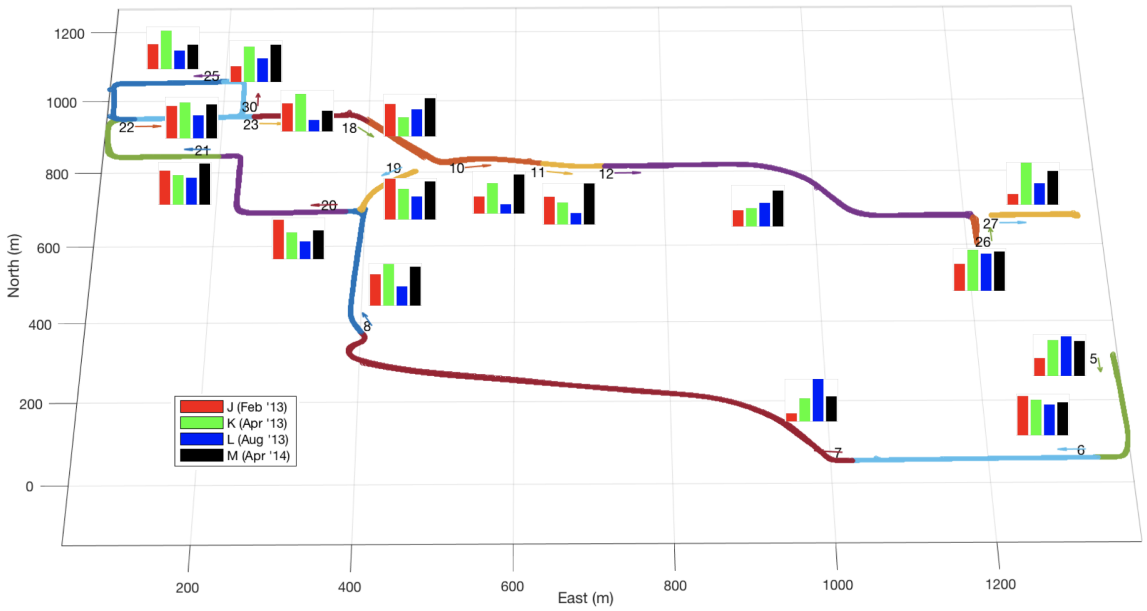


Figure 32: Landmark distribution for each of the submaps which have full dataset coverage. Each histogram represents the relative number of landmarks contributed by each of the four datasets.

Table 4: Number of landmarks observed in each of the submaps which have full dataset coverage.

Submap	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)
5	25719	51819	57260	50524
6	47343	42865	37086	39792
7	56619	163098	299965	177118
8	103325	137281	63345	128456
10	40131	71602	21881	92052
11	40065	31730	16472	59354
12	136489	153935	199199	302841
18	61064	35963	51054	71994
19	29524	22150	16677	27574
20	151901	102898	68338	110839
21	64132	55613	50743	77534
22	93755	103915	66669	98372
23	67717	90445	27557	49868
25	70258	108629	52306	68960
26	19025	28954	26259	27769
27	24933	98853	50454	79501
30	13249	29814	19942	31442
Total	1045249	1329564	1125207	1493990

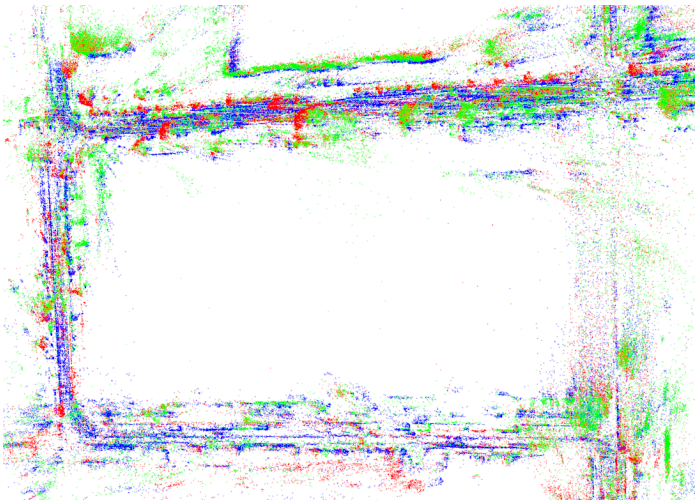


Figure 33: Closeup top-down view of STM, covering a single city block. Landmarks from three different datasets are shown in red, green, and blue.

block, with landmarks from three different datasets shown in red, green, and blue. The figure shows that no matter where the camera faces, some landmarks from each dataset should always be visible.

4.8 Summary

In this chapter I have presented the STM and Submap-STM, and procedures to build these two variants of spatio-temporal maps. The STM was built from multiple datasets, and encodes the spatio-temporal visibility of each landmark. Landmarks are annotated with observation times, as well as feature descriptors in the map. Having a good map is absolutely necessary for good localization. In evaluating the map I have verified these claims to be true:

- **High-quality** - The map built of multiple datasets is of high quality as measured by reprojection errors.
- **Efficient** - Submap-STM construction is efficient compared to monolithic map construction. Submap construction requires less memory and time.
- **Robust to change** - The spatio-temporal map is inherently robust to change because it incorporates landmarks observed at different times of year. This is demonstrated by the distribution of landmarks in the map, as well as the localization experiments in the following chapter.

The STM and Submap-STM will be used in the next chapter as part of the full localization system.

Chapter V

LOCALIZATION

In this chapter I present the algorithms which implement real-time localization in the STM. Figure 34 shows the main components of the system and the general flow of information between these components. The localization module (shown at bottom) works together with visual odometry (top left) and the Submap-STM (top left) to form the complete real-time localization system. At a high level, image features extracted by the VO pipeline are sent to the localization module along with incremental poses. The localization module uses its most recent pose estimate to retrieve the most appropriate submap from the Submap-STM module. The features extracted by VO are then matched against landmarks stored in the submap. And finally, if matching was successful, the camera pose is estimated. This global pose is then used to update the pose estimate maintained by the localization module, which is then used again in the next iteration as the process iterates.

I begin this chapter by restating the claims about the SRM-STM localization system, followed by a formal problem formulation and results. The early work in this chapter was initially presented in [9].

5.1 Claims

The claims I introduced in the introduction of this thesis are directly applicable to the localization system as a whole. Specifically, the claims are:

- **High-quality** - Localization results have high accuracy, which is crucial for autonomous driving. Accuracy is validated experimentally, with translational and rotational errors compared to the optimized poses in the map.
- **Efficient** - The Submap-STM localization algorithm provides localization in real-time, which is also a critical requirement for autonomous driving. Timings are provided for each experiment.

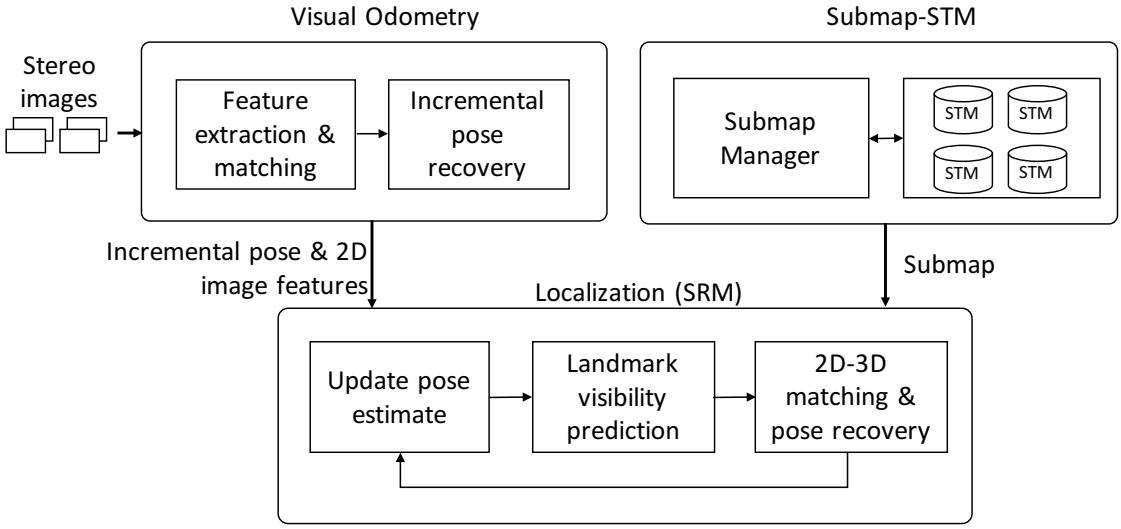


Figure 34: Localization system diagram showing three distinct components: Visual odometry, Submap-STM, and the Localization module. Visual odometry ingests stereo images and provides image features and incremental poses to the localization module. The most recent pose estimate is used to retrieve the closest submap from the Submap-STM module. The SRM then finds the landmarks most likely to be visible, which are used for localization. Finally, the SRM feeds its result back to the fixed-lag smoother where it is incorporated as a pose prior, yielding a refined pose estimate.

- **Robust to change** - The SRM-STM localization system is robust to change because it utilizes a spatio-temporal map. By storing landmarks from multiple datasets collected at different times, it enables more robust localization than a single-dataset map in the presence of appearance changes. This robustness is validated experimentally with leave-out experiments.

5.2 Problem Formulation

The goal of localization is to estimate the current vehicle position Θ_t, provided some sensor data S and some knowledge about the world. This being the case, I want to find the best estimate for the current vehicle pose $\Theta_t \in SE(3)$, given all of the sensor measurements S^t taken up to and including the current time t, and the spatio-temporal map. This thesis is focused on vision-based localization, so the sensor used for the experiments is a camera, but other sensors could be added to augment the system. Here what matters is that it includes landmarks $L \in \mathbb{R}^3$, annotated with feature descriptors and the time t at which each landmark was observed during the map-building phase.

I take, as is standard, a Maximum a Posteriori (MAP) approach to the problem, maximizing the posterior density

$$P(\Theta_t|S^t, STM) \propto L(\Theta_t; S_t, STM)P(\Theta_t|S^{t-1}, STM)$$

Above, Bayes law was applied to decompose this into a likelihood term $L(\Theta_t; S_t, STM)$, explained below, and a prior $P(\Theta_t|S^{t-1}, STM)$ that predicts the pose Θ_t at time t given past sensor readings S^{t-1} and the STM. While our technique can work with images alone, the prior can be augmented—where possible or needed—by additional sensors such as GPS, wheel odometry, etc.

The key to localization, however, is the likelihood term $L(\Theta_t; S_t, STM)$, and just like the work of many other researchers [49, 78, 38], a key ingredient of my approach is a feature-based data-association step with landmarks stored in the STM. Indeed, after extracting 2D features Z_t from the current image $I_t \in S_t$, we assume that the likelihood can be factored over features and other information

$$L(\Theta_t; S_t, STM) = L(\Theta_t; Z_t, STM)L(\Theta_t; S_t \setminus Z_t, STM)$$

and we concentrate on the former from now on.

What complicates matters is that the assignment of image measurements Z_t to landmarks in the map STM is not known. This is referred to as the data association problem. Let the variable J represent an arbitrary data association assignment as in [25], and (**??**) can then be rewritten as

$$\Theta_t^* \stackrel{\Delta}{=} \arg\max_{\Theta_t} P(\Theta_t|Z^{t-1}, STM) \sum_J P(Z_t|\Theta_t, STM, J)P(J|\Theta_t, STM)$$

where we sum over all possible data associations J. Given that the current image contains up to thousands of measurements Z_t, and the map STM has millions of landmarks, the number of possible assignments is vast. Instead, I will introduce a single best-guess correspondence between image measurements and the map. Let J^R denote the best-guess data association, and I can finally write

$$\Theta_t^* \approx \arg\max_{\Theta_t} P(\Theta_t|Z^{t-1}, STM)P(Z_t|\Theta_t, STM, J^R)P(J^R|\Theta_t, STM)$$

To find J^R I employ a well known algorithm, RANSAC [12], which from a set of hypothetical correspondences (also called putatives) determines an inlier set of correspondences. Formally, the goal of RANSAC is to obtain a set of inlier data points I in a set of putatives P. The algorithm randomly selects a minimal subset $S \subseteq P$, where the size of S is the minimum required to compute a desired model $M1$, e.g. in the case of camera pose estimation, $|S| = 3$. Given model $M1$ the set of inliers I is computed. If $|I|$ exceeds a predetermined threshold the algorithm terminates and a refined model $M1^*$ is computed from all inliers I. Otherwise, a new subset is randomly selected, and the algorithm iterates until a good inlier set has been found, or a maximum number of iterations is reached, thus indicating failure.

Finding a good set of putatives is itself a really hard problem when the map is large, i.e. it contains millions of landmarks, most of which are not observable from any given location. Measurements detected in the current image could be matched to landmarks in many different places in the map using a naive appearance-based data association approach. Given that outdoor scenes experience seasonal and weather-related appearance changes, the situation is complicated further.

However, for the purpose of this research I assume that the camera is moving smoothly through the map, i.e. it is mounted to a driving vehicle, and not observing different map locations at random. This assumption simplifies the data association considerably, as I do not need to consider the entire map to build the set of putatives P. Given the previously estimated pose Θ_{t-1}, the data association problem is reduced to finding a set of putatives from the landmarks which are in the vicinity of Θ_{t-1}, and which are therefore likely to be visible from Θ_t.

I build on the landmark visibility prediction framework introduced in [6, 5] to solve the data-association problem in a tractable way. In light of the STM having many millions of landmarks, it is important to only attempt matching against landmarks which are likely to be visible in the current image frame. The key idea here is that frames which were taken at camera poses X which were nearby the current pose prior $P(\Theta_t|S^{t-1}, STM)$, and also facing in roughly the same direction, are likely to have observed a similar set of landmarks L_v.

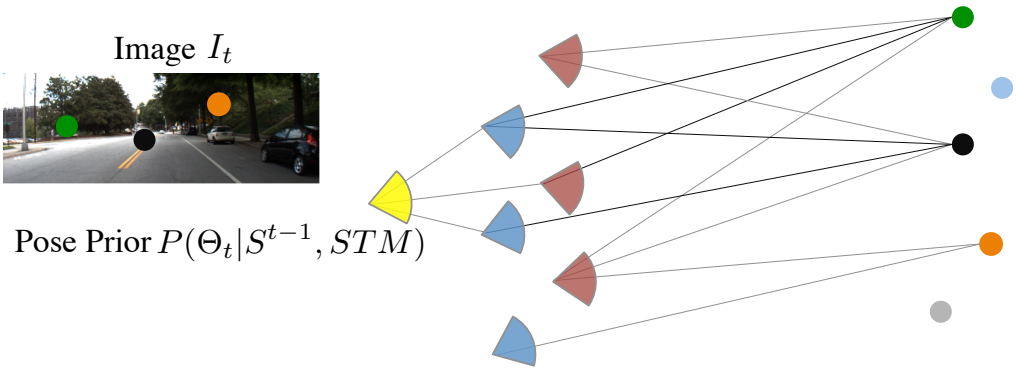


Figure 35: Cameras from two different sequences observe a set of landmarks. The lines represent the visibility graph encoded in the STM. Given the previous pose estimate Θ_{t-1}, and a set of image measurements Z_t, landmarks which are likely to be visible can be retrieved from the map.

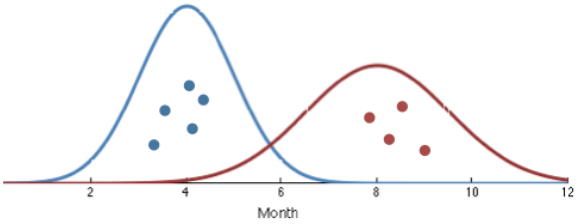


Figure 36: Two clusters of features are most likely to be observed in April, and August, respectively.

$$P(v_j|\Theta_t) \approx \frac{\sum_{i=1}^{T} k\left(\Theta_t, x_i^{v_j=true}\right)}{\sum_{i=1}^{T} k\left(\Theta_t, x_i\right)} \tag{7}$$

In this thesis I also explore whether geometric visibility can be augmented with a time-dependent term $P(v_j|\Theta_t, STM)$, operating under the assumption that landmarks which were observed at similar times of year might also be effective predictors of visibility. See Figure 36. This "visibility score" is then computed as

$$P(v_j|\Theta_t, STM) \approx P(v_j|t, STM)\frac{\sum_{i=1}^{T} k\left(\Theta_t, x_i^{v_j=true}\right)}{\sum_{i=1}^{T} k\left(\Theta_t, x_i\right)} \tag{8}$$

where in practice only the T nearest camera poses are considered, as visibility of very distant landmarks can safely be assumed to be zero. v_j is the visibility of landmark l_j. $P(v_j|t, STM)$ is the probability of a landmark to be visible given the current time, explained in more detail in the following section. $k(.)$ is a function which measures the similarity (distance) between two camera poses:

$$k\left(\overrightarrow{\theta}_t, \overrightarrow{\theta}_i\right) = exp\left(-\left\|\mathbf{A}\left(\overrightarrow{\theta}_t - \overrightarrow{\theta}_i\right)\right\|_2\right) \tag{9}$$

where $\overrightarrow{\theta}$ are vectors encoding the translation and rotation of each of the camera poses in question. **A** is learned from the existing 3D reconstruction, and is obtained using the visibility learning framework in [6]. Intuitively speaking, this function optimally weights the rotation & translation distances between two poses, as these quantities are scaled very differently.

Figure 35 shows a prior camera pose, and some nearby map camera poses, which in turn are connected to some landmarks via the visibility graph stored in the map. A landmark is connected to cameras in multiple data sequences if the landmark was part of a loop closure.

As each landmark, and consequently each putative now has a visibility score associated with it, I can exploit this information during pose recovery. I use PROSAC [18] instead of RANSAC, which assumes that P is sorted according to some quality criterion q:

$$p_i, p_j \in P_N : i < j \Rightarrow q(p_i) \geq q(p_j)$$

The PROSAC algorithm begins by drawing samples from the landmarks with highest predicted visibility (i.e. during the first iteration $S = \{p_0, p_1, p_2\}$), and progressively increases the sample pool until termination. The algorithm makes the assumption that samples of high quality putatives are more likely to yield a valid model, and hence allows termination much sooner than RANSAC.

I take the view that the visibility score meets this requirement, in the same way that it has been shown that SIFT match scores work well in this context.

As mentioned previously, the STM annotates each landmark with observation times, which I can use during localization to compute the time-dependent visibility probability $P(v_j|t, STM)$. We make the assumption that landmarks which have previously been observed at a particular time of year (season), are more likely to be re-observed during similar conditions. Hence, the time-dependent visibility prediction term $P(v_j|t, STM)$ is computed from distance in time. For example, we need to compute the absolute distance between months, and wish to assign a score in the range of [0,1]. Neglecting wrap-around due to subtraction, we have $1 - |(t_1 - t_2)/6|$. This equation gives us a visibility score of 1 for any given STM landmark if it was observed in the same month, i.e. $t_1 = t_2$, and 0 if it was

observed 6 months apart, e.g. $t_1 = 7$ (July), and $t_2 = 1$ (January.)

Given L_v, the standard approach is followed to compute a pose estimate: Detect features in the current stereo pair, match and verify with RANSAC/PROSAC.

In practice, some steps can be taken to further speed up the algorithm described above. Computing the visibility score with respect to all poses X can be costly for large STM. Instead, I pre-prune the set of poses, and only compute the visibility for poses that fall within a small neighborhood of the predicted pose.

5.3 Methodology and Implementation Details

In this section I describe the algorithm and implementation details of the localization module. As shown in Figure 34, there are essentially four steps to iterate over. Pose prediction, and the pose update step are handled by a fixed-lag smoother, which maintains the pose estimate of the camera. The SRM, on the other hand, interfaces with the STM to compute a pose estimate given the STM and the most recent features from VO.

5.3.1 Snapshot Recognition Module

The heavy lifting during localization is done by the SRM, which is responsible for computing an absolute global pose given the Submap-STM and a pose prior. The latest pose from the fixed-lag smoother is used as pose prior to select the most appropriate submap. The SRM then uses this pose prior to compute the set of most likely visible landmarks. The STM contains the visibility graph which was uncovered during map building, so the algorithm begins by finding the set of cameras which are nearest to the pose prior, and then collects landmarks which were observed by those nearest cameras. The search for nearest cameras is bounded by the size of the map, so this search does not affect scalability. Once the set of landmarks is identified, I compute the geometric (and/or seasonal) visibility score for each of the landmarks. This is followed by 2D-3D matching between the VO features for the current frame and the landmarks, and the pose is recovered using RANSAC/PROSAC with the three-point algorithm. If a pose with sufficient number of inliers is found, the pose is provided to the fixed-lag smoother, which means it is added as a pose prior. Without this prior the fixed-lag smoother would be reduced to outputting the VO solution.

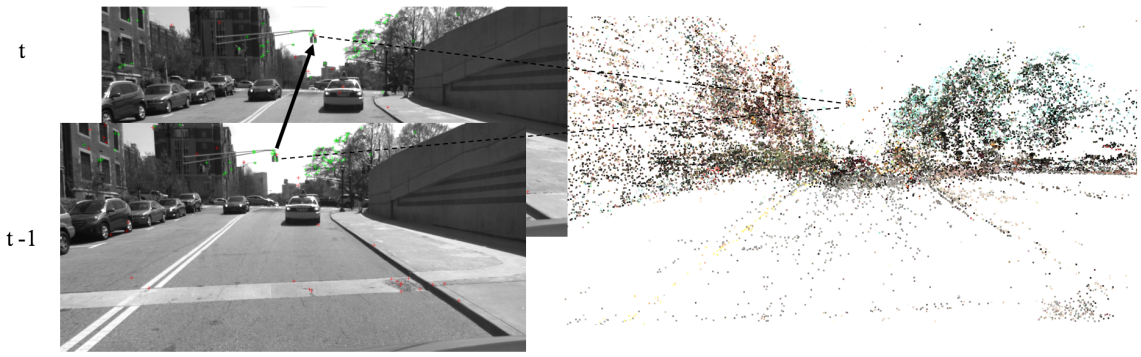


Figure 37: Inlier propagation: By propagating inliers from frame x_{t-1} to frame x_t, these features and landmarks are removed from the feature association step which is very time consuming.

5.3.2 Inlier Propagation

Localization with the SRM is very fast because of landmark visibility prediction. However, some time can also be saved by taking advantage of the fact that we are localizing a continuous stream of images, without reducing the number of landmarks. As explained in Section 3.3, image features are tracked from frame to frame by the visual odometry algorithm. During localization each of these 2D features is matched to a 3D landmark for pose recovery. Now, instead of starting this process from scratch for each frame, I propagate the 2D-3D inliers from one frame to the next. This means that the involved 2D features and 3D landmarks are removed from the regular descriptor matching at this step. Figure 37 shows an example of the inlier propagation idea. The 2D features on the traffic signal appear in two consecutive images, and remain associated with the same 3D landmark from the STM without having to redo the feature matching step.

5.4 Results for Monolithic SRM-STM

I first show results for the monolithic version of the SRM-STM in the following two sections.

5.4.1 Two Data Sequences

The results for two data sequences were first reported in [9]. The purpose of this experiment was to ascertain whether combining several datasets into a single map was at all useful, or if this had entirely detrimental effects on localization performance. Table 5 shows the localization performance of the three sequences with respect to a map constructed from

Table 5: Number of successfully localized frames of sequences F, K, L against maps created from sequence K alone, and from sequences K and L.

Map	F	K	L	Total
K	5335	22969	10540	38844
K+L	4417	22819	21245	48481

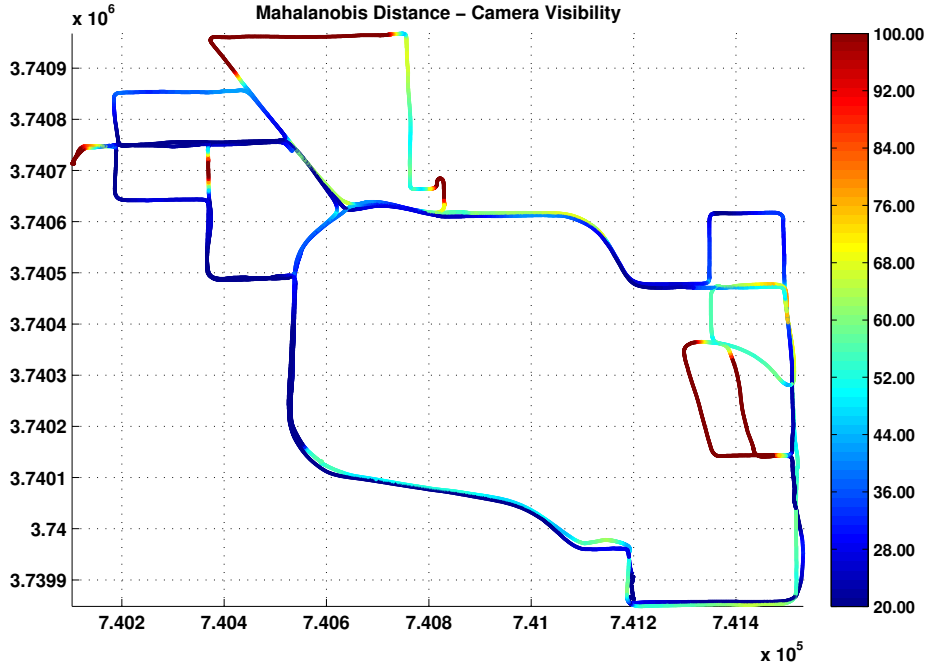


Figure 38: Camera visibility scores (Mahalanobis distance) per GPS query pose for sequence F with respect to the full database. Lower (blue) is better. Streets which were not covered by the database, or which were traveled in the opposite direction have a large distance (red).

sequence K alone vs. a map constructed from K+L.

Figure 38 shows a visualization of the smallest visibility distance metric for each query pose. Note that to the east there is a loop which was not covered in the map, and therefore has a very large visibility distance (deep red).

5.4.2 Three Data Sequences

For this experiment, the STM contains landmarks observed from an overlapping area of datasets J, K & L shown in Table 6. All areas of this region of interest (ROI) are visited by our vehicle in at least two sequences. Frames taken outside of this ROI are ignored completely.

This STM contains 34509 camera poses, and over 2.3 million landmarks with SIFT

Table 6: Datasets used in experiments. The "localized" column represents the number of frames which were able to localize with respect to the STM.

Label	Date	ROI Frames	Localized	Resolution
F	Sep 11, 2012	11377	4280 (37.6%)	1380×480
J	Feb 2, 2013	12304	11493 (93.4%)	1380×480
K	Apr 2, 2013	11327	10833 (95.6%)	1384×680
L	Aug 1, 2013	10878	10686 (98.2%)	1384×680

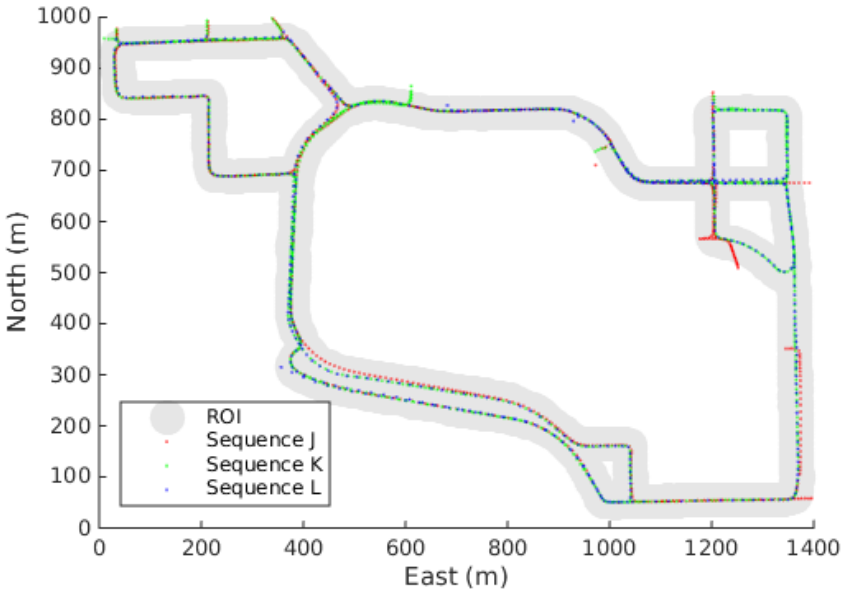


Figure 39: STM camera poses after full bundle adjustment.

feature descriptors. This map takes 1.4GB of storage when serialized out to disk. Optimized camera trajectories are shown in Figure 39.

When it comes to localization results from the STM, I first show that my approach serves to relocalize the images that comprise the map itself very well. I use the noisy GPS-INS readings as the pose prior, retrieve all of the landmarks observed by the 15 nearest cameras in the STM, and require an inlier ratio of at least 10%. Localization results for this scenario are shown in Figure 40.

Next, I evaluate the localization performance on a sequence which was not used to build the STM. This dataset -F- was collected well before the others, and the appearance of some streets has changed significantly. Even so, localization rate of 37.6% was achieved.

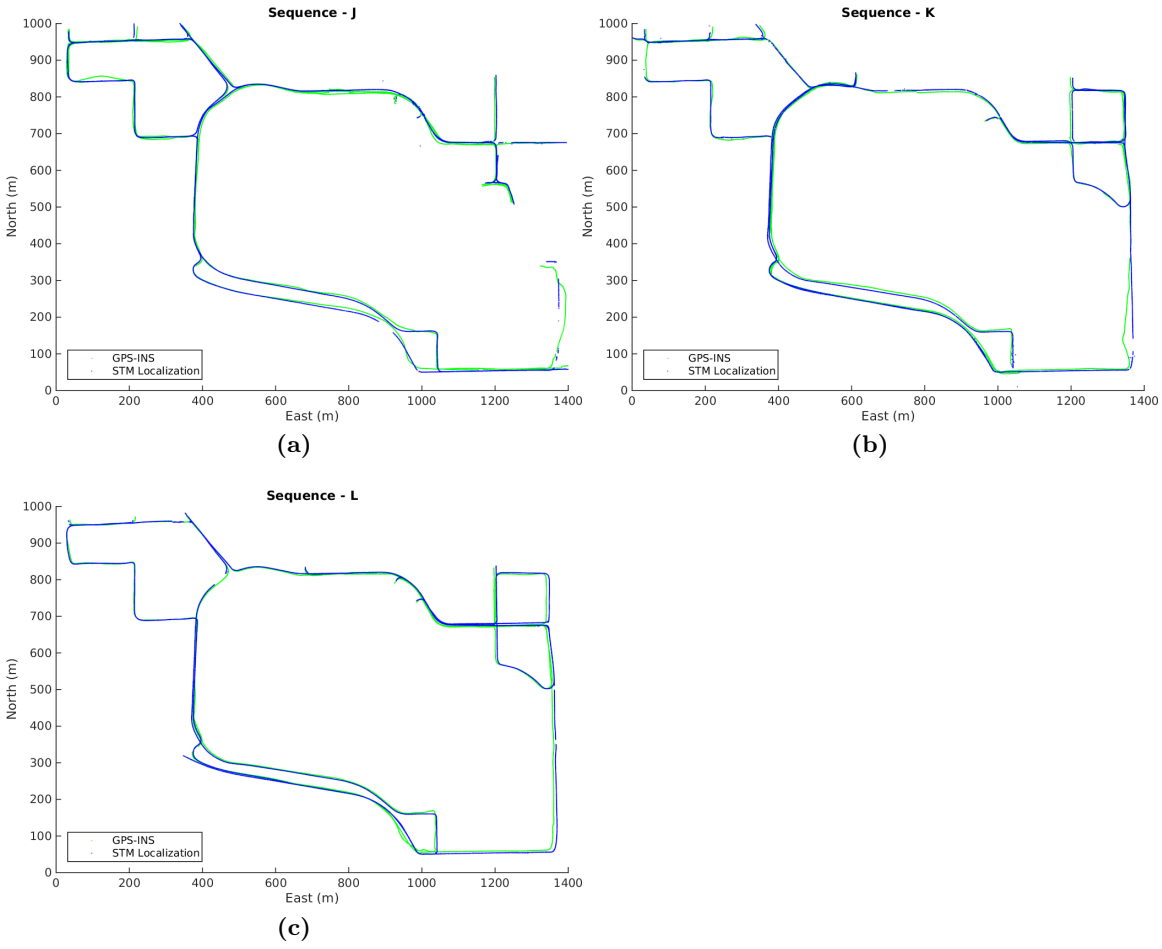


Figure 40: STM Localization Results for the images in the three sequences which contributed to the STM. The GPS-INS pose priors are shown in green, and the localization result in blue.

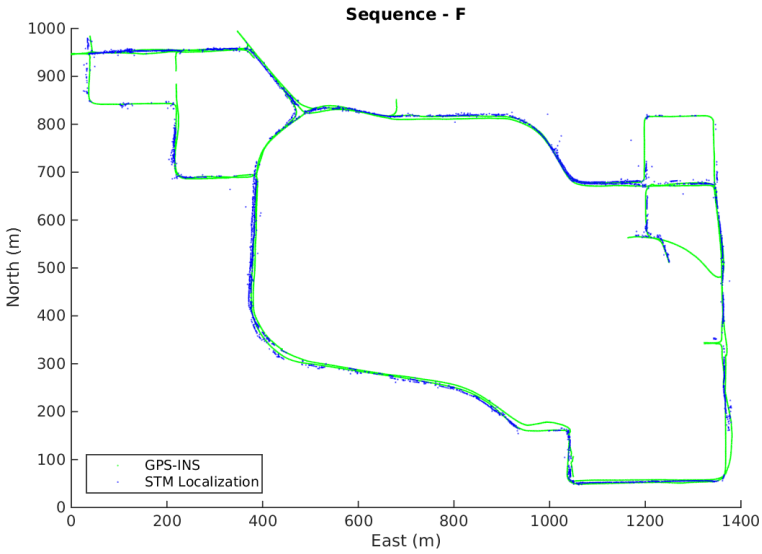


Figure 41: Localization of sequence F in the STM. Pose priors are shown in green, and localization results in blue.

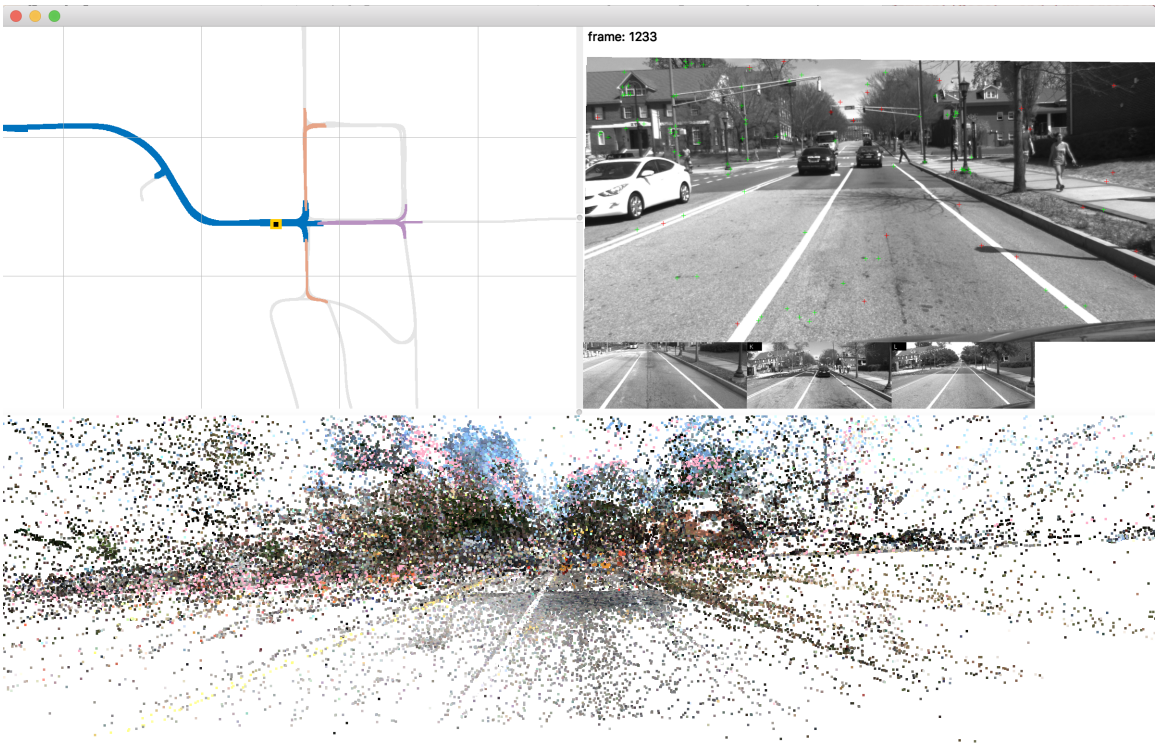


Figure 42: Localization GUI developed to monitor the progress of the vehicle through the Submap-STM. Top left: Zoomed in top-down view of the map, with vehicle location shown in black, nearby cameras from the map are shown in yellow, and the active Submap-STM is shown in blue. As the eastwardly traveling car comes up to the intersection, three possible follow-on submaps are highlighted in light orange and purple. Top right: Current camera frame, with thumbnail of the closest image from each dataset in the map. Bottom: Sparse point cloud of all 3D landmarks in the map.

5.5 Localization in the Submap SRM-STM

Localization within the Submap SRM-STM requires an additional layer to handle the loading and unloading of submaps, and the decision-making about which submap to load. As the submaps are non-overlapping and unidirectional, only one must ever be used at one time.

There are two operational modes: A "lost" mode which is used at the beginning, and a "locked-in" mode which is used for normal operation.

In the "lost" mode the pose prior—at startup this comes from GPS—is used to find the Submap-STM which comes closest, and is facing most nearly in the same direction as the pose prior. There are generally two submaps covering each street: one in each direction of travel. The selected submap is then used to make a localization attempt. If successful, this becomes the active submap, and we are now in "locked" mode. Otherwise, the process

repeats at the next frame.

In "locked-in" mode the choice of submap is restricted to the currently active submap, or any of the allowable successor submaps, according to the directed submap graph. Of these, whichever submap is closest to the pose prior is chosen as the active submap. The algorithm for submap selection and localization is shown in Algorithm 5.1.

Algorithm 5.1 localizeSubmapSrmStm: Algorithm for Submap-STM selection during localization.

Input: Pose Prior p
Input: Last used submap *activeSubmap*
Input: Directed graph of submap transitions $submapTransitions$
Input: Stereo features for current frame $stereoFeatures$
Output: New active submap $activeSubmapStm$
Output: Pose estimate $submapSrmStmPose$

```
submapSrmStmPose ← ∅                                   ▷ Initialize empty pose
if activeSubmapStm = NULL then
    activeSubmapStm ← getNearestSubmapWithSimilarHeading(p)
    submapSrmStmPose ← activeSubmapStm.localize(p, stereoFeatures)
    if submapSrmStmPose = NULL then
        activeSubmapStm ← ∅                    ▷ Reset, and try again on next frame
    end if
else
    permissibleSubmapStm ← submapTransitions.validTransitionsFrom(activeSubmapStm)
    permissibleSubmapStm ← permissibleSubmapStm ∪ activeSubmapStm
    activeSubmapStm = selectClosestToPrior(p, permissibleSubmapStm)
    submapSrmStmPose ← activeSubmapStm.localize(p, stereoFeatures)
end if
```

5.6 Results for Submap SRM-STM

I evaluate the performance of the Submap-SRM-STM localization algorithms in terms of the claims. I first show that the STM supports very high quality localization with four datasets, and leave-one-out cross-validation experiments in Section 5.6.1. This is followed by validation of real-time performance, and a set of experiments that show how the speed varies with the number of nearby cameras and landmarks in Section 5.6.2. I also present an inlier propagation algorithm which can further reduce the time required during data association and pose recovery. Finally, I validate robustness to seasonal change with experiments comparing the localization performance of the Submap-STM against results with just a single dataset in Section 5.6.3. I conclude with a head to head comparison of geometric and

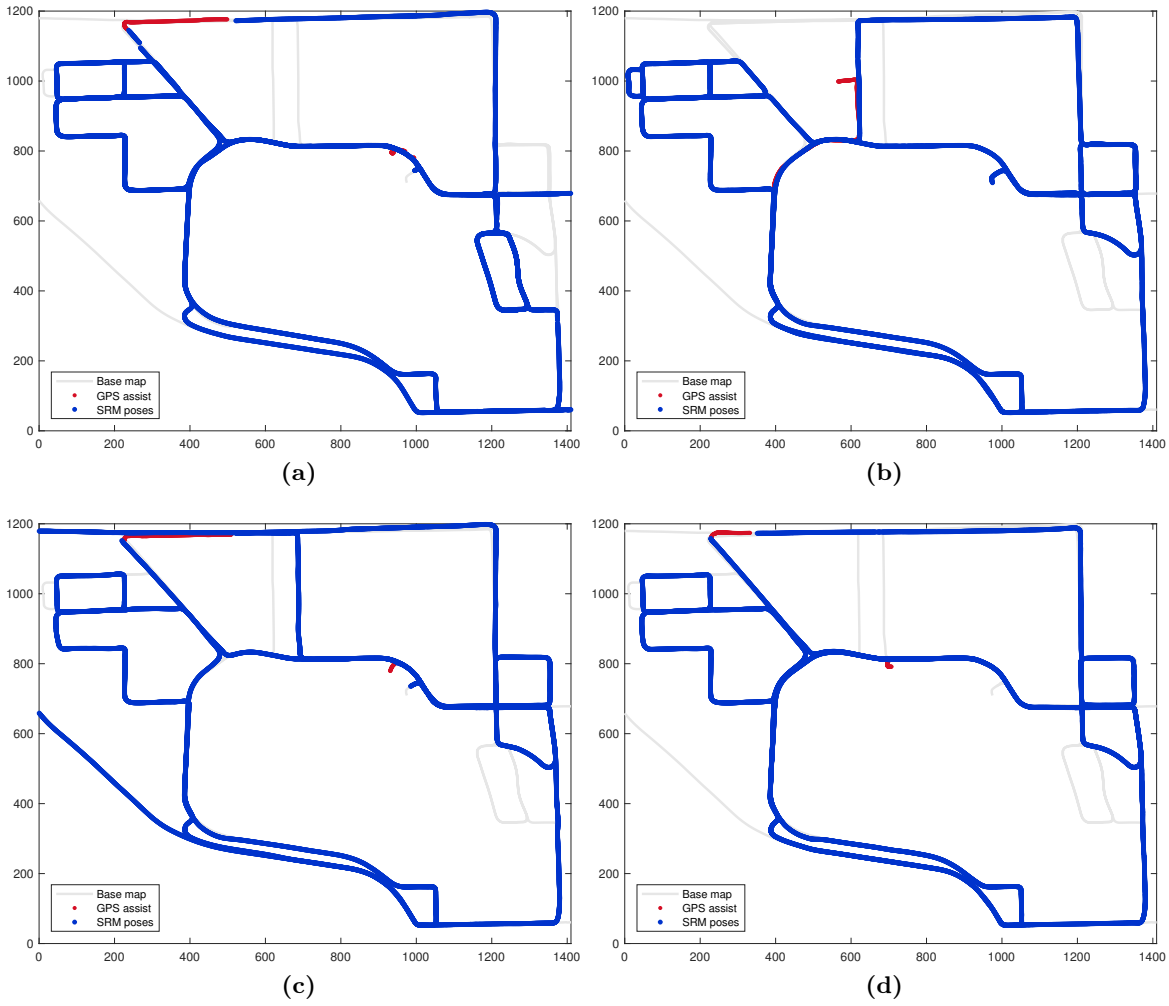


Figure 43: Complete localization results, including non-overlapping areas of datasets. Poses returned by the SRM are shown in blue, and frames where a GPS-assist prior was used are shown in red. The complete map is shown in light gray for reference.

seasonal visibility.

5.6.1 High Quality

As shown in the experiments in this chapter, translational errors with respect to the STM are very good, from an average low of 8 cm, up to a maximum error of 0.56 m. Rotational errors are very low across all experiments, with a low of 0.017 deg in the ideal case, and an average up to 0.2 degrees otherwise.

Table 7: Number and percentages of frames which were successfully localized.

	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)
Localized frames	17023	13323	14809	13593
Query frames	17442	14014	15229	13841
Localized frames (%)	97.6	95.1	97.3	98.2
Frames in map	16959	13685	14768	13733

5.6.1.1 Basic Localization

Before moving on to more interesting leave-one-out cross-validation experiments, it is necessary to establish a baseline showing the maximum performance that is attainable. After all, one might reasonably expect that the addition of several datasets into a single metric map could actually degrade localization performance. I start out with the most trivial localization experiment where each sequence contained in the map is localized against the map containing all four datasets. Geometric visibility prediction is used in this experiment to retrieve landmarks from the map, with the number of nearby cameras fixed at 15. RANSAC is used for pose recovery, which means that the visibility score is not used for pose estimation itself. Ideally we can expect localization recall of 100% in this scenario. Quantitative results are shown in Table 7. Note that the query dataset has slightly more frames than are contained in the map, and the map does not have perfect coverage, meaning that the data collection routes varied and do not fully overlap. The complete map contains data from 59145 individual stereo frames. Figure 43 shows successful localizations in blue, and areas where GPS was used as the pose prior in red. GPS assistance is always used at the beginning to bootstrap the localization module, and again whenever the SRM has not been able to return a pose, as is evidenced at the northern part of trajectory J.

I focus the leave-one-out experiments in the following sections on just those roads which were traversed at least once by each of the datasets contained in the map to avoid any potential biases. Figure 44shows the relevant subset of submaps. These reduced submaps contain observations from a total of 32571 frames. The quantitative results of the same experiment as above, focused on this core area are shown in Table 8. The table shows that localization recall is almost perfect, mainly being impacted only by the use of GPS at startup. Sequence K has a harder time locking in at the beginning, hence the lower overall

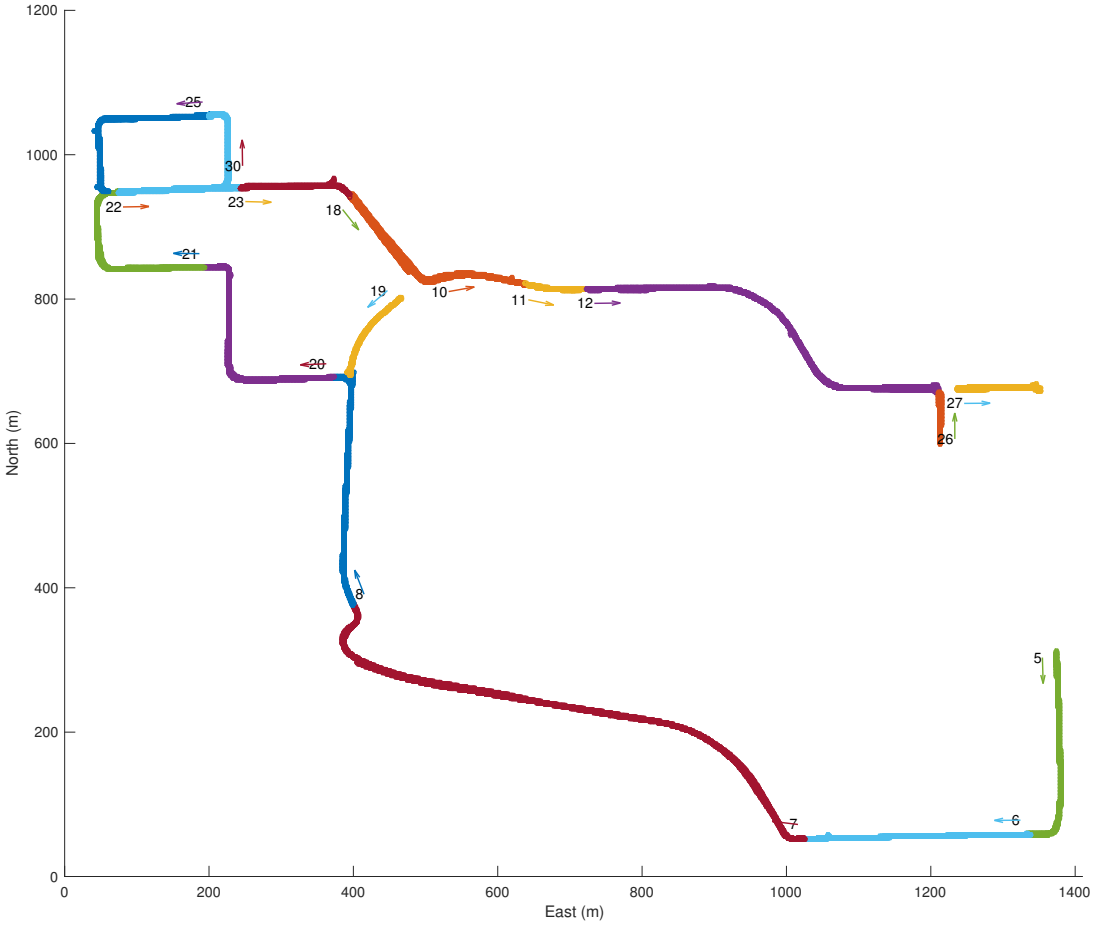


Figure 44: Subset of 17 submaps which were traversed at least once by each dataset.

Table 8: Number and percentages of frames which were successfully localized in full-coverage submaps.

	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)	Total
Localized frames	9685	7431	7050	8241	32407
Frames in ROI	9686	7590	7052	8243	32571
Localized frames (%)	99.9	97.9	99.9	99.9	99.5
Mean rotational error (deg)	0.024	0.022	0.018	0.017	0.020
Mean translational error (m)	0.083	0.082	0.167	0.081	0.103

performance of 97.9%. The table also shows mean rotational and translation error, with rotational error almost negligible, and translational error in the centimeter range. This is the ideal baseline result against which I compare all other experiments. Error histograms are shown in Figure 46

Figure 45 shows localized frames and their inlier counts. In this experiment inlier counts in the hundreds are commonly observed, but this is not typical—or expected—during leave-one out experiments. Inlier counts are impacted by the number of nearby landmarks retrieved from the Submap-STM, by the number of features detected in each frame, and finally by how many of these were successfully matched. Regions with lower inlier counts, relatively speaking, could be indicative of significant changes in the scene due to construction, etc. This may also be indicative of poor expected performance in other experiments to follow.

5.6.1.2 Leave-one-out cross-validation

To evaluate the system in a more realistic scenario, I have conducted leave-one-out experiments for each of the four datasets. Here, the same Submap-STMs as in the previous experiments are used, but any landmarks occurring only in the query sequence are suppressed during landmark retrieval. This effectively means we are localizing one sequence against a map constructed of three sequences. Again, geometric visibility prediction is used in this experiment only to retrieve landmarks from the map, with the number of nearby cameras fixed at 15. The three-point algorithm with RANSAC is used for pose recovery.

Quantitative results for this leave-one-out experiment are shown in Table 9. Paradoxically, dataset J has the lowest percentage of localized frames, but those have the lowest translational error, while dataset M has the largest percentage of localized frames, but the highest average translational error. Inlier counts per pose are shown in Figure 47, and error

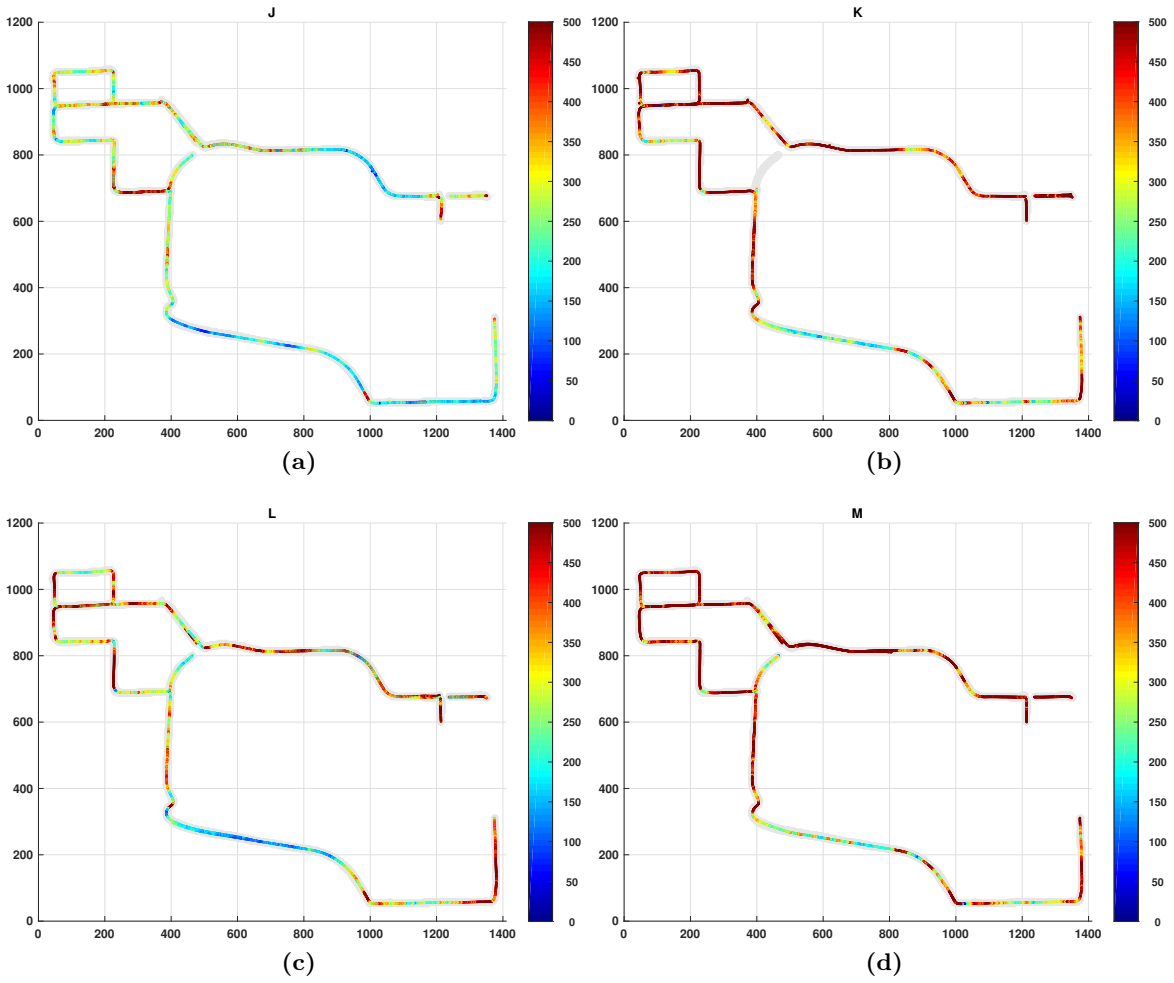


Figure 45: Localized poses in overlapping map area with inlier counts. Higher (red) is better than low (blue). The scale tops out at 500 inliers, but there are some frames which had even more inliers.

Table 9: Number and percentages of frames which were successfully localized in full-coverage submaps for leave-one-out experiment with RANSAC.

	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)	Total
Localized frames	5213	6396	5131	7114	23854
Frames in ROI	9686	7590	7052	8243	32571
Localized frames (%)	53.8	84.2	72.8	86.3	73.2
Mean rotational error (deg)	0.167	0.102	0.206	0.179	0.164
Mean translational error (m)	0.209	0.252	0.499	0.536	0.374
SRM wall time (ms)	148	126	136	123	133
SRM CPU time (ms)	791	733	753	712	747

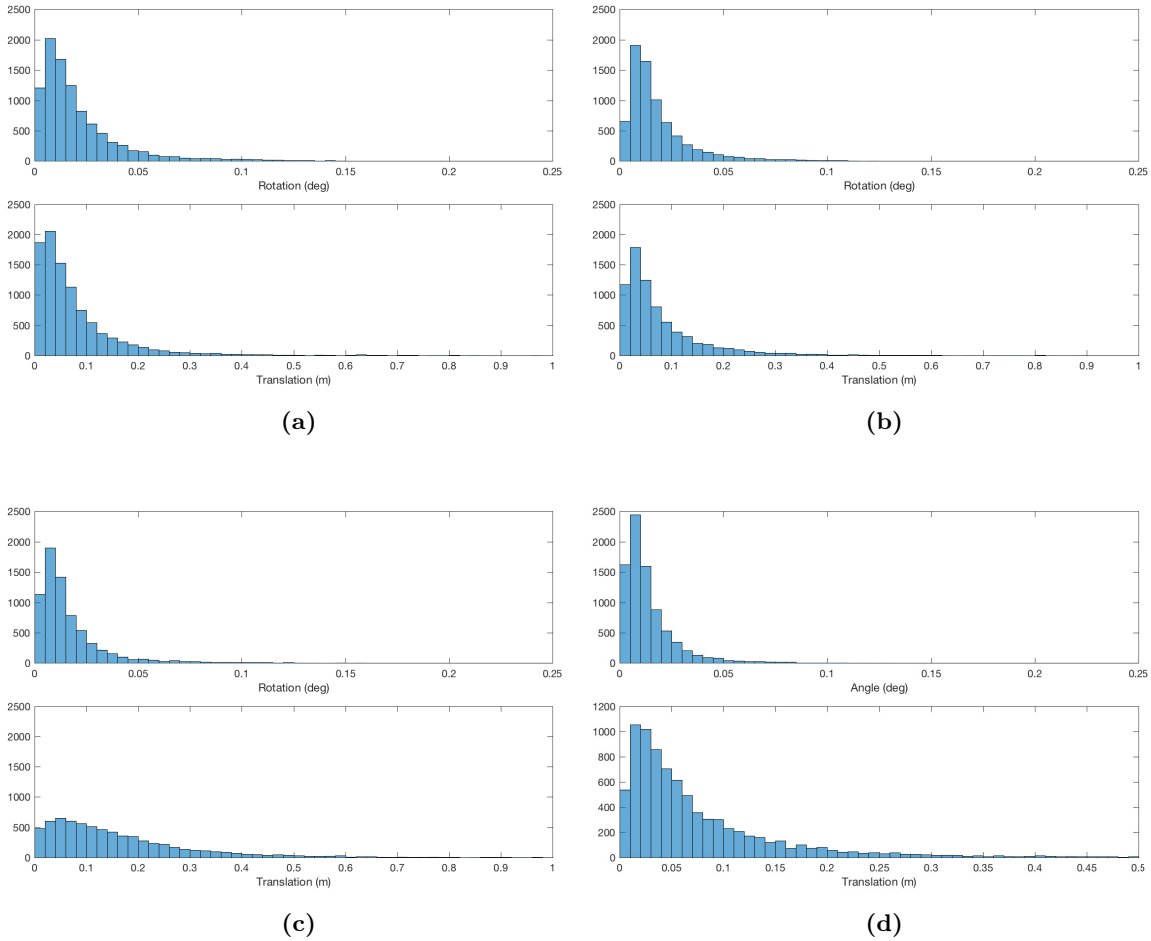


Figure 46: Error histograms showing rotational error in degrees, and translational error in meters per localized frame.

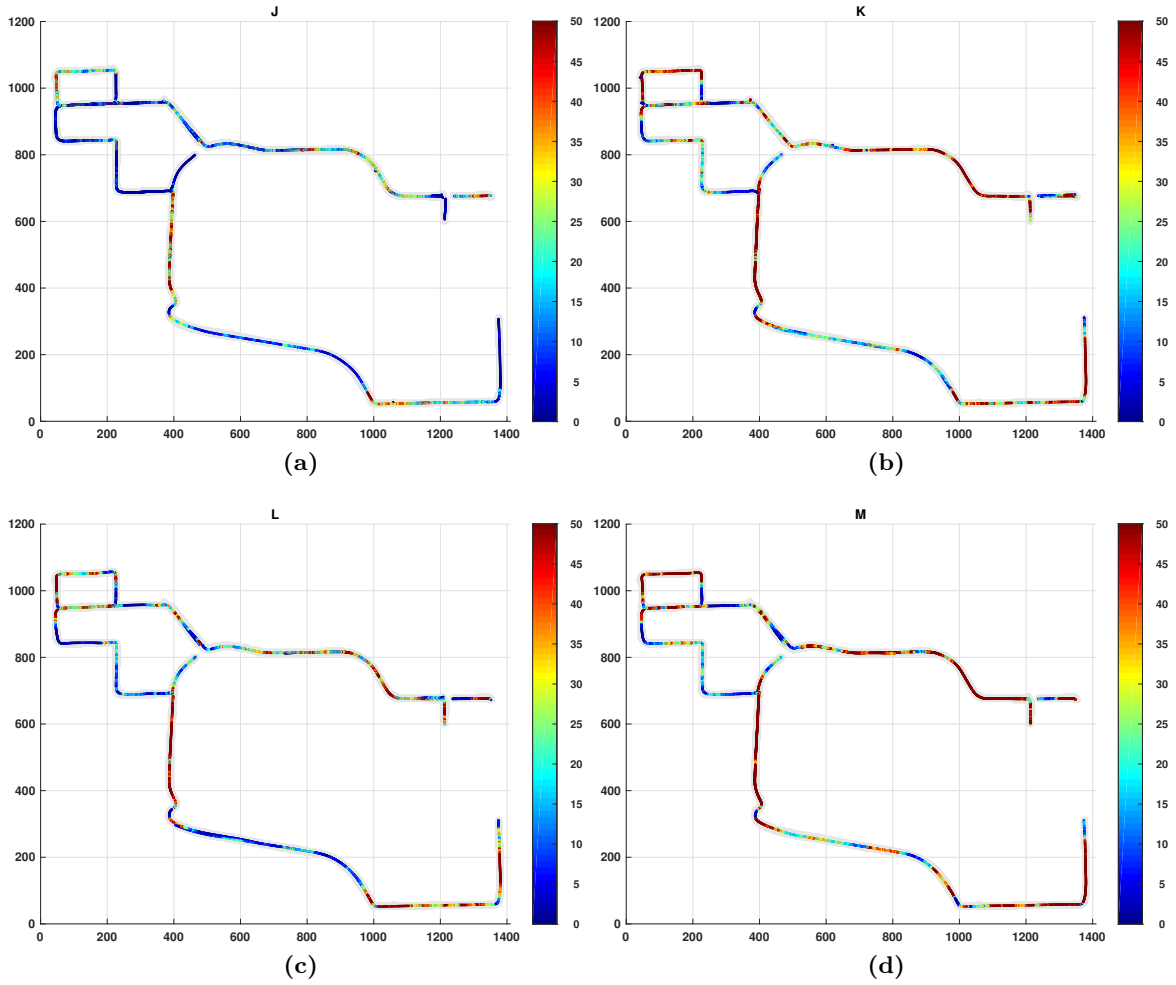


Figure 47: Inlier counts for leave-one-out experiment. Higher (red) is better than low (blue).

distributions are shown in Figure 48.

5.6.2 Efficient

The speed of the localization module varies with the number of landmarks that are retrieved, which can be tuned with several parameters. Overall timings for the SRM are are reported in Table 10. The SRM wall time includes the retrieval of nearby cameras and visible landmarks from the Submap-STM, as well as data association, and three-point pose recovery with PROSAC. Real-time performance (faster than 10fps) was only achieved after multi-threading and optimizing the data association function, which is apparent in the actual CPU time spent. All timings are obtained on an 8 core Intel i7 CPU @4Ghz.

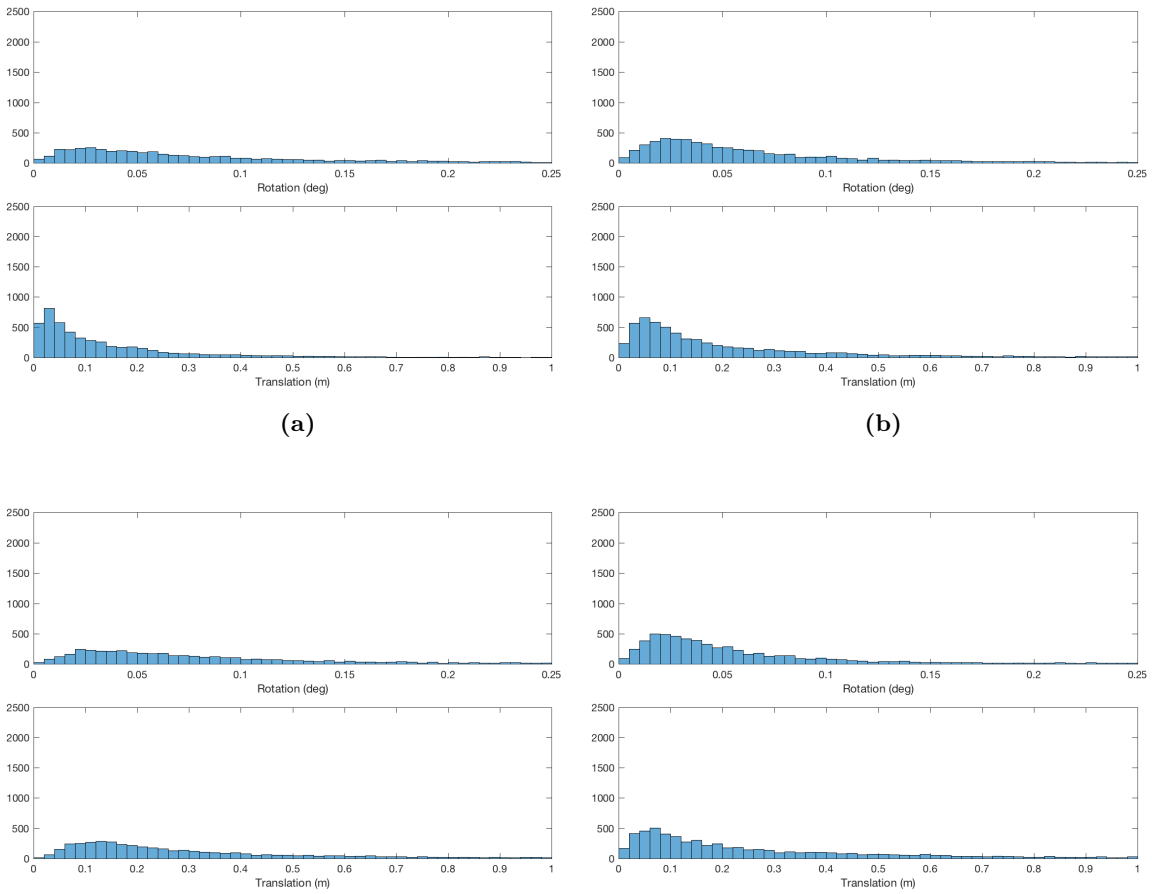


Figure 48: Error histograms showing rotational error in degrees, and translational error in meters per localized frame for the leave-one-out experiment.

Table 10: Total localization timings with 15 nearby cameras and PROSAC.

	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)	Total
Localized frames	5059	6264	4978	7041	23342
Frames in ROI	9686	7590	7052	8243	32571
Localized frames (%)	52.2	82.5	70.6	85.4	71.7
Mean rotational error (deg)	0.158	0.101	0.481	0.166	0.226
Mean translational error (m)	0.205	0.256	0.489	0.525	0.369
SRM wall time (ms)	87	98	89	92	92
SRM CPU time (ms)	449	552	497	522	505

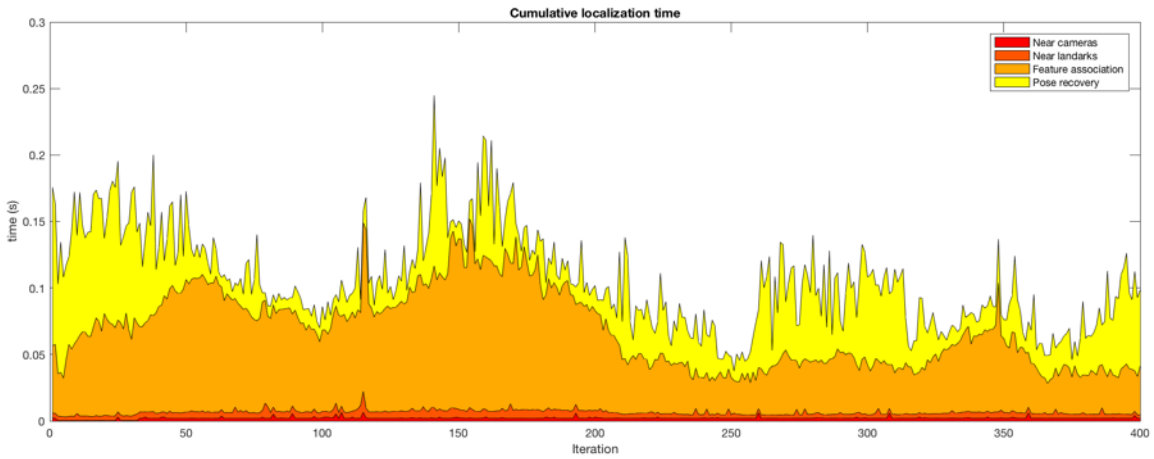


Figure 49: Cumulative time spent in the SRM. Data association is by far the most expensive part (orange), followed by the three-point algorithm and RANSAC (yellow). Camera pose and landmark/descriptor retrieval from the STM are almost negligible (red and dark red).

A breakdown of the timings for just the SRM is shown in Figure 49. Data association (orange) is by far the most expensive part, followed by the three-point algorithm and RANSAC (yellow). A negligible amount of time is spent finding the closest set of cameras given the query pose (dark red), and retrieving the visible landmarks and computing their scores (red). The camera and landmark retrieval time is bounded by the size of the Submap-STM.

5.6.2.1 Reducing the Number of Landmarks

As shown in Figure 49, data association is the most costly time consuming operation, and this is directly related to the number of features in the image, and the number of landmarks retrieved from the STM. The number of landmarks retrieved from the map is easily tuned by adjusting the number of nearby cameras from which visible landmarks are retrieved. For example, repeating the leave-one out experiment with the number of nearby cameras fixed at 5 instead of 15 frames yields a 17% speedup, with a total localization rate of 66.4% instead of 71.7%. Quantitative results for the leave-one out experiment with five nearby cameras are shown in Table 11.

Table 11: Total localization timings with 5 nearby cameras and PROSAC.

	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)	Total
Localized frames	4464	5920	4589	6661	21634
Frames in ROI	9686	7590	7052	8243	32571
Localized frames (%)	46.1	78.0	65.1	80.8	66.4
Mean rotational error (deg)	0.171	0.247	0.212	0.179	0.202
Mean translational error (m)	0.213	0.254	0.53	0.554	0.387
SRM wall time (ms)	72	78	76	76	75
SRM CPU time (ms)	373	441	415	431	415

Table 12: Total localization timings with 5 nearby cameras and PROSAC, as well as inlier propagation between consecutive frames.

	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)	Total
Localized frames	4996	6331	4990	7053	23370
Frames in ROI	9686	7590	7052	8243	32571
Localized frames (%)	51.6	83.4	70.8	85.6	71.8
Mean rotational error (deg)	0.163	0.278	0.197	0.174	0.203
Mean translational error (m)	0.222	0.284	0.55	0.597	0.413
SRM wall time (ms)	68	63	68	63	65.5
SRM CPU time (ms)	342	361	374	365	360.5

5.6.2.2 Inlier Propagation

As shown in the previous section, the total computation time spent in data association is easily tuned by adjusting the number of landmarks, with relatively minor impact on the localization rate accuracy. However, some time can also be saved by taking advantage of the fact that we are localizing a continuous stream of images, without reducing the number of landmarks. Table 12 shows quantitative results for the SRM with inlier propagation.

5.6.3 Robust to Change

To prove the claim that the STM provides increased robustness to change as compared to a map constructed from a single data collection run, I compare the leave-one-out experiment from above against leave-out-three experiment. Specifically, the total number of successful localizations of sequences KLM against map J alone was only 11916, while leave-one-out experiments for those three datasets yielded 18641 successful localizations, a 56% improvement!

Table 13: Number and percentages of frames which were successfully localized against dataset J.

	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)
Localized frames	9686	4662	3555	3699
Frames in ROI	9686	7590	7052	8243
Localized frames (%)	100	61.4	50.4	44.9
Mean rotational error (deg)	0.024	0.124	0.244	0.215
Mean translational error (m)	0.087	0.216	0.463	0.393

Table 14: Number and percentages of frames which were successfully localized in full-coverage submaps against dataset M.

	J (Feb '13)	K (Apr '13)	L (Aug '13)	M (Apr '14)
Localized frames	2070	5325	3280	8241
Frames in map	9686	7590	7052	8243
Localized frames (%)	21.3%	70.2%	46.5%	99.9%
Mean rotational error (deg)	0.488	0.165	0.390	0.018
Mean translational error (m)	0.403	0.443	0.644	0.080

5.6.3.1 Leave out three

In this experiment I leave out all but one dataset, localizing against only landmarks observed in sequence J. This is a very important experiment, because it confirms the hypothesis that a multi-dataset STM achieves better localization success rates than localizing against just a single dataset. While it may seem like an obvious conclusion that this ought to be better, there are actually several things which may cause this to turn out false: Adding additional landmarks from another sequence to the map can introduce matching ambiguities, reducing the number of available putatives. Even worse, map inconsistencies can result in lower inlier ratios, and thereby also causing localization failure. Quantitative results are shown in Table 13. An interesting trend is apparent in the localization rates, that is they decline as time goes on.

To rule out the effect on any biases I repeat this experiment, localizing only against landmarks contained in dataset M, the youngest dataset contained in the map. Results are summarized in Table 14. As expected, sequence M is localized almost perfectly, while the other sequences do quite poorly. Interestingly, sequence L does a lot worse than sequence K.

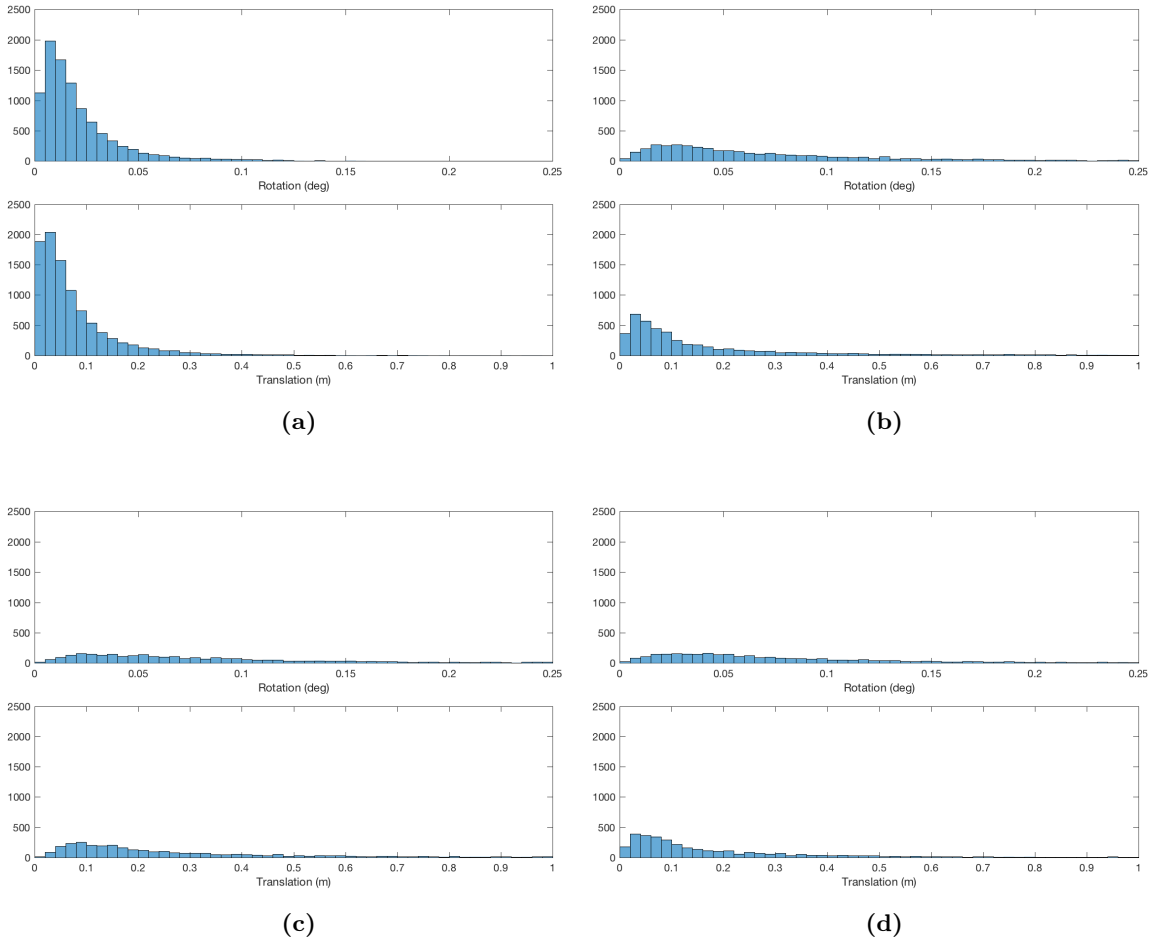


Figure 50: Error histograms showing rotational error in degrees, and translational error in meters per localized frame for the leave-one-out experiment. Note the different scale compared to the previous figure.

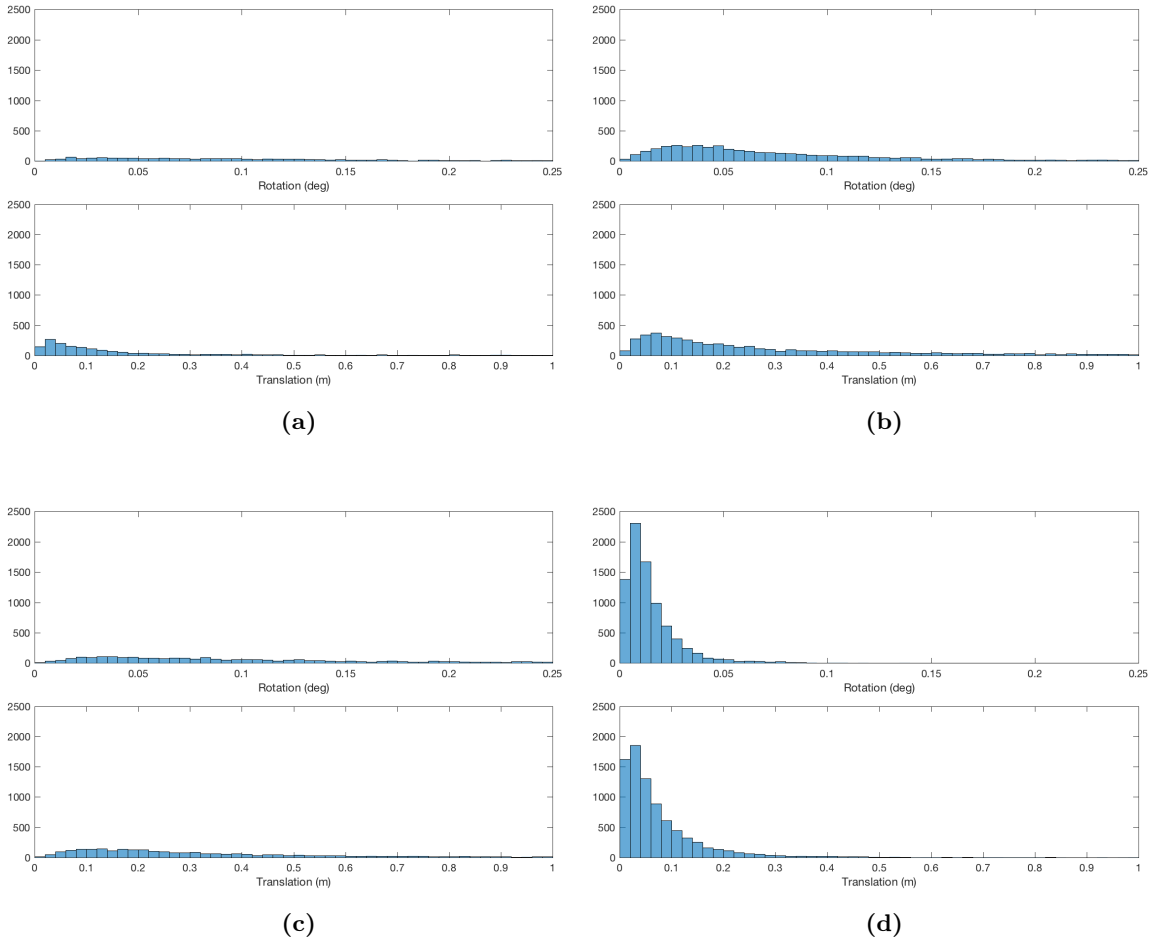


Figure 51: Error histograms showing rotational error in degrees, and translational error in meters per localized frame for the leave-one-out experiment. Note the different scale compared to the previous figure.

5.6.3.2 Geometric vs. Seasonal Visibility

In this section I evaluate the hypothesis that seasonal visibility is a good predictor of landmark visibility. As a reminder, if the hypothesis were true, it would mean that landmarks observed at a certain time of year are more likely to be reobserved at that same time of year. To evaluate the seasonal visibility hypothesis, I repeat the leave-one-out experiment shown in Section 5.6.2, except that putatives provided to PROSAC are now sorted according to the seasonal visibility score instead of the geometric visibility score. Table 15 shows the two results comparing PROSAC prioritization according to geometric and seasonal visibility side by side. As this table shows, there is only a very small difference between the two results. Looking at the overall totals, the difference in the total number of successfully localized frames is only 67, a mere 0.29%. The rotational error with seasonal visibility scoring was improved by 28%, which is largely due to the rotational error of 0.481 degrees for the geometric visibility experiment. However, it is important to note that geometric visibility alone did better for the other three datasets as far as rotational and translational errors were concerned. Overall, the translational error was 4% worse for seasonal visibility. The total wall time spent with the seasonal visibility ranking was 5% better than with the geometric visibility ranking.

While initial experiments seemed promising with regards to seasonal visibility, it has become quite clear that seasonal visibility carries far less importance than we had hoped. A number of researchers have speculated that weather and lighting are the dominant drivers of visibility, and this was finally confirmed with some hard data in [83].

5.7 Summary

In this chapter I have presented the algorithm which performs real-time localization with respect to the pre-built STM. The use of submaps and visibility prediction ensures real-time performance, even as the size of the map is scaled up. The performance and accuracy of the approach were evaluated experimentally. I have shown that the map constructed from multiple datasets performs better than a map made up of a single dataset. I have also shown that inlier propagation can be utilized to further speed up localization by taking advantage

Table 15: Number and percentages of frames which were successfully localized in full-coverage submaps for geometric visibility and seasonal visibility.

	Geometric visibility					Seasonal visibility				
	J Feb'13	K Apr'13	L Aug'13	M Apr'14	Total	J Feb'13	K Apr'13	L Aug'13	M Apr'14	Total
Localized frames	5059	6264	4978	7041	23342	5116	6252	5048	6993	23409
Frames in ROI	9686	7590	7052	8243	32571	9686	7590	7052	8243	32571
Localized frames (%)	52.2	82.5	70.6	85.4	71.7	52.8	82.3	71.6	84.8	71.8
Mean rot. error (deg)	0.158	0.101	0.481	0.166	0.226	0.164	0.104	0.210	0.167	0.161
Mean trans. error (m)	0.205	0.256	0.489	0.525	0.369	0.216	0.263	0.510	0.559	0.387
SRM wall time (ms)	87	98	89	92	92	80	93	83	91	87
SRM CPU time (ms)	449	552	497	522	505	444	566	486	549	511

of the fact that I am localizing a continuous stream of images, which means that feature associations do not have to be recomputed from scratch at every time step. In summary, the claims with respect to localization are:

- **High-quality** - Localization results of the SRM have high accuracy, which is crucial for autonomous driving applications. A highly accurate 3D map of sparse landmarks from multiple data sequences is effective as the basis for high-quality localization. Furthermore, a highly accurate stereo visual odometry system forms the backbone of both the map-building and localization modules.

- **Efficient** - The SRM-STM localization algorithm provides localization in real-time, which is a critical requirement for autonomous driving.

- **Robust to change** - The SRM-STM localization system is robust to change because it utilizes a spatio-temporal map. By storing landmarks from multiple datasets collected at different times, it enables more robust localization than a single-dataset map in the presence of appearance changes.

Chapter VI

CONCLUSIONS

In this thesis I have presented a methodology for real-time, metric localization of a moving camera in a pre-built 3D map, which is inherently robust with respect to appearance changes. This is achieved by utilizing a novel spatio-temporal map (STM) representation which is built up from multiple drives worth of data, as well as a snapshot recognition module (SRM), which efficently retrieves landmarks from the STM to perform appearance-based localization in real-time. The map encodes the visibility structure of the datasets which were captured to build the map, and this information is exploited for efficient localization. Furthermore, with an improved version called the Submap-STM, real-time performance and scalability were demonstrated.

6.1 Review of Claims

In this thesis I have shown that the spatio-temporal map (STM) and snapshot recognition modules (SRM) together are able to provide real-time, high-quality metric localization in changing environments.

In particular, I have demonstrated that the three following claims are true about the system as a whole.

- **High-quality** - Localization results of the SRM were shown to have high accuracy. A highly accurate 3D map of sparse landmarks from multiple data sequences is effective as the basis for high-quality localization. Furthermore, a highly accurate stereo visual odometry system forms the backbone of both the map-building and localization modules.

- **Real-time** - The SRM-STM localization algorithm provides localization in real-time, which is a critical requirement for autonomous driving.

- **Robust to change** - The SRM-STM localization system is robust to change because it

utilizes a spatio-temporal map. By storing landmarks from multiple datasets collected at different times, it enables more robust localization than a single-dataset map in the presence of appearance changes.

6.2 Future Work

While I have shown in this thesis that building maps from multiple datasets can be used to achieve reliable localization results in changing environments, numerous challenges remain to be addressed. For instance, one problem I did not address in this work at all is that of long-term map maintenance. It clearly does not make sense to continually grow the map as more observations are made, but identifying which landmarks should be removed as time goes on is not trivial. Similarly, algorithms are needed to decide when enough landmarks have been added to the map, as an alternative to adding all landmarks which were observed from a minimum number of images, as was the case in this work.

The experiments shown in this thesis were all conducted in daytime. However, for this work to be truly useful it must work across all manner of lighting conditions, including nighttime. This presents a serious problem for the map-building phase, as matching of landmarks across such drastic illumination differences is extremely challenging. Other sensors, such as LIDAR might prove useful in this area.

The lack of reliable ground truth makes it difficult to fairly evaluate large-scale localization results. In this thesis I used the optimized map as the basis for localization accuracy evaluation, since this was much better than GPS. Large-scale, long-term datasets across seasons with high-quality ground truth derived from other sensors are needed.

Finally, at the onset of this work we had hypothesized that in outdoor scenarios, landmark visibility would follow a cyclical pattern, but in reality, the data did not support this hypothesis. In fact, we now know, as recent research has shown, that lighting and other effects completely dominate any seasonal effects.

REFERENCES

[1] Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R., "Building Rome in a day," in *Intl. Conf. on Computer Vision (ICCV)*, 2009.

[2] Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R., "Bundle adjustment in the large," in *European Conf. on Computer Vision (ECCV)*, pp. 29–42, 2010.

[3] Agarwal, S., Mierle, K., and Others, "Ceres solver." http://ceres-solver.org.

[4] Alcantarilla, P. F., Beall, C., and Dellaert, F., "Large-scale dense 3d reconstruction from stereo imagery," in *5th Workshop on Planning, Perception and Navigation for Intellegent Vehicles (PPNV)*, (Tokyo, Japan), November 2013. Best Paper Award.

[5] Alcantarilla, P. F., Ni, K., Bergasa, L. M., and Dellaert, F., "Visibility learning for large-scale urban environment," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, 2011.

[6] Alcantarilla, P. F., Oh, S. M., Mariottini, G. L., Bergasa, L. M., and Dellaert, F., "Learning visibility of landmarks for vision-based localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, IEEE, May 2010.

[7] Amestoy, P., Davis, T., and Duff, I., "An approximate minimum degree ordering algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 886–905, 1996.

[8] Bay, H., Tuytelaars, T., and Gool, L. V., "Surf: speeded up robust features," in *European Conf. on Computer Vision (ECCV)*, 2006.

[9] Beall, C. and Dellaert, F., "Appearance-based localization across seasons in a metric map," in *Planning, Perception and Navigation for Intelligent Vehicles Workshop*, 2014.

[10] Beall, C., Dellaert, F., Mahon, I., and Williams, S., "Bundle adjustment in large-scale 3D reconstructions based on underwater robotic surveys," in *OCEANS, 2011 IEEE-Spain*, pp. 1–6, IEEE, 2011.

[11] Beall, C., Lawrence, B., Ila, V., and Dellaert, F., "3D reconstruction of underwater structures," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[12] Bolles, R. and Fischler, M., "A RANSAC-based approach to model fitting and its application to finding cylinders in range data," in *Intl. Joint Conf. on AI (IJCAI)*, (Vancouver, BC, Canada), pp. 637–643, 1981.

[13] Bosse, M., Newman, P., Leonard, J., and Teller, S., "Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework," *Intl. J. of Robotics Research*, vol. 23, pp. 1113–1139, Dec 2004.

[14] Brubaker, M. A., Geiger, A., and Urtasun, R., "Map-based probabilistic visual self-localization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 38, pp. 652–665, April 2016.

[15] Cadena, C., Gálvez-López, D., Tardós, J. D., and Neira, J., "Robust place recognition with stereo sequences," *IEEE Trans. Robotics*, vol. 28, no. 4, pp. 871–885, 2012.

[16] Carlone, L., Kira, Z., Beall, C., Indelman, V., and Dellaert, F., "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[17] Choudhary, S. and Narayanan, P., "Visibility probability structure from sfm datasets and applications," in *European Conf. on Computer Vision (ECCV)*, pp. 130–143, Springer, 2012.

[18] Chum, O. and Matas, J., "Matching with PROSAC - progressive sample consensus," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[19] Churchill, W. and Newman, P., "Continually improving large scale long term visual navigation of a vehicle in dynamic urban environments," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 1371–1376, IEEE, 2012.

[20] Churchill, W. and Newman, P., "Practice makes perfect? managing and leveraging visual experiences for lifelong navigation," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 4525–4532, IEEE, 2012.

[21] Churchill, W. and Newman, P., "Experience-based navigation for long-term localisation," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, 2013.

[22] Cummins, M. and Newman, P., "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *Intl. J. of Robotics Research*, vol. 27, pp. 647–665, June 2008.

[23] Cummins, M. and Newman, P., "Highly scalable appearance-only slam - FAB-MAP 2.0," *Robotics: Science and Systems (RSS)*, 2009.

[24] Dayoub, F., Morris, T., Upcroft, B., and Corke, P., "Vision-only autonomous navigation using topometric maps," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1923–1929, IEEE, 2013.

[25] Dellaert, F., *Monte Carlo EM for Data Association and its Applications in Computer Vision*. PhD thesis, School of Computer Science, Carnegie Mellon, September 2001. Also available as Technical Report CMU-CS-01-153.

[26] Dellaert, F., "Square Root SAM: Simultaneous location and mapping via square root information smoothing," in *Robotics: Science and Systems (RSS)*, 2005.

[27] Dellaert, F., "Factor graphs and GTSAM: A hands-on introduction," Tech. Rep. GT-RIM-CP&R-2012-002, Georgia Institute of Technology, 2012.

[28] Dellaert, F. and Kaess, M., "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, pp. 1181–1203, Dec 2006.

[29] Dymczyk, M., Lynen, S., Cieslewski, T., Bosse, M., Siegwart, R., and Furgale, P., "The gist of maps - summarizing experience for lifelong localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 2767–2773, May 2015.

[30] Estrada, C., Neira, J., and Tardós, J., "Hierarchical SLAM: Real-time accurate mapping of large environments," *IEEE Trans. Robotics*, vol. 21, pp. 588–596, Aug 2005.

[31] Fischler, M. and Bolles, R., "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[32] Floros, G., van der Zander, B., and Leibe, B., "Openstreetslam: Global vehicle localization using openstreetmaps," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1054–1059, IEEE, 2013.

[33] Fraundorfer, F. and Scaramuzza, D., "Visual odometry: Part ii: Matching, robustness, optimization, and applications," vol. 19, no. 2, pp. 78–90, 2012.

[34] Garcia-Fidalgo, E. and Ortiz, A., "Vision-based topological mapping and localization methods: A survey," *Robotics and Autonomous Systems*, vol. 64, pp. 1–20, 2015.

[35] Geiger, A., Lenz, P., and Urtasun, R., "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, (Providence, USA), pp. 3354–3361, June 2012.

[36] Glover, A., Pepperell, E., Wyeth, G., Upcroft, B., and Milford, M., "Repeatable condition-invariant visual odometry for sequence-based place recognition," in *Australasian Conference on Robotics and Automation*, 2015.

[37] Harris, C. and Stephens, M., "A combined corner and edge detector," *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, August 1988.

[38] Hartley, R. I. and Zisserman, A., *Multiple View Geometry in Computer Vision*. Cambridge University Press, second ed., 2004.

[39] Jeong, Y., Nister, D., Steedly, D., Szeliski, R., and Kweon, I., "Pushing the envelope of modern methods for bundle adjustment," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2010.

[40] Johns, E. and Yang, G.-Z., "Dynamic scene models for incremental, long-term, appearance-based localisation," in *Proc. ICRA*, 2013.

[41] Johns, E. and Yang, G.-Z., "Feature co-occurrence maps: Appearance-based localisation throughout the day," in *Proc. ICRA*, 2013.

[42] Johns, E. and Yang, G.-Z., "Generative methods for long-term place recognition in dynamic scenes," *Intl. J. of Computer Vision*, vol. 106, no. 3, pp. 297–314, 2014.

[43] Johns, E. D. and Yang, G.-Z., "Pairwise probabilistic voting: Fast place recognition without ransac," in *European Conf. on Computer Vision (ECCV)*, pp. 504–519, Springer, 2014.

[44] Konolige, K. and Agrawal, M., "FrameSLAM: from bundle adjustment to realtime visual mapping," *IEEE Trans. Robotics*, vol. 24, no. 5, pp. 1066–1077, 2008.

[45] Konolige, K., Agrawal, M., Bolles, R., Cowan, C., Fischler, M., and Gerkey, B., "Outdoor mapping and navigation using stereo vision," in *Intl. Sym. on Experimental Robotics (ISER)*, Jul 2006.

[46] Krajnik, T., Cristóforis, P., Nitsche, M., Kusumam, K., and Duckett, T., "Image features and seasons revisited," in *Mobile Robots (ECMR), 2015 European Conference on*, pp. 1–7, IEEE, 2015.

[47] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W., "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, (Shanghai, China), May 2011.

[48] Lategahn, H., Schreiber, M., Ziegler, J., and Stiller, C., "Urban localization with camera and inertial measurement unit," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 719–724, IEEE, 2013.

[49] Lategahn, H. and Stiller, C., "City gps using stereo vision," *ICVES*, 2012.

[50] Leonard, J. and Newman, P., "Consistent, convergent, and constant-time SLAM," in *Intl. Joint Conf. on AI (IJCAI)*, 2003.

[51] Leutenegger, S., Furgale, P., Rabaud, V., Chli, M., Konolige, K., and Siegwart, R., "Keyframe-based visual-inertial slam using nonlinear optimization," in *Robotics: Science and Systems (RSS)*, 2013.

[52] Li, Y., Snavely, N., Huttenlocher, D., and Fua, P., "Worldwide pose estimation using 3d point clouds," pp. 15–29, 2012.

[53] Li, Y., Snavely, N., and Huttenlocher, D. P., "Location recognition using prioritized feature matching," pp. 791–804, 2010.

[54] Linegar, C., Churchill, W., and Newman, P., "Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation," pp. 90–97, 2015.

[55] Linegar, C. and Newman, P., "Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Stockholm, Sweden), May 2016.

[56] Lowe, D., "Distinctive image features from scale-invariant keypoints," *Intl. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[57] Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., and Milford, M. J., "Visual place recognition: A survey," *IEEE Trans. Robotics*, vol. 32, pp. 1–19, Feb 2016.

[58] Lowry, S., Milford, M., and Wyeth, G., "Transforming morning to afternoon using linear regression techniques," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3950–3955, IEEE, 2014.

[59] Maddern, W., Milford, M., and Wyeth, G., "Cat-slam: probabilistic localisation and mapping using a continuous appearance-based trajectory," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 429–451, 2012.

[60] McManus, C., Upcroft, B., and Newman, P., "Learning place-dependant features for long-term vision-based localisation," *Autonomous Robots*, vol. 39, no. 3, pp. 363–387, 2015.

[61] Milford, M. and Wyeth, G., "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1643 –1649, may 2012.

[62] Mühlfellner, P., Bürki, M., Bosse, M., Derendarz, W., Philippsen, R., and Furgale, P., "Summary maps for lifelong visual localization," *J. of Field Robotics*, 2015.

[63] Neubert, P. and Protzel, P., "Beyond holistic descriptors, keypoints, and fixed patches: Multiscale superpixel grids for place recognition in changing environments," *IEEE Robotics and Automation Letters*, vol. 1, pp. 484–491, Jan 2016.

[64] Ni, K., Steedly, D., and Dellaert, F., "Tectonic SAM: Exact; out-of-core; submap-based SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Rome; Italy), April 2007.

[65] Ni, K. and Dellaert, F., "Multi-level submap based slam using nested dissection," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[66] Nistér, D., Naroditsky, O., and Bergen, J., "Visual odometry," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 652–659, Jun 2004.

[67] Nistér, D. and Stewénius, H., "Scalable recognition with a vocabulary tree," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[68] Pascoe, G., Maddern, W., and Newman, P., "Direct visual localisation and calibration for road vehicles in changing city environments," in *Intl. Conf. on Computer Vision (ICCV)*, pp. 9–16, 2015.

[69] Paz, L. M., Pinies, P., Tardós, J. D., and Neira, J., "Large scale 6DOF SLAM with stereo-in-hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[70] Pepperell, E., Corke, P., and Milford, M., "Routed roads: Probabilistic vision-based place recognition for changing conditions, split streets and varied viewpoints," *Intl. J. of Robotics Research*, 2016.

[71] Pepperell, E., Corke, P. I., and Milford, M. J., "All-environment visual place recognition with smart," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1612–1618, IEEE, 2014.

[72] Piniés, P. and Tardós, J. D., "Large-scale slam building conditionally independent local maps: Application to monocular vision," *IEEE Trans. Robotics*, vol. 24, no. 5, pp. 1094–1106, 2008.

[73] Raguram, R., Frahm, J.-M., and Pollefeys, M., "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," in *Computer Vision–ECCV 2008*, pp. 500–513, Springer, 2008.

[74] Raguram, R., Tighe, J., and Frahm, J.-M., "Improved geometric verification for large scale landmark image collections.," in *BMVC*, pp. 1–11, 2012.

[75] Ros, G., Ramos, S., Granados, M., Bakhtiary, A., Vazquez, D., and Lopez, A. M., "Vision-based offline-online perception paradigm for autonomous driving," in

Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on, pp. 231–238, IEEE, 2015.

[76] Sattler, T., Havlena, M., Radenovic, F., Schindler, K., and Pollefeys, M., "Hyperpoints and fine vocabularies for large-scale location recognition," in *Intl. Conf. on Computer Vision (ICCV)*, December 2015.

[77] Sattler, T., Leibe, B., and Kobbelt, L., "Fast image-based localization using direct 2d-to-3d matching," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 667–674, IEEE, 2011.

[78] Sattler, T., Leibe, B., and Kobbelt, L., "Improving image-based localization by active correspondence search," in *Computer Vision–ECCV 2012*, pp. 752–765, Springer, 2012.

[79] Scaramuzza, D. and Fraundorfer, F., "Visual odometry: Part i the first 30 years and fundamentals," 2011.

[80] Senlet, T. and Elgammal, A., "A framework for global vehicle localization using stereo images and satellite and road maps," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 2034–2041, IEEE, 2011.

[81] Shi, J. and Tomasi, C., "Good features to track," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, 1994.

[82] Snavely, N., Seitz, S., and Szeliski, R., "Photo tourism: Exploring photo collections in 3D," in *SIGGRAPH*, pp. 835–846, 2006.

[83] Stylianou, A., Abrams, A., and Pless, R., "Characterizing feature matching performance over long time periods," in *IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 892–898, IEEE, 2015.

[84] Torii, A., Sivic, J., and Pajdla, T., "Visual localization by linear combination of image descriptors," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 102–109, IEEE, 2011.

[85] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A., "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice* (Triggs, W., Zisserman, A., and Szeliski, R., eds.), LNCS, pp. 298–375, Springer Verlag, Sep 1999.

[86] Ullman, S., *The interpretation of visual motion*. The MIT press, Cambridge, MA, 1979.

[87] Valgren, C. and Lilienthal, A. J., "Sift, surf & seasons: Appearance-based long-term localization in outdoor environments," *Robotics and Autonomous Systems*, vol. 58, no. 2, pp. 149–156, 2010.

[88] Vysotska, O., Naseer, T., Spinello, L., Burgard, W., and Stachniss, C., "Efficient and effective matching of image sequences under substantial appearance changes exploiting gps priors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 2774–2779, IEEE, 2015.

[89] Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J., "A comparison of loop closing techniques in monocular slam," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[90] Xu, D., Badino, H., and Huber, D., "Topometric localization on a road network," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3448–3455, IEEE, 2014.